Computing Updates in Description Logics

Dissertation

zur Erlangung des akademischen Grades
Doktoringenieur (Dr.-Ing.)

vorgelegt an der
Technischen Universität Dresden
Fakultät Informatik

eingereicht von
Dipl.-Inform. Hongkai Liu
geboren am 10. April 1980 in Anshan, Liaoning, China

verteidigt am 28. Januar 2010

Gutachter:

Prof. Dr.-Ing. Franz Baader
Technische Universität Dresden

Prof. Gerhard Lakemeyer, Ph.D.
Rheinisch-Westfälische Technische Hochschule Aachen

Dresden, im Februar 2010

Acknowledgements

I am deeply indebted to Franz Baader and Carsten Lutz, my thesis advisors, for their valuable guidance of my researching work and constant support of all technical problems. Without either of them, this thesis would not exist. I am grateful to Maja Miličić and Frank Wolter who have also helped me a lot in the last four years.
I would like to thank Conrad Drescher, Boontawee Suntisrivaraporn, and Rafael Peñaloza for proofreading the preliminary version of my thesis. I would also like to thank all members of the chair for automata theory. It is a pleasure to work with you. Many thanks go to our secretary, Kerstin Achtruth, for her support in countless matters. I am also thankful to Anni-Yasmin Turhan for her encouragement.
I would like to express my gratitude to my mother for her endless love and unconditional support. No one is prouder of this thesis than she is. This thesis is dedicated to her.

Contents

Chapter 1

Introduction

Knowledge representation is an area in artificial intelligence that concentrates on the design of formalisms which can explicitly represent knowledge about a particular domain, and the development of reasoning methods for inferring implicit knowledge from the represented explicit knowledge. Description Logics form a family of knowledge representation formalisms which can be used to represent and reason with conceptual knowledge about a domain of interest [BCM$^+$03]. The knowledge represented by Description Logics is mainly static. In many applications, the domain knowledge is dynamic. This observation motivates the research on how to update the knowledge when changes in the application domain take place. This thesis is dedicated to the study of updating knowledge, more precisely, assertional knowledge represented in Description Logics. We start with introducing Description Logics in Section 1.1 and proceed by illustrating the kinds of updates considered in this thesis in Section 1.2. We introduce the integration of updates into linear temporal Description Logics in Section 1.3. The structure of this thesis is outlined in Section 1.4.

1.1 Knowledge Representation with Description Logics

Description Logics (DLs) evolve from early knowledge representation formalisms such as *semantic networks* [Qui67] and *frames* [Min74]. In these systems, knowledge is represented by characterizing classes of objects and relationships between them. A semantic network essentially is a directed labeled graph in which vertices represent *classes* of objects (also called *concepts*) and edges represent *relations* between them. The counterparts of these notions of concepts and relations in frame-based systems are referred to as *frames* and *slots*. The main problem with both semantic networks and frames is that they lack a formally defined semantics. Therefore, the standard meaning of the knowledge and the results of reasoning provided by the systems are strongly dependent on the implementation strategies. As a consequence, for the same input different systems may return different results [Sow92]. Utilizing logic to supply a formal semantics of the knowledge avoids this ambiguity and clarifies how reasoning services have to interpret the knowledge in the domain of discourse. The earliest logic-based knowledge representation formalism is KL-ONE [BS85], which inherits the

notions of concepts, roles, and individuals from semantic networks and frames, and provides them with a first-order logic semantics. Although reasoning in the logic used in KL-ONE proves to be undecidable [SS89], the foundations of systax, semantics, and reasoning tasks of "logic-based concept languages" have been established. By making a tradeoff between the expressivity of KL-ONE-like languages and the computational complexity of reasoning, Description Logics are developed. Most DLs can be thought of as decidable fragments of first-order logic.

A DL knowledge base usually consists of two components. The first one is an assertional box, ABox for short, which is a world description about specific individuals. With an ABox, one can assert that some individual exhibits the attribute described by a concept and that two individuals are connected with a relation (also called *role* in DLs). For instance, we can state that John has a daughter or a child named Peter, Mary has a brother named Peter, and Mary has at least 2 brothers who are teachers in the following way:

$$\begin{aligned} &\mathsf{JOHN} : \exists \mathsf{hasChild}.(\mathsf{Female} \sqcup \{\mathsf{PETER}\}) \\ &\mathsf{hasBrother}(\mathsf{MARY}, \mathsf{PETER}) \\ &\mathsf{MARY} : (\geqslant 2\ \mathsf{hasBrother}\ \mathsf{Teacher}) \end{aligned}$$

Each of the expressions above is called an *assertion*. An ABox is an incomplete description of the world, i.e., there may be some assertions of which the truth value cannot be determined by the assertions represented in the ABox. For instance, from the above assertions, we cannot conclude that Mary is John's daughter; likewise, we cannot disprove it.

The second component of a DL knowledge base is a terminological box, TBox for short, which describes relationships between concepts or roles. For example, the following statements express that a father is exactly a man who has a child, and that every woman who has a son is a parent.

$$\begin{aligned} \mathsf{Father} &\equiv \mathsf{Man} \sqcap \exists \mathsf{hasChild}.\mathsf{Person} \\ \mathsf{Woman} \sqcap \exists \mathsf{hasChild}.\mathsf{Male} &\sqsubseteq \mathsf{Parent} \end{aligned}$$

The first expression is a *concept definition* and the second one is a *general concept inclusion (GCI)*. An *acyclic TBox* contains only concept definitions without cyclic definitions such as

$$\mathsf{Human} \equiv \exists \mathsf{hasChild}^{-}.\mathsf{Human}$$

in which the concept name Human is used to define itself (A human is defined as a child of a human). Acyclic TBoxes are mainly used as a way to introduce abbreviations of complex concepts. A *general TBox* consists of GCIs. General TBoxes have more expressive power than acyclic TBoxes.

The semantics of DLs is given in a model-theoretic way. The meaning of a concept, a role, an assertion, etc. is formally defined by interpretations, which, in contrast to ABoxes, provide complete descriptions of the world. An interpretation $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non-empty set (the domain of $\mathcal{I}$) and $\cdot^{\mathcal{I}}$ is a function which interprets concepts, roles, and individuals just as unary predicates, binary predicates, and constants, respectively, in first-order logic. More specifically, $\cdot^{\mathcal{I}}$ assigns a subset

of $\Delta^{\mathcal{I}}$ to every concept name, a binary relation on $\Delta^{\mathcal{I}}$ to every role name, and an element of $\Delta^{\mathcal{I}}$ to every individual name. The extension of $\cdot^{\mathcal{I}}$ to complex concepts and roles is defined in a straightforward way. An interpretation is a *model* of an assertion if the interpretation of the individual, or the pair of individuals is in the interpretation of the corresponding concept or role. A model of a concept definition $A \equiv C$ is an interpretation that maps A and C to the same subset of the interpretation domain. Similarly, an interpretation is a model of a GCI $C \sqsubseteq D$ if it maps C to a subset of the set it maps D to. ABoxes and TBoxes may be seen as conjunctions of expressions they consists of, i.e., an interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ (a TBox $\mathcal{T}$, respectively) if $\mathcal{I}$ is a model of *every* element of $\mathcal{A}$ ($\mathcal{T}$, respectively).

Reasoning in DLs is also formally defined. For instance, the *consistency problem* of an ABox $\mathcal{A}$ is to check whether $\mathcal{A}$ has a model. The computational complexity of reasoning depends on the expressivity of the DL under consideration. The expressivity of a DL is determined by the constructors used to build concepts and roles. The smallest propositionally closed DL $\mathcal{ALC}$ [SS91] provides the following concept constructors: negation ($\neg$), conjunction ($\sqcap$), disjunction ($\sqcup$), value restriction ($\forall$), and existential restriction ($\exists$). DLs are closely related to Modal Logics [BdRV01]. In [Sch91], the DL $\mathcal{ALC}$ has been shown to be a notational variant of the Modal Logic $\mathbf{K}_m$. By extending $\mathcal{ALC}$ with qualified number restriction ($(\geqslant n\ r\ C)$ and $(\leqslant n\ r\ C)$), inverse role (r^-), and nominals ($\{a\}$), we get the DL $\mathcal{ALCQIO}$, which is the core language of the Web Ontology Language (OWL) [HPS03] recommended by the World Wide Web Consortium to represent knowledge in the Semantic Web [BLHL01, BHS05, HPSMW07]. In this thesis, we mainly focus on DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ to investigate the computation of updates. The standard reasoning tasks such as consistency in those DLs have been investigated in the past twenty years. They are supported by state-of-the-art DL reasoners, such as Pellet [SPG$^+$07], FaCT++ [TH06], and RacerPro [HM01], etc.

1.2 Updates

In this section, the first part is dedicated to introducing the problem of updating logical theories, to illustrating how it can be applied to reasoning about action, and to presenting motivations of computing updates in DLs. In the second part, we explain why we focus on updating ABoxes, give the intuition behind various kinds of updates considered in this thesis, and present an overview of the results. In the third part, we discuss some related work on reasoning about action and computing updates in DLs.

Updating Logical Theories

A *logical theory* is a finite set of logical formulas.[1] The problem of updating logical theories has been studied for a long time in both the database and artificial intelligence community. As discussed in the previous section, a logical theory can be used to

[1] In this section, we use terminology from first-order logic, such as literal and formula. Cf. [Dav93] for their formal definitions. A knowledge base in DLs can also be viewed as a logical theory since most DLs are fragments of first-order logic.

describe what is known about the state of the world. As neither data nor knowledge bases are static entities representing an unchanging monolithic domain once created, solutions to the update problem are crucial for maintaining both data and knowledge bases [Win90, Rei82, EG92, FUV83, Gär88].

Updating a logical theory can be formulated as follows. Assume that we are given a logical theory T representing a certain domain of interest and a change that has occurred within that domain represented by a formula φ. How should the theory T be modified so that it represents the domain after the change expressed by φ? This is the *problem of updating logical theories*. We call φ the *update* and the modified theory $T * \varphi$ the *update of T with φ*.

Various, partly conflicting, proposals have been made to tackle the update problem [Gin86, FUV83, Dal88, Bor85, Web86, KM89, Win88]. The differences between the proposals as well as almost all research problems in the field are caused by incomplete information which has to be dealt with if

- the update φ is nondeterministic (because it contains, say, a disjunction or an existential quantifier) [Lin96], or
- the updated theory must satisfy additional domain constraints which are formulas satisfied in each state of the world no matter which changes take place [Lin95, Thi97].

In fact, if the update φ consists of a conjunction of ground literals only and no domain constraints must be met by the updated theory, it appears to be a consensus that the following model-theoretic definition of the updated theory $T * \varphi$ is, at least from a theoretical viewpoint, the most satisfactory one: $T * \varphi$ should be defined as the theory of the set M of all models which are obtained from models of T by changing the interpretation of the atoms occurring in φ in such a way that φ becomes true and leaving the interpretation of all other atoms unchanged. Each model of T is thought of as a possible complete description of the world. The changes in the world caused by φ are realized by changing the possible models of the world. Because it is not clear which model describes the actual state of the world, the updated theory should capture all changed models. We call this semantics of update the *Winslett's possible model approach semantics*, which has initially been proposed in [Win88] and further elaborated e.g., in [Win90, BH93, Her96, DLMB98]. Sometimes, we also call this semantics the Winslett's semantics, or the PMA semantics for short. For the logical languages considered so far by the database and artificial intelligence community, hardly any problem arises in this case.

As addressed in [KM92], updating a logical theory can be applied to reasoning about action. The most prominent two action theories are the Situation Calculus [Rei01] and the Fluent Calculus [Thi05b]. In both action theories, a dynamic application domain is axiomatized as a logical theory to express the initial state of the world, action's pre-conditions and post-conditions, and domain constraints. The preconditions of an action describe under which conditions the action is executable while the post-conditions are the changes which the action is going to make in the world

when the action is applied. When the changes in an action's post-condition take place by executing the action, the axiom which represents the initial state of the world can be updated by the post-conditions of the action for representing the new state of the world. This procedure is called *progression* in the reasoning about action literature.

A fundamental problem in reasoning about action is *projection*, which is concerned with determining whether or not a formula φ holds after a finite sequence of actions has been applied, given a description of the pre-conditions and post-conditions of the actions and what the world is like initially. Progression is one of two basic mechanisms in action theory to do reasoning about action. The other one is called *regression*. Both regression and progression are mechanisms that can be used to solve the projection problem. Regression rewrites the formula φ into a new formula φ' such that φ holds after applying the actions iff φ' is entailed by the initial world. Progression, instead of rewriting φ, updates the description T of the initial world into a new description T' such that φ holds after applying the actions iff φ follows from T'. The action programming language Golog [LRL$^+$97] which is based on the Situation Calculus adopts regression to solve the projection problem, while the action programming language FLUX [Thi05a] which is based on the Fluent Calculus uses progression to do reasoning about action. One advantage of progression is that the updated theory can be used to decide the entailment for different formulas. If regression is adopted, a new rewritten formula has to be calculated when the entailment of a new formula needs to be decided. Moreover, when the history of actions is very long, regression becomes expensive. For those reasons, progression in the Situation Calculus has been investigated [LR97, LL09]. However, both of the action calculi are formulated in first- or higher-order logic and thus they are so expressive that some restrictions on the expressiveness of the theory have to be applied in order to make progressing a knowledge base or updating a theory in those action calculi feasible [LL09, Thi05a].

In DLs, the reasoning support provided by systems such as Pellet, FaCT++, and RacerPro mainly concerns static knowledge bases and the support provided for reasoning about dynamic domains is limited and rather ad-hoc without theoretical foundation. Action formalisms based on DLs were first proposed in [BLM$^+$05]. There, the projection problem is solved by an approach similar to regression and it is reduced to checking logical consequences of ABoxes w.r.t. acyclic TBoxes. In this thesis, we focus our attention on computing updates in DLs and thus provide a progression mechanism in DLs to do reasoning about action.

All of the work on regression and progression in DLs establishes the theoretical foundations to build DL-based action theories in which the expressiveness and decidability of DLs are naturally inherited. On the one hand, DL-based action theories are more expressive than the action theories based on propositional logic such as STRIPS [Byl94]. On the other hand, they do not suffer from undecidable reasoning like both the Situation Calculus and the Fluent Calculus. Regardless of adopting regression or progression, the projection problem is reduced to checking logical consequences. In the implementation based on such a DL-based action theory, DL reasoners can be employed to decide logical consequences.

Updates in DLs

As introduced in Section 1.1, a DL knowledge base consists of a TBox representing relationships of concepts or roles using the names to model the domain of interest, and an ABox representing properties of individuals in terms of names. Thus, one can distinguish at least three distinct notions of updates for DL knowledge bases:

- the update problem *for TBoxes* without considering a particular ABox. In this case one is interested in updates regarding the meaning of the names used to represent a domain but not in changes to the underlying data. This topic is outside the scope of this thesis. Observe, however, that it appears plausible to assume that changes regarding the names are mostly required not because the meaning of the names has changed but because the ontology engineer has changed his/her beliefs regarding the meaning of the names. It follows that, when changing TBoxes, the belief revision problem [AGM85, KM92] is more important than the update problem. Work on this can be found in [Flo06, FPA05, FPA06, HWKP06].

- the update problem *for ABoxes* without considering a particular TBox. It is a basic form of the update problem and the main task of this thesis is to investigate this problem.

- the update problem *for ABoxes* where the TBox is regarded as a set of domain constraints which remains invariant under any changes of the world. This is evidently an extension of the previous case. According to expressiveness, TBoxes are divided into acyclic TBoxes and general TBoxes. The former one is mainly used to introduce concept names as abbreviations of complex concepts. Actually, most of the results in this thesis can be extended to this case. Some work related to this has been done in [LLMW06c]. Alternatively, we can compile acyclic TBoxes by unfolding these abbreviations [BCM$^+$03], so that updates only operate on ABoxes. For general TBoxes, there is still no satisfactory semantics for update even in the DL $\mathcal{ALC}$ [BLM$^+$05]. Moreover, under the Winslett's semantics, allowing general TBoxes as domain constraints leads to undecidable reasoning about action. We will discuss this in more detail in the third part "related work" of this section.

Hereby, the main focus of this thesis is updating ABoxes in DLs in the following context:

- We adopt the Winslett's PMA semantics and allow only for the updates that consist of a conjunction of ground literals;

- We do not allow for TBoxes at all, regardless of whether they are interpreted as domain constraints or as dynamic data.

It turns out that a number of new and challenging problems arise when the Winslett's semantics is applied to updating ABoxes. To discuss them recall that from the informal description of the Winslett's semantics above we obtain, given the set M of models

of an ABox $\mathcal{A}$ and an update $\mathcal{U}$ consisting of ground literals, the set M' of updated models. Our goal is to capture this set of models syntactically using an ABox $\mathcal{A}'$. If the set of models of some ABox $\mathcal{A}'$ coincides with M', then $\mathcal{A}'$ is a *logical update* of $\mathcal{A}$ with $\mathcal{U}$. One question is which DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ have enough expressive power to do this? Unfortunately, the answer is mostly negative. Either we can show that a logical update does not exist, e.g., in $\mathcal{ALC}$, $\mathcal{ALCO}$ and $\mathcal{ALCQIO}$, or we do not know whether it exists, e.g., in $\mathcal{ALCQI}$. This is a major problem, indeed. DLs have been designed so as to find an optimal compromise between expressiveness and computational behavior. Moving from one DL to another often has a major impact on the performance of reasoners both from a theoretical and practical viewpoint. So, we explore new ways of expressing updates in DLs for which reasoning support is available. To this end, we consider, apart from exact updates, weaker forms of expressing updates: an ABox $\mathcal{A}'$ is

- an *approximate update* of $\mathcal{A}$ with $\mathcal{U}$ if, and only if, exactly the ABox assertions entailed by M' follow from $\mathcal{A}'$;
- a *projective update* of $\mathcal{A}$ with $\mathcal{U}$ if, and only if, $\mathcal{A}'$ captures the set of reducts, to the names used in $\mathcal{A}$ and $\mathcal{U}$, of interpretations in M' (in this case $\mathcal{A}'$ might contain additional names);
- a *approximate projective update* of $\mathcal{A}$ with $\mathcal{U}$ if, and only if, exactly the ABox assertions entailed by M' over the names of $\mathcal{A}$ and $\mathcal{U}$ follow from $\mathcal{A}'$ (again $\mathcal{A}'$ might contain additional names).

It is easy to see that a logical update matches also the other three kinds of updates and that any approximate update or projective update is also an approximate projective update. Logical updates are obviously the most desirable form of updates as they describe exactly the set of updated models without using additional names. However, projective updates are a reasonable alternative. In contrast to approximate updates, logical and projective updates are invariant under moving to a more expressive DL. More precisely, if $\mathcal{A}'$ is a logical (projective, respectively) update in the DL $\mathcal{L}$, then it is a logical (projective, respectively) update in any DL $\mathcal{L}'$ extending $\mathcal{L}$. At the cost of adding additional names, projective updates can be regarded as a reasonably well-behaved alternative to logical updates. Obviously, if neither logical nor projective update exists, then approximate or even approximate projective updates are still very useful, as they completely describe the logical consequences of an update in a given DL.

Which DLs under consideration are expressive enough to describe weaker forms of updates? For approximate updates, the answer is no better than for logical updates. Positive results emerge when we shift to projective updates: For every DL between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}$, projective updates exist. For DLs without nominals (between $\mathcal{ALC}$ and $\mathcal{ALCQI}$), either a projective update does not exist, e.g., in $\mathcal{ALC}$ and $\mathcal{ALCI}$, or its existence is unknown, e.g., in $\mathcal{ALCQI}$. We believe that nominals are essential even for approximate projective updates, although a proof of this claim is still missing.

By showing that logical updates do not necessarily exist for $\mathcal{ALCO}$, it is revealed that adding nominals is not enough to capture logical updates. In order to express

logical updates, we further increase the expressivity of ABoxes. One way to achieve this is to allow for the @ constructor, which is known from hybrid logic [AdR01]. For every DL between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}$ extended with the @ constructor, logical updates exist. For most of those extended DLs without nominals, e.g., $\mathcal{ALC}^{@}$, the existence of a logical update does not hold. The other option to describe logical updates is to allow for Boolean operators on ABox assertions. ABoxes formulated this way are called *Boolean ABoxes*. It turns out that for every DL $\mathcal{L}$ between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}$, the expressive power of non-Boolean $\mathcal{L}^{@}$-ABoxes is exactly the same as that of Boolean $\mathcal{L}$-ABoxes. For convenience, the construction of logical updates we present uses both the @ constructor and Boolean operators on ABox assertions. Although neither of them is supported directly by DL reasoners, we can use the reduction approach [ABHM03, Bon07] or the DPLL(T) approach [NORCR07] to deal with reasoning with logical updates. We can also resort to the reasoner Spartacus [Göt09] for hybrid logic.

We summarize the relationship of the expressiveness and updates' existence as follows:

- Given only the @ constructor or only nominals, approximate updates do not exist. This implies the same for logical updates.
- With nominals only, projective updates exist. This implies the same for approximate projective updates. Without nominals, projective updates do not exist independently of the @ constructor.

Another important problem addressed in this thesis is how big the constructed logical update $\mathcal{A}'$ is. In the worst case, the size of $\mathcal{A}'$ is exponential in the size of the whole input, i.e., in the size of $\mathcal{A}$ plus the size of $\mathcal{U}$. Moreover, we show that an exponential blowup cannot be completely avoided by proving that, even in the case of propositional logic, the size of logical updates is not polynomial in the size of the input unless every PTime-algorithm is LogTime-parallelizable, which is believed to be similarly unlikely as PTime = NP [Pap94]. In contrast to the results by Cadoli et al. [CDLS99], our result even applies to the restricted form of updates, i.e., updates in propositional logic where the update is a conjunction of literals. Thus, our argument provides further evidence for the claims in [CDLS99], where it is shown that, with unrestricted updates, an exponential blowup in the size of the update cannot be avoided unless the first levels of the polynomial hierarchy collapse.

In the update literature, an exponential blowup in the size of the update only is viewed as much more tolerable than an exponential blowup in the size of the original ABox since the former tends to be small compared to the latter. We believe that the exponential blowup in the size of the original ABox cannot be avoided. While the proof is left as an open problem, we exhibit two ways around the blowup. The first, and more important one, is to consider projective updates instead of logical updates. We present a construction of projective updates of polynomial size both in the original ABox and in the update. They can be computed in polynomial time. The second is to move to DLs which do not only contain nominals but also allow Boolean operators on

roles, thus eliminating the asymmetry between concepts and roles found in standard DLs. The size of a logical update is polynomial in the size of the original ABox (still exponential in the size of the update). This is less interesting because it involves a move to DLs for which developing reasoning support is still an open research problem.

Related Work

The literature on updates in databases and artificial intelligence is too vast to be discussed here in general. We concentrate instead on recent work related to updates in DLs.

The notions of approximate updates and approximate projective updates were first proposed in [Bon07] in which the existence of updates is investigated for the DL $\mathcal{ALC}$ combined with nominals and the @ constructor. In the present thesis, we extend the results there to more expressive DLs. We also enhance some positive results. For instance, Theorem 54 in [Bon07] shows that $\mathcal{ALCO}$ has approximate projective updates by encoding the @ constructor in the logical update which is an $\mathcal{ALCO}^{@}$-ABox with additional names. The size of the approximate projective update obtained in this way may be exponential in the size of the input. Here, we give a polynomial construction of even stronger, projective updates, directly from the input.

In [DT07], based on updating ABoxes, DLs are integrated into a general action theory, the Fluent Calculus. This yields a fragment of the the Fluent Calculus in which some reasoning tasks about action are decidable, shows that the Winslett's semantics is captured by state update axioms in the Fluent Calculus, and provides theoretical foundations for integrating DL reasoning into action programming language like Golog and Flux.

Closely related to the work presented in this thesis are [GLPR06, GLPR07], considering the problem of updating knowledge bases expressed in DLs from the DL-Lite family. These are lightweight DLs which are tailored toward capturing conceptual modeling constructs while keeping reasoning, in particular conjunctive query answering, tractable. In [GLPR06] it is proved that in DL-Lite$_F$, updates exist when ABoxes are extended by assertions of the form $C(x)$, where x is a variable. In the terminology introduced above, this means that projective updates exist for DL-Lite$_F$. This is shown even for ABox updates of DL-Lite$_F$ knowledge bases with general TBoxes as domain constraints under the Winslett's possible model approach semantics. Giacomo et al. consider the existence of approximate ABox updates for DL-Lite$_F$ knowledge bases with general TBoxes in [GLPR07]. This study is motivated by the fact that, as mentioned above, in general no exact updates exist. It is shown that approximate ABox of polynomial size (in the knowledge base and the update) always exist and can be computed in polynomial time. Thus, DL-Lite$_F$ exhibits a much better behavior regarding updates than the DLs considered in this thesis which have more expressive power.

It is worth noting that it is not possible to generalize this approach to expressive DLs like the ones considered in this thesis. For example, the following result follows immediately from Theorem 22 in [BML$^+$05]: There exists an $\mathcal{ALCQI}$-ABox $\mathcal{A}$ and an update $\mathcal{U}$ which consists of one assertion $a : C$ where C is a *complex* $\mathcal{ALCQI}$-concept

such that it is undecidable whether an $\mathcal{ALCQI}$-assertion $a : D$ is true after the update of $\mathcal{A}$ with $\mathcal{U}$ (D contains only names from $\mathcal{A}$ and $\mathcal{U}$). It follows that, no update (any of the four kinds of updates) of $\mathcal{A}$ with $\mathcal{U}$ can be effectively computed in any decidable extension of $\mathcal{ALCQI}$. Allowing GCIs as domain constraints indirectly makes updates contain complex concepts and thus also leads to undecidable reasoning about action.

As mentioned before, the projection problem in DLs was first investigated in [BLM$^+$05]. It is shown there that for every DL $\mathcal{L}$ between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$, the projection problem in $\mathcal{L}$ has the same complexity as the (in)consistency problem in $\mathcal{LO}$ which is obtained by extending $\mathcal{L}$ with nominals. [LLM08] proves that in the DL $\mathcal{EL}$ in which reasoning tasks such as consistency checking [BBL05, Sun09] are tractable, the projection problem is already as hard as the one in $\mathcal{ALC}$ (PSPACE-complete). Even when allowing only empty TBoxes, projection in $\mathcal{EL}$ is co-NP-complete. The planning problem, which is to look for a sequence of actions in order to achieve a given goal, is investigated in these DL-based action formalisms [Mil07]. Planning is solved there with the help of projection.

In [LLMW06a, LLMW06b], the DL-based action formalisms from [BLM$^+$05] are extended to allow general TBoxes as domain constraints. In order to deal with the ramification problem, which is concerned with indirect effects of an action, coming with general TBoxes, more information has to be offered when actions are defined. It is shown that projection is still decidable for $\mathcal{ALCQIO}$. One negative result about those extended action formalisms is that consistency of an action is undecidable, i.e., it is undecidable whether there exists an updated interpretation for every interpretation under the semantics of actions defined there.

1.3 Integration of Updates into Linear Temporal DLs

The projection problem is to check whether or not some assertion φ holds after finitely many updates.[2] In other words, the assertion is verified in a time point in the future determined by updates. In this way, projection can be viewed as verification of a temporal property. In the literature, temporal extensions of DLs are employed to describe more expressive temporal properties [BGL08, WZ00]. In those works, DLs are combined with propositional temporal logics using a two-dimensional semantics, where one dimension is for time and the other for the DL domain. In the consecutive time points, the DL domain treats concept names and role names either independently or immutably, i.e., the interpretations of those names either change freely or stay the same [BGL08]. In the latter case, they are called *rigid names*. Following this approach, we integrate ABox updates into linear temporal logic (LTL) [VW94, Pnu81]. More specifically, the changes occurring in the DL domain are caused by application of updates and thus *the semantics of updates is respected in the DL domain along the linear time points*. Thus, all the names in the DL domains are either completely rigid, e.g., the names not occurring in the updates, or partially rigid, e.g., the names

[2]Updates can be viewed as actions without pre-conditions. Actually, the pre-conditions of actions will not affect the answer to the projection problem. Since this thesis mainly concentrates on updates, in order to keep uniformity, we henceforth use updates instead of actions to refer to the input of projection.

on which interpretations minimally changes according to updates. On the one hand, this extends the projection problem in the sense that it can inspect a property using more expressive LTL temporal operators, such as the until operator. For example, the formula

$$\Diamond(\mathsf{CA931} : \exists \mathsf{hasAccessTo.Runway}) \tag{1}$$

says that there is a time point when the airplane CA931 will have access to a runway. On the other hand, we can use an *infinite* sequence of updates to specify a course of processes during which the prospected property could happen.

We follow the notion of $\mathcal{ALC}$-LTL formulas in [BGL08] where temporal operators are allowed only in front of ABox assertions and extend assertions in $\mathcal{ALC}$ to the ones in any DL between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$. Such a formula is called a *DL-LTL formula*. Equivalently, a DL-LTL formula can be obtained by replacing propositional letters in a propositional LTL formula with an ABox assertion.

We focus on infinite sequences of updates of the form $\alpha_1 \cdots \alpha_m(\beta_1 \cdots \beta_n)^\omega$. Such a sequence of updates is called a *Büchi sequence of updates* since for every Büchi automaton $\mathcal{B}$, if the language accepted by $\mathcal{B}$ is not empty, then $\mathcal{B}$ accepts a word with this form [TB73]. For instance, suppose that $\beta_1 \cdots \beta_n$ are the repeatedly applied updates after the update sequence $\alpha_1 \cdots \alpha_m$ has been applied for initialization in the air traffic control system in some airport. Then the formula (1) holds in the system means that execution of $\alpha_1 \cdots \alpha_m(\beta_1 \cdots \beta_n)^\omega$ will ensure the airplane will have an opportunity to use a runway. This extends the projection problem in the sense that it checks whether a DL-LTL formula holds when a Büchi sequence of updates is applied to an ABox. This problem can further be generalized to the one that allows for sequences of updates accepted by a Büchi automaton instead of a fixed sequence of updates. Suppose that Σ is a set of updates. In the more general problem, we are offered a way to describe any infinite sequence of updates in an ω-regular language over Σ instead of only Büchi sequences of updates since the class of languages accepted by Büchi automata coincides with the class of ω-regular languages [BK08]. In this thesis, we will present decision procedures for verifying DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates (or Büchi automata, respectively).

1.4 Structure of the Thesis

In this section, we outline the structure of the present thesis.

- In Chapter 2 we formally introduce DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}^{@}$: concepts, roles, individuals, assertions, ABoxes, acyclic TBoxes, and their semantics. The inference problems related to this thesis are also defined. We discuss the expressive power of the @ constructor and Boolean ABoxes because both of them will be used in the construction of logical updates. Based on the Winslett's possible model approach semantics, four kinds of updates of ABoxes, namely, logical updates, approximate updates, projective updates, and approximate projective updates are introduced. We address some basic properties of those updates that will be used later on in the thesis.

- In Chapter 3 we concentrate on logical updates. A construction which explicitly introduces both nominals and the @ constructor is presented. For an ABox $\mathcal{A}$ and an update $\mathcal{U}$, the size of the constructed logical update $\mathcal{A}'$ is in the worst case exponential both in the size of $\mathcal{A}$ and in the size of $\mathcal{U}$. We show that totally avoiding the exponential blowup on the input is not possible unless the complexity classes PTime and NC coincide. Two ways to avoid the exponential blowup in the size of the ABox are exhibited: to allow only for concept literals in updates and to compute logical updates in a more expressive DL with more role constructors. We show that the blowup produced by iterated updates is not worse than the blowup produced by a single update.

- In Chapter 4 we concentrate on projective updates. A construction which explicitly introduces nominals is presented. For an ABox $\mathcal{A}$ and an update $\mathcal{U}$, the size of the computed projective update is polynomial both in the size of $\mathcal{A}$ and in the size of $\mathcal{U}$. The direct application of this construction to updating an ABox iteratively leads to an exponential blowup in the size of $\mathcal{A}$. Instead of computing projective updates iteratively, we exhibit a direct construction of a projective update of the original ABox with a finite sequence of updates. The updated ABox constructed in this way is bounded polynomially in the size of the input. We illustrate how to employ these construction in practice.

- In Chapter 5 we show that the constructions of logical updates and projective updates given in Chapter 3 and Chapter 4 introduce only necessary constructors. Dropping either the @ constructor or nominals, even approximate updates are not expressible. Without nominals, projective updates are not expressible.

- In Chapter 6 we present experimental results. We compare the construction time and the size of logical update and projective update for a batch of randomly generated ABoxes and updates. Three approaches to deal with consistency checking of Boolean ABoxes are compared on randomly generated Boolean ABoxes and the logical updates generated before. The algorithm of solving the projection problem in [BLM$^+$05] is implemented. We compare its performance with logical updates and projective updates against the same assertion φ which is the checked assertion in the input of the projection problem. We also test those implementations on updating ABoxes with iterative updates and compare different approaches of computing the updated ABoxes and reasoning with them.

- In Chapter 7 we introduce the inference problems to verify DL-LTL formulas: the satisfiability problem and the validity problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates. According to different forms of updates in the input, two algorithms are presented to solve those problems. One of them even works for the more general problems: the satisfiability problem and the validity problem of DL-LTL formulas w.r.t. ABoxes and Büchi automata.

- In Chapter 8, we summarize the results of this thesis and discuss possible future work.

Most of the results of this thesis have already been published. The computation of logical updates and the analysis of their sizes were published in [LLMW06c]. We plan to publish this, together with the results about the other three kinds of updates, as a journal paper. The optimizations on the computation of logical updates are published in [DLB+09b, DLB+09a]. The results in Chapter 7 have not been published yet. Some results about the projection problem in DLs mentioned as "related work" in Section 1.2 have been published in [LLMW06a, LLMW06b, LLM08] and are a part of Maja Miličić's Ph.D thesis [Mil08].

Chapter 2

Preliminaries

We introduce the basic notions of DLs that are related to this thesis in Section 2.1. Boolean ABoxes and their expressivity compared to the @ constructor are investigated in Section 2.2. Section 2.3 is dedicated to illustrating the updates considered in this thesis and showing some properties of those updates.

2.1 Description Logics

In DLs, concepts are inductively defined with the help of a set of constructors over three disjoint sets of names: the set $\mathsf{N_C}$ of concept names, the set $\mathsf{N_R}$ of role names, and the set $\mathsf{N_I}$ of individual names. The set of constructors determines the expressivity of a specific DL. The DLs used in this thesis are between $\mathcal{ALC}$ and $\mathcal{ALCQIO}^{@}$. In what follows, we introduce the syntax of the DL $\mathcal{ALCQIO}^{@}$.

Definition 1 ($\mathcal{ALCQIO}^{@}$ Syntax)**.** A *role* is of the form r or r^- (the *inverse role* of r) for a role name $r \in \mathsf{N_R}$. The *set of* $\mathcal{ALCQIO}^{@}$*-concepts* over $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$ is inductively defined as follows:

- Every concept name $A \in \mathsf{N_C}$ is a concept.
- If C and D are concepts, r is a role, a is an individual name in $\mathsf{N_I}$, and n is a natural number, the following are also concepts:

$\top$	*top*
$\bot$	*bottom*
$\neg C$	*negation*
$C \sqcap D$	*conjunction*
$C \sqcup D$	*disjunction*
$\exists r.C$	*existential restriction*
$\forall r.C$	*value restriction*
$\{a\}$	*nominal*
$(\geqslant n\ r\ C)$	*at-least qualified number restriction*
$(\leqslant n\ r\ C)$	*at-most qualified number restriction*
$@_a C$	*at*

A concept is *atomic* if it is a concept name. A concept is *complex* otherwise. △

As usual, we use $C \rightarrow D$ as an abbreviation for $\neg C \sqcup D$, and $C \leftrightarrow D$ for $(C \rightarrow D) \sqcap (D \rightarrow C)$. Throughout this thesis, we adopt the following notational conventions in definitions and examples:

- In definitions and abstract examples, we use A, B for atomic concepts, C, D for complex concepts, r, s for roles, a, b for individual names, and m, n for natural numbers. In all cases, subscripts may be used.
- In concrete examples, concept names start with an uppercase letter followed by lowercase letters (e.g., Happy, Clever), role names start with a lowercase letter (e.g., hasChild), and individual names are composed of uppercase letters (e.g., MARY, JOHN).

The smallest propositionally closed DL is called $\mathcal{ALC}$, which allows only for top, bottom, negation, conjunction, disjunction, existential restriction and value restriction. Allowing for additional constructors forms extensions of $\mathcal{ALC}$. Letters and symbols in the name of a DL indicate the availability of constructors in it. For instance, the name $\mathcal{ALCQIO}^{@}$ stands for the DL obtained by extending $\mathcal{ALC}$ with $\mathcal{Q}$ualified number restriction, $\mathcal{I}$nverse role, n$\mathcal{O}$minal and @. Among the constructors of $\mathcal{ALCQIO}^{@}$, the inverse constructor is the only role constructor whereas the others are concept constructors.

A DL $\mathcal{L}_1$ is a *sublanguage* of a DL $\mathcal{L}_2$ if every constructor available in $\mathcal{L}_1$ is also available in $\mathcal{L}_2$. In this case, we can also say that $\mathcal{L}_2$ is a *superlanguage* of $\mathcal{L}_1$. If a DL $\mathcal{L}_1$ is a sublanguage of a DL $\mathcal{L}$ and $\mathcal{L}$ is a sublanguage of a DL $\mathcal{L}_2$, we say that $\mathcal{L}$ is *between* $\mathcal{L}_1$ and $\mathcal{L}_2$.

Concepts and roles are interpreted w.r.t. a model-theoretic *semantics* [Tar56], where concepts are taken to refer to sets of objects in the domain of interest and roles to relationships between them. Each individual name denotes an object in the domain. Throughout this thesis, we use $\sharp S$ to denote the cardinality of a set S.

Definition 2 ($\mathcal{ALCQIO}^{@}$ Semantics)**.** An interpretation $\mathcal{I}$ is a pair $(\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where the *domain* $\Delta^{\mathcal{I}}$ is a non-empty set and the *interpretation function* $\cdot^{\mathcal{I}}$ is a function which assigns

- a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ to every concept name $A \in \mathsf{N_C}$;
- a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ to every role name $r \in \mathsf{N_R}$;
- an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ to every individual name $a \in \mathsf{N_I}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ for all $a, b \in \mathsf{N_I}$ with $a \neq b$.

Let x be an element of $\Delta^{\mathcal{I}}$. Then, x is a *named object of* $\mathcal{I}$ if there exists some $a \in \mathsf{N_I}$ such that $x = a^{\mathcal{I}}$; x is an *anonymous object of* $\mathcal{I}$ otherwise.

The interpretation function is extended to inverse roles by the following definition:

$$(r^{-})^{\mathcal{I}} = \{(y, x) \mid (x, y) \in r^{\mathcal{I}}\}.$$

The interpretation function is extended to complex concepts as follows:

$$\begin{array}{rcl}
\top^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} & = & \emptyset \\
(\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists r.C)^{\mathcal{I}} & = & \{x \mid \exists y : ((x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\} \\
(\forall r.C)^{\mathcal{I}} & = & \{x \mid \forall y : ((x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}})\} \\
\{a\}^{\mathcal{I}} & = & \{a^{\mathcal{I}}\} \\
(\geqslant n\ r\ C)^{\mathcal{I}} & = & \{x \mid \#\{y \in C^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \geq n\} \\
(\leqslant n\ r\ C)^{\mathcal{I}} & = & \{x \mid \#\{y \in C^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \leq n\} \\
(@_a C)^{\mathcal{I}} & = & \begin{cases} \Delta^{\mathcal{I}} & \text{if } a^{\mathcal{I}} \in C^{\mathcal{I}} \\ \emptyset & \text{otherwise} \end{cases}
\end{array}$$

An interpretation $\mathcal{I}$ is called a *model* of a concept C if $C^{\mathcal{I}} \neq \emptyset$. △

Note that we adopt the *unique name assumption (UNA)*, i.e., different individual names are interpreted as different elements of the domain for all interpretations. In general, the UNA is not required in DLs, nevertheless it is natural in reasoning about action. We will illustrate this with the definition of semantics of updates (cf. Definition 17). Moreover, there are some proofs in this thesis which require the UNA, e.g., the proof of Lemma 56.

From Definition 2, we can see that $(\exists r.C)^{\mathcal{I}} = (\geqslant 1\ r\ C)^{\mathcal{I}}$ and $(\forall r.C)^{\mathcal{I}} = (\leqslant 0\ n\ \neg C)^{\mathcal{I}}$ for every interpretation $\mathcal{I}$. For this reason, in the DLs that contain qualified number restrictions, we will not treat existential restrictions and universal restrictions explicitly.

In order to refer to the computational complexity of the inference problems later on, we need to measure the size of inputs. Intuitively, the size of a concept or a role is defined as the number of necessary symbols to write it down. The *size of a role* r, denoted by $|r|$, is 1 if r is a role name and is 2 if it is an inverse role.

Definition 3 (Concept Size)**.** Let E be a concept over $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$. The *size of* E, denoted by $|E|$, is defined inductively as follows:

- $|E| = 1$ if $E = \top$, $E = \bot$, $E = A$ with $A \in \mathsf{N_C}$, or $E = \{a\}$ with $a \in \mathsf{N_I}$;
- $|E| = |C| + |D| + 1$ if $E = C \sqcap D$ or $E = C \sqcup D$; $|E| = |C| + 1$ if $E = \neg C$; $|E| = |C| + |r| + 1$ if $E = \exists r.C$ or $E = \forall r.C$; $|E| = |C| + 2$ if $E = @_a C$;
- the size of E depends on coding of n if $E = (\geqslant n\ r\ C)$ or $E = (\leqslant n\ r\ C)$.

$$|E| = \begin{cases} \lceil \log_2(\max\{2,n\}) \rceil + |C| + |r| + 1 & \text{if binary coding is used for } n \\ n + |C| + |r| + 1 & \text{if unary coding is used for } n \end{cases}$$

△

For later reference, we introduce the notion of *subconcepts*. Intuitively, a subconcept of a concept E is a concept occurring in E. The set of the subconcepts of E is the set containing all of subconcepts of E.

Definition 4 (Subconcept)**.** Let E be a concept over $\mathsf{N_C}$, $\mathsf{N_R}$, and $\mathsf{N_I}$. The *set of subconcepts* $\mathsf{Sub}(E)$ *of* E is inductively defined as follows:

- $\mathsf{Sub}(E) = \{E\}$ if $E = \top$, $E = \bot$, $E = A$ with $A \in \mathsf{N_C}$, or $E = \{a\}$ with $a \in \mathsf{N_I}$;
- $\mathsf{Sub}(E) = \{E\} \cup \mathsf{Sub}(C) \cup \mathsf{Sub}(D)$ if $E = C \sqcap D$ or $E = C \sqcup D$;
- $\mathsf{Sub}(E) = \{E\} \cup \mathsf{Sub}(C)$ if $E = \neg C$, $E = \exists r.C$, $E = \forall r.C$, $E = @_a C$, $E = (\geqslant n\ r\ C)$, or $E = (\leqslant n\ r\ C)$.

△

In DLs, an ABox is used to describe a specific state of affairs of an application domain in terms of concepts and roles. Individual names can be asserted to have properties. Such assertions are represented in an ABox.

Definition 5 (ABox)**.** A *concept assertion* is of the form $C(a)$ with a concept C and an individual name a. A *role assertion* is of the form $r(a, b)$ or $\neg r(a, b)$ with a role r and individual names a, b. An *ABox assertion* (or just *assertion*) is a concept assertion or a role assertion. An *ABox* is a finite set of assertions.

An ABox $\mathcal{A}$ is *simple* iff every assertion in $\mathcal{A}$ is a *literal*, i.e., it has one of the following forms:

$$A(a), \neg A(a), r(a, b), \text{ or } \neg r(a, b),$$

where A is a concept name, r is a role name, and a, b are individual names.

An interpretation $\mathcal{I}$ *satisfies* a concept assertion $C(a)$ iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$, a role assertion $r(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$, and a role assertion $\neg r(a, b)$ iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}$. We denote satisfaction of an assertion φ by an interpretation $\mathcal{I}$ with $\mathcal{I} \models \varphi$. An interpretation $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$ (written $\mathcal{I} \models \mathcal{A}$) if $\mathcal{I} \models \varphi$ for all assertions $\varphi \in \mathcal{A}$. Two ABoxes $\mathcal{A}$ and $\mathcal{A}'$ are *logically equivalent* (written $\mathcal{A} \equiv \mathcal{A}'$) iff they have the same set of models. △

In order to increase readability, we sometimes write a concept assertion as $a : C$ instead of $C(a)$.

In contrast to an interpretation, an ABox $\mathcal{A}$ is an incomplete description of the world status, i.e., there can exist some assertion φ such that neither φ nor its negation follows from $\mathcal{A}$. This is a consequence of the adopted *open world assumption* for ABoxes in DLs [BCM$^+$03].

Based on the notion of the size of a concept and a role, we introduce the size of an assertion and the size of an ABox in the next definition.

Definition 6 (ABox Size)**.** The *size of a concept assertion* $C(a)$ is $|C| + 1$. The *size of a role assertion* $r(a, b)$ is $|r| + 2$. The size of an assertion φ is denoted with $|\varphi|$. The *size of an ABox* $\mathcal{A}$, denoted with $|\mathcal{A}|$, is $\sum_{\varphi \in \mathcal{A}} |\varphi|$. △

We illustrate the notions introduced so far by the following example:

Example 7. *Consider the following ABox:*

$$\mathcal{A} = \{\mathsf{JOHN{:}}\exists \mathsf{hasChild.Happy}, \mathsf{MARY{:}Happy} \sqcap \mathsf{Clever}\}.$$

Apparently, all constructors used in $\mathcal{A}$ are available in $\mathcal{ALC}$. The first assertion means that John has a happy child and the second assertion tells us that Mary is happy and clever. Since there are complex concepts occurring in $\mathcal{A}$, $\mathcal{A}$ is not simple. The size of $\mathsf{JOHN{:}\exists hasChild.Happy}$ *is 4.* $\mathsf{Sub(Happy \sqcap Clever) = \{Happy \sqcap Clever, Happy, Clever\}}$.

TBoxes are used in DLs to make statements about how concepts (and roles) are related to each other. Those statements are background knowledge about the world and can play the role of domain constraints in action theory [BLM$^+$05] based on DLs. We only consider the simplest kind of TBoxes, *acyclic TBoxes*, in this thesis. One can introduce atomic concepts as abbreviations for complex concepts by an acyclic TBox.

Definition 8 (Acyclic TBox)**.** A *concept definition* is of the form $A \equiv C$, where A is a concept name and C is a concept. A concept name A *directly uses* a concept name B in $\mathcal{T}$ if $A \equiv C \in \mathcal{T}$ and B occurs in C. The transitive closure of directly uses is *uses*. An *acyclic TBox* $\mathcal{T}$ is a finite set of concept definitions such that

- there is not a concept name occurring more than once on the left-hand side of concept definitions, and
- there is not a concept name that uses itself.

A concept name A in $\mathcal{T}$ is a *defined concept name in* $\mathcal{T}$ if there exists a concept definition $A \equiv C$ in $\mathcal{T}$. Otherwise, it is a *primitive concept name in* $\mathcal{T}$. We use $\mathsf{def}(\mathcal{T})$ and $\mathsf{pri}(\mathcal{T})$ to denote the set of defined concept names and the set of primitive concept names in $\mathcal{T}$, respectively.

An interpretation $\mathcal{I}$ *satisfies* a concept definition $A \equiv C$ (written $\mathcal{I} \models A \equiv C$) iff $A^{\mathcal{I}} = C^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* of an acyclic TBox $\mathcal{T}$ (written $\mathcal{I} \models \mathcal{T}$) if $\mathcal{I} \models A \equiv C$ for all concept definitions $A \equiv C \in \mathcal{T}$. △

In the following definition, we introduce the size of a concept definition and the size of a TBox.

Definition 9 (TBox Size)**.** The *size of a concept definition* $A \equiv C$ is $|C| + 2$. The *size of an acyclic TBox* $\mathcal{T}$, denoted with $|\mathcal{T}|$, is $\sum_{A \equiv C \in \mathcal{T}}(|C| + 2)$. △

Although updating ABoxes is the main focus of this thesis, acyclic TBoxes are also useful in this thesis. We will see in Chapter 7 how to employ acyclic TBoxes to verify DL-LTL formulas w.r.t. ABoxes and infinite sequences of updates.

Throughout this thesis, we use $M(\mathcal{A})$, $M(\mathcal{T})$, and $M(\varphi)$ to denote, respectively, the set of all models of the ABox $\mathcal{A}$, the TBox $\mathcal{T}$, and the assertion φ. For a set Γ of interpretations and an assertion φ, we write $\Gamma \models \varphi$ iff $\mathcal{I} \models \varphi$ for all $\mathcal{I} \in \Gamma$.

Reasoning in DLs makes implicit knowledge explicit. A whole range of reasoning problems has already been investigated in the past twenty years of DL research. In the next definition, we introduce the ones related to this thesis.

Definition 10 (Reasoning Problems)**.** Let $\mathcal{A}$ be an ABox, $\mathcal{T}$ an acyclic TBox and φ an assertion. Then

- $\mathcal{A}$ is *consistent w.r.t.* $\mathcal{T}$ iff $M(\mathcal{A}) \cap M(\mathcal{T}) \neq \emptyset$.

- φ is a *(logical) consequence* of $\mathcal{A}$ w.r.t. $\mathcal{T}$ (written $\mathcal{A} \models_{\mathcal{T}} \varphi$) iff $(M(\mathcal{A}) \cap M(\mathcal{T})) \subseteq M(\varphi)$.

△

In those reasoning problems the phrase "w.r.t. $\mathcal{T}$" will simply be dropped instead of writing "w.r.t. $\emptyset$" if the TBox $\mathcal{T}$ under consideration is empty. Existing DL reasoners implement different kinds of algorithms solving the aforementioned and other reasoning problems. Although negated role assertions and the UNA are slightly unusual and may not be directly supported by DL reasoners, it has been shown in [Mil08] that the consistency problem and the consequence problem with them do not increase the computational complexity for the DLs.

2.2 The @ constructor and Boolean ABoxes

The @ constructor from hybrid logic [AdR01] is slightly non-standard in DLs and it is not supported directly by DL reasoners. However, even approximate updates are not expressible without it (see details in Section 5.2). In this sense, the @ constructor can be regarded as the missing *logical connective* required in order to obtain (logical and approximate) updates. Alternatively, we can use *Boolean ABoxes* [ABHM03], which are closely related to the @ constructor but strictly more expressive.

Definition 11 (Boolean ABox)**.** A *Boolean ABox assertion* (or just *Boolean assertion*) is inductively defined as follows:

- Every ABox assertion is a Boolean ABox assertion;
- If φ and ψ are Boolean ABox assertions, then so are $\varphi \wedge \psi$ and $\varphi \vee \psi$.

A *Boolean ABox* is a finite set of Boolean ABox assertions.

Satisfaction of an ABox assertion by an interpretation $\mathcal{I}$ can be extended to a Boolean ABox assertion in a straightforward way:

- $\mathcal{I} \models \varphi \wedge \psi$ if $\mathcal{I} \models \varphi$ and $\mathcal{I} \models \psi$;
- $\mathcal{I} \models \varphi \vee \psi$ if $\mathcal{I} \models \varphi$ or $\mathcal{I} \models \psi$.

An interpretation $\mathcal{I}$ is a model of a Boolean ABox $\mathcal{A}$ (written $\mathcal{I} \models \mathcal{A}$) if $\mathcal{I} \models \varphi$ for every Boolean assertion $\varphi \in \mathcal{A}$. △

Note that we do not need to explicitly introduce negation since we admit negated role assertions and concept negation is contained in every DL considered in this thesis. If a Boolean ABox $\mathcal{A}$ contains only one Boolean assertion φ, then we sometimes write $\mathcal{A}$ as φ instead of $\{\varphi\}$.

We also use $M(\mathcal{A})$ to denote the set of all models of a Boolean ABox $\mathcal{A}$. In the same pattern as the ones of ABoxes, consistency of a Boolean ABox, equivalence of two Boolean ABoxes, and logical consequence are defined.

The notion of the size of an ABox is extended to a Boolean ABox in the next definition.

Definition 12 (Boolean ABox Size)**.** Let ϕ be a Boolean assertion of the form $\varphi \wedge \psi$ or $\varphi \vee \psi$. The *size of a Boolean assertion* ϕ, denoted with $|\phi|$, is $|\varphi| + |\psi| + 1$. The *size of a Boolean ABox* $\mathcal{A}$, denoted with $|\mathcal{A}|$, is $\sum_{\varphi \in \mathcal{A}} |\varphi|$. △

For a given DL $\mathcal{L}$, a concept description C is an *$\mathcal{L}$-concept* iff all constructors of building C are available in $\mathcal{L}$. Similarly, we can define *$\mathcal{L}$-assertions*, *$\mathcal{L}$-ABoxes*, *$\mathcal{L}$-TBoxes*, and *Boolean $\mathcal{L}$-ABoxes*. The following lemma relates Boolean ABoxes and the @ constructor. It shows that non-Boolean $\mathcal{L}^{@}$-ABoxes have exactly the same expressive power as Boolean $\mathcal{L}$-ABoxes provided that $\mathcal{L}$ is a DL between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}$.

Lemma 13.
(i) Let $\mathcal{L}$ be a DL between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$. Then for every Boolean $\mathcal{L}^{@}$-ABox, there exists an equivalent Boolean $\mathcal{L}$-ABox;

(ii) Let $\mathcal{L}$ be a DL between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}$. Then for every Boolean $\mathcal{L}$-ABox, there exists an equivalent non-Boolean $\mathcal{L}^{@}$-ABox.

Proof. Concerning (i), let $\mathcal{A}$ be a Boolean $\mathcal{L}^{@}$-ABox, and let φ be an assertion from $\mathcal{A}$ such that $@_b D$ is a subconcept of some concept occurring in φ.

Claim 1. For all interpretations $\mathcal{I}$, $\mathcal{I} \models \varphi$ iff $\mathcal{I} \models (D(b) \wedge \varphi[\top/@_b D]) \vee (\neg D(b) \wedge \varphi[\bot/@_b D])$, where $\varphi[X/@_b D]$ denotes the concept obtained from φ by replacing all occurrences of $@_b D$ with X.
Proof of Claim 1: $\mathcal{I} \models \varphi$ iff $b^{\mathcal{I}} \in D^{\mathcal{I}}$ and $\mathcal{I} \models \varphi$, or $b^{\mathcal{I}} \notin D^{\mathcal{I}}$ and $\mathcal{I} \models \varphi$ iff $(@_b D)^{\mathcal{I}} = \top^{\mathcal{I}}$ and $\mathcal{I} \models \varphi$, or $(@_b D)^{\mathcal{I}} = \bot^{\mathcal{I}}$ and $\mathcal{I} \models \varphi$ iff $\mathcal{I} \models \neg D(b) \wedge \varphi[\bot/@_b D]$ or $\mathcal{I} \models \neg D(b) \wedge \varphi[\bot/@_b D]$ iff $\mathcal{I} \models (D(b) \wedge \varphi[\top/@_b D]) \vee (\neg D(b) \wedge \varphi[\bot/@_b D])$. This finishes the proof of Claim 1.

Let $\mathcal{A}'$ be the ABox obtained from $\mathcal{A}$ by replacing φ with $(D(b) \wedge \varphi[\top/@_b D]) \vee (\neg D(b) \wedge \varphi[\bot/@_b D])$. By Claim 1, $\mathcal{A}'$ is equivalent to $\mathcal{A}$. Iterating this replacement results in an equivalent Boolean $\mathcal{L}$-ABox.

Concerning (ii), define a mapping $\cdot^*$ from ABox assertions in $\mathcal{L}$ to $\mathcal{L}^{@}$-concepts as follows:

$$\begin{aligned} C(a)^* &= @_a C \\ r(a,b)^* &= @_a \exists r.\{b\} \\ \neg r(a,b)^* &= @_a \forall r.\neg\{b\} \end{aligned}$$

Every Boolean ABox assertion φ can be converted into an $\mathcal{L}^{@}$-concept φ^* by replacing $\wedge$ with $\sqcap$, $\vee$ with $\sqcup$, and every (possibly Boolean) assertion ψ with ψ^*.

Claim 2. For all interpretations $\mathcal{I}$ and for all individual names a, $\mathcal{I} \models \varphi$ iff $\mathcal{I} \models \varphi^*(a)$.
Proof of Claim 2: This can be shown by induction on the Boolean structure of φ.

- If φ is a non-Boolean assertion, then Claim 2 follows directly from the definition of the mapping $\cdot^*$.
- $\varphi = \psi_1 \vee \psi_2$. Thus, $\mathcal{I} \models \varphi$ iff $\mathcal{I} \models \psi_1$ or $\mathcal{I} \models \psi_2$ iff (by I.H.) $\mathcal{I} \models \psi_1^*(a)$ or $\mathcal{I} \models \psi_2^*(a)$ iff $\mathcal{I} \models \psi_1^* \sqcup \psi_2^*(a)$.
- The case of $\varphi = \psi_1 \wedge \psi_2$ can be shown similarly to the above one.

This finishes the proof of Claim 2.

Let $\mathcal{A} = \{\varphi_1, \ldots, \varphi_n\}$ be a Boolean $\mathcal{L}$-ABox. Define a non-Boolean $\mathcal{L}^{@}$-ABox $\mathcal{A}' := \{(\varphi_1^* \sqcap \cdots \sqcap \varphi_n^*)(a)\}$, where a is an arbitrary individual name. As a direct consequence of Claim 2, $\mathcal{A}'$ is logically equivalent to $\mathcal{A}$. ❑

Note that (ii) does not hold for $\mathcal{ALC}$. More specifically, there are Boolean $\mathcal{ALC}$-ABoxes for which no equivalent non-Boolean $\mathcal{ALC}^{@}$-ABox exists. We will show this in Corollary 57 in Chapter 5.

From the proof of Lemma 13, we can also see that the translation of Boolean $\mathcal{L}$-ABoxes into non-Boolean $\mathcal{L}^{@}$-ABoxes is polynomial, while the reverse translation induces an exponential blowup. More precisely, this blowup is exponential in the nesting depth of the @ constructor.

The DLs introduced so far are all used to represent static information of the world. In the next section, we introduce the central notion of this thesis, *updates*, which employ ABox assertions to express dynamic changes of the world.

2.3 Updates

In many applications of DLs, an ABox is used to represent the current state of affairs in the application domain [BCM+03]. In such applications, it is necessary to update the ABox in the case that the world has changed. A collection of new information describing changes which take place in the world is referred to as an update in this thesis.

Definition 14 (Update)**.** A *conditional update* (or just *update* for short) $\mathcal{U}$ is a finite set of expressions of the form φ/ψ, where φ is an ABox assertion (possibly involving non-atomic concepts) and ψ is a literal, i.e., it has one of the following forms:

$$A(a), \neg A(a), r(a,b), \text{ or } \neg r(a,b)$$

where A is a concept name, r is a role name, and a, b are individual names. Let φ/ψ be an expression in an update $\mathcal{U}$. Then, ψ is called an *effect* of $\mathcal{U}$ and φ is called the *precondition* of ψ in $\mathcal{U}$.

An update $\mathcal{U}$ is *unconditional* iff for all $\varphi/\psi \in \mathcal{U}$, $\varphi = \top(a)$ for some individual name a. In this case, we write the element of $\mathcal{U}$ as ψ instead of $\top(a)/\psi$. Thus, an unconditional update is a simple ABox. △

In order to avoid contradictory information resulting from updates, a consistency condition on updates is required: if φ/ψ and $\varphi'/\neg\psi$ are both in $\mathcal{U}$, then the ABox $\{\varphi, \varphi'\}$ has to be inconsistent. In this thesis, we consider only consistent updates. In order to analyze computational properties of updating ABoxes, we need to define the size of updates.

Definition 15 (Update Size)**.** The *size of an update* $\mathcal{U}$ is denoted with $|\mathcal{U}|$ and defined by $|\mathcal{U}| = \sum_{\varphi/\psi \in \mathcal{U}} (|\varphi| + |\psi|)$. △

Example 16. *Consider the following update:*

$$\begin{aligned}\mathcal{U} \;=\; \{&\mathsf{PETER{:}}\exists\mathsf{hasCredit.}\{\mathsf{TRS}\}/\mathsf{PETER{:}Happy},\\ &\mathsf{MARY{:}}\neg\exists\mathsf{hasTopic.Seminar}/\mathsf{MARY{:}}\neg\mathsf{Happy}\}.\end{aligned}$$

As a result of achieving the credits of the term rewriting systems lecture, Peter is happy, while Mary is unhappy because she cannot get a topic in a seminar. The size of $\mathcal{U}$ *is 14.*

To define the semantics of updates, we must define how the application of an update changes the world, i.e., how it transforms a given interpretation $\mathcal{I}$ into a new one $\mathcal{I}'$. We adopt the possible models approach (PMA) initially proposed in [Win88] and further elaborated e.g., in [Win90, BH93, Her96, DLMB98].[1] The PMA was first used in action formalisms based on DLs in [BLM+05]. Intuitively, an expression φ/ψ means that if φ holds in the current state of the world, then the effect ψ holds in the state after applying the update. The idea underlying the PMA is that the interpretation should change as little as possible while still satisfying all the effects whose preconditions are satisfied. We apply this minimization of change to updating ABoxes in a way such that the interpretations of all concept and role names remain as before unless they are affected by the update. Formally, the semantics of updates is given by updating interpretations.

Definition 17 (Interpretation Update)**.** Let $\mathcal{U}$ be an update and $\mathcal{I}, \mathcal{I}'$ interpretations such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}'}$ and $\mathcal{I}$ and $\mathcal{I}'$ agree on the interpretation of individual names. Then $\mathcal{I}'$ is the *result of updating* $\mathcal{I}$ *with* $\mathcal{U}$, written $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$, if the following conditions hold:

- for all concept names A,

$$A^{\mathcal{I}'} = \; (A^{\mathcal{I}} \cup \{a^{\mathcal{I}} \mid \varphi/A(a) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}) \setminus \{a^{\mathcal{I}} \mid \varphi/\neg A(a) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}, \text{ and}$$

- for all role names r,

$$\begin{aligned} r^{\mathcal{I}'} \;=\; &(r^{\mathcal{I}} \cup \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/r(a,b) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}) \\ &\setminus \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \varphi/\neg r(a,b) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}. \end{aligned}$$

Let $\mathcal{I}$ be an interpretation and $\mathcal{U}$ an update. Then, an effect ψ of $\mathcal{U}$ is *triggered* in $\mathcal{I}$ if there exists some φ such that $\varphi/\psi \in \mathcal{U}$ and $\mathcal{I}$ is a model of φ. Let $\mathcal{U}_1 \cdots \mathcal{U}_n$ be a finite sequence of updates and $\mathcal{I}, \mathcal{I}'$ two interpretations. Then, $\mathcal{I}'$ is the *result of updating* $\mathcal{I}$ *with* $\mathcal{U}_1 \cdots \mathcal{U}_n$, written $\mathcal{I} \Longrightarrow_{\mathcal{U}_1 \cdots \mathcal{U}_n} \mathcal{I}'$, if there are $\mathcal{I}_0, \ldots, \mathcal{I}_n$ such that $\mathcal{I}_0 = \mathcal{I}$, $\mathcal{I}_n = \mathcal{I}'$, and $\mathcal{I}_i \Longrightarrow_{\mathcal{U}_{i+1}} \mathcal{I}_{i+1}$ for all i with $0 \leq i < n$. △

It is easily seen that, for each consistent update $\mathcal{U}$, the relation $\Longrightarrow_{\mathcal{U}}$ is functional. Therefore, we can write $\mathcal{I}^{\mathcal{U}}$ to denote the unique $\mathcal{I}'$ with $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$.

[1] As discussed in [BH93, Her96, DLMB98], the PMA is inadequate when disjunctive effects or integrity constraints are under consideration. In this thesis, neither of them is allowed and thus the PMA is uncontroversial.

The UNA plays an important role for giving reasonable semantics of updates. Consider the unconditional update $\{A(a), \neg A(b)\}$ and some interpretation $\mathcal{I}$ with $a^{\mathcal{I}} = b^{\mathcal{I}}$. It is clear that $\mathcal{U}$ is a consistent update. However, $\mathcal{I}^{\mathcal{U}} \not\models A(a)$ according to the above definition, which means that triggered effects are not satisfied after the update. Adopting the UNA avoids this unintuitive result.

As a direct consequence of Definition 17, updating an interpretation $\mathcal{I}$ can make changes only on the named objects of $\mathcal{I}$. More precisely, for all interpretations $\mathcal{I}$, for all $x, y \in \Delta^{\mathcal{I}}$, for all $A \in \mathsf{N_C}$, and for all $r \in \mathsf{N_R}$, if x is an anonymous object of $\mathcal{I}$, then we have the following:

- $x \in A^{\mathcal{I}}$ iff $x \in A^{\mathcal{I}^{\mathcal{U}}}$,
- $(x, y) \in r^{\mathcal{I}}$ iff $(x, y) \in r^{\mathcal{I}^{\mathcal{U}}}$, and
- $(y, x) \in r^{\mathcal{I}}$ iff $(y, x) \in r^{\mathcal{I}^{\mathcal{U}}}$.

The above observation is frequently used in this thesis.

An ABox $\mathcal{A}$ is an incomplete description of the world and an model of $\mathcal{A}$ gives a complete description of the world. The set of models of $\mathcal{A}$ is composed of all possible world states. Updating an ABox $\mathcal{A}$ can be performed by updating the set of models of $\mathcal{A}$.

Definition 18 (Updated Model)**.** Let $\mathcal{A}$ be an ABox. The *set of models of the update of $\mathcal{A}$ with $\mathcal{U}$* is denoted by $M(\mathcal{A} * \mathcal{U})$ and defined by

$$M(\mathcal{A} * \mathcal{U}) = \{\mathcal{I}^{\mathcal{U}} \mid \mathcal{I} \in M(\mathcal{A})\}.$$

△

The main concern in the ABox update problem is to find out in which DLs $M(\mathcal{A}*\mathcal{U})$ can always be "described" and, if so, what is the minimal size of such a description. The rather vague term "described" is deliberately used here and in what follows, four different formalizations will be investigated. We start with two of them.

Definition 19 (Logical Update and Approximate Update)**.** Let $\mathcal{L}$ be a DL. Let $\mathcal{A}$ be an $\mathcal{L}$-ABox and $\mathcal{U}$ an update. An $\mathcal{L}$-ABox $\mathcal{A}'$ is

- a *logical update of $\mathcal{A}$ with $\mathcal{U}$*, in symbols $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$, if
$$M(\mathcal{A} * \mathcal{U}) = M(\mathcal{A}').$$
- an *approximate update of $\mathcal{A}$ with $\mathcal{U}$ w.r.t. $\mathcal{L}$*, in symbols $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{L}} \mathcal{A}'$, if for all $\mathcal{L}$-assertions φ, we have
$$M(\mathcal{A} * \mathcal{U}) \models \varphi \text{ iff } M(\mathcal{A}') \models \varphi.$$

△

We often refer to the ABox $\mathcal{A}$ in the above definition as the *original* ABox and the ABox $\mathcal{A}'$ as the *updated* ABox if it is clear from the context which kind of updates it is.

Example 20. *Consider the $\mathcal{ALC}$-ABox $\mathcal{A}$ in Example 7*

$$\mathcal{A} = \{\mathsf{JOHN}{:}\exists\mathsf{hasChild}.\mathsf{Happy}, \mathsf{MARY}{:}\mathsf{Happy} \sqcap \mathsf{Clever}\}.$$

and the unconditional update $\mathcal{U} = \{\mathsf{MARY}{:}\neg\mathsf{Happy}\}$. A logical update of $\mathcal{A}$ with $\mathcal{U}$ is the following $\mathcal{ALCO}$-ABox $\mathcal{A}'$:

$$\mathcal{A}' = \{\mathsf{JOHN}{:}\exists\mathsf{hasChild}.(\mathsf{Happy} \sqcup \{\mathsf{MARY}\}), \mathsf{MARY}{:}\neg\mathsf{Happy} \sqcap \mathsf{Clever}\}.$$

Since the construction of logical updates will be addressed in Section 3.1, instead of formally showing here that $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$, we only give some intuition.

Because ABoxes adopt the open world assumption and thus present the domain in an incomplete way, we have no information about whether or not Mary is a child of John. However, because we cannot exclude that this is the case, after the update John may have an unhappy child that is Mary. Mary is still clever even if she is no longer happy.

As a direct consequence of Definition 17, a logical update of an ABox $\mathcal{A}$ with an update $\mathcal{U}$ is also an approximate update of $\mathcal{A}$ with $\mathcal{U}$ w.r.t. any DL $\mathcal{L}$. This justifies the notation introduced for logical updates, i.e., $\mathcal{L}$ does not appear in $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$. Moreover, logical updates neither depend on the syntactic form of the given ABox nor on the underlying DL. This is formalized as follows:

Lemma 21. *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two DLs, $\mathcal{A}_1$ an $\mathcal{L}_1$-ABox logically equivalent to the $\mathcal{L}_2$-ABox $\mathcal{A}_2$, and $\mathcal{U}$ an update. If $\mathcal{A}_i'$ is a logical update of $\mathcal{A}_i$ with $\mathcal{U}$ formulated in $\mathcal{L}_i$ for $i = 1, 2$, then $\mathcal{A}_1'$ is logically equivalent to $\mathcal{A}_2'$.*

Proof. $M(\mathcal{A}_1') = M(\mathcal{A}_1 * \mathcal{U}) = \{\mathcal{I}^{\mathcal{U}} \mid \mathcal{I} \in M(\mathcal{A}_1)\} = \{\mathcal{I}^{\mathcal{U}} \mid \mathcal{I} \in M(\mathcal{A}_2)\} = M(\mathcal{A}_2 * \mathcal{U}) = M(\mathcal{A}_2')$. ❏

Based on the above lemma, in what follows we can talk about *the* logical update of $\mathcal{A}$ with $\mathcal{U}$ without referring to any underlying DL.

The situation is different for approximate updates. Approximate updates describe updated ABoxes in the sense that they capture the logical consequences of the update *for a fixed DL $\mathcal{L}$*. However, an approximate update w.r.t. some DL $\mathcal{L}$ is not necessarily an approximate update w.r.t. to its sublanguage $\mathcal{L}'$. To prove this, we introduce the notion of "logical equivalence w.r.t. an underlying DL $\mathcal{L}$" (called $\mathcal{L}$-indistinguishability):

Definition 22 ($\mathcal{L}$-indistinguishable)**.** Let $\mathcal{L}$ be a DL. Two ABoxes $\mathcal{A}$ and $\mathcal{A}'$ are *$\mathcal{L}$-indistinguishable*, written $\mathcal{A} \equiv_{\mathcal{L}} \mathcal{A}'$, if for all $\mathcal{L}$-assertions φ, we have $\mathcal{A} \models \varphi$ iff $\mathcal{A}' \models \varphi$. △

With the help of this notion we can formulate the following properties of approximate updates:

Lemma 23. *Let $\mathcal{A}$ be an ABox, $\mathcal{U}$ an update, and $\mathcal{L}$ a DL.*

1. *If $\mathcal{A}'$ is a logical update of $\mathcal{A}$ with $\mathcal{U}$, then $\mathcal{A}''$ is an approximate update of $\mathcal{A}$ with $\mathcal{U}$ w.r.t. $\mathcal{L}$ iff $\mathcal{A}' \equiv_{\mathcal{L}} \mathcal{A}''$.*

2. *If $\mathcal{A}_1$ and $\mathcal{A}_2$ are logically equivalent $\mathcal{L}$-ABoxes and $\mathcal{A}_i'$ is an approximate update of $\mathcal{A}_i$ with $\mathcal{U}$ w.r.t. $\mathcal{L}$, for $i = 1, 2$, then $\mathcal{A}_1'$ is logically equivalent to $\mathcal{A}_2'$.*

3. *If $\mathcal{A}'$ is a logical update of $\mathcal{A}$ with $\mathcal{U}$ and $\mathcal{A}''$ is an approximate update of $\mathcal{A}$ with $\mathcal{U}$ w.r.t. $\mathcal{L}$, then $M(\mathcal{A}') \subseteq M(\mathcal{A}'')$.*

Proof.

1. "⇒": Suppose that $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$ and $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{L}} \mathcal{A}''$. Thus, for all $\mathcal{L}$-assertions φ, $\mathcal{A}' \models \varphi$ iff $M(\mathcal{A}') \models \varphi$ iff (since $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$ implies $M(\mathcal{A}') = M(\mathcal{A} * \mathcal{U})$) $M(\mathcal{A} * \mathcal{U}) \models \varphi$ iff (since $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{L}} \mathcal{A}''$) $M(\mathcal{A}'') \models \varphi$ iff $\mathcal{A}'' \models \varphi$. Hence, $\mathcal{A}' \equiv_{\mathcal{L}} \mathcal{A}''$.
   "⇐": Suppose that $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$ and $\mathcal{A}' \equiv_{\mathcal{L}} \mathcal{A}''$. Thus, for all $\mathcal{L}$-assertions φ, $M(\mathcal{A} * \mathcal{U}) \models \varphi$ iff (since $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$ implies $M(\mathcal{A}') = M(\mathcal{A} * \mathcal{U})$) $M(\mathcal{A}') \models \varphi$ iff $\mathcal{A}' \models \varphi$ iff (since $\mathcal{A}' \equiv_{\mathcal{L}} \mathcal{A}''$) $\mathcal{A}'' \models \varphi$ iff $M(\mathcal{A}'') \models \varphi$. Hence, $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{L}} \mathcal{A}''$.

2. Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are logically equivalent, $M(\mathcal{A}_1 * \mathcal{U}) = \{\mathcal{I}^{\mathcal{U}} \mid \mathcal{I} \in M(\mathcal{A}_1)\}$ and $M(\mathcal{A}_2 * \mathcal{U}) = \{\mathcal{I}^{\mathcal{U}} \mid \mathcal{I} \in M(\mathcal{A}_2)\}$, we have $M(\mathcal{A}_1 * \mathcal{U}) = M(\mathcal{A}_2 * \mathcal{U})$. Since $\mathcal{A}_i'$ is an approximate update of $\mathcal{A}_i$ with $\mathcal{U}$ w.r.t. $\mathcal{L}$ for $i = 1, 2$, we have $M(\mathcal{A}_1') \models \varphi$ iff $M(\mathcal{A}_1 * \mathcal{U}) \models \varphi$ and $M(\mathcal{A}_2') \models \varphi$ iff $M(\mathcal{A}_2 * \mathcal{U}) \models \varphi$ for every $\mathcal{L}$-assertion φ. Thus, $M(\mathcal{A}_1') \models \varphi$ iff $M(\mathcal{A}_2') \models \varphi$, i.e., $\mathcal{A}_1' \equiv_{\mathcal{L}} \mathcal{A}_2'$.

   Assume that $M(\mathcal{A}_1') \neq M(\mathcal{A}_2')$. Thus, there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{A}_1' \wedge \mathcal{I} \not\models \mathcal{A}_2'$ or $\mathcal{I} \models \mathcal{A}_2' \wedge \mathcal{I} \not\models \mathcal{A}_1'$. In the case that $\mathcal{I} \models \mathcal{A}_1' \wedge \mathcal{I} \not\models \mathcal{A}_2'$, we know that there exists an assertion $\varphi \in \mathcal{A}_2'$ such that $\mathcal{I} \not\models \varphi$. This, together with $\mathcal{I} \models \mathcal{A}_1'$, yields $\mathcal{A}_1' \not\models \varphi$. Since $\mathcal{A}_2'$ is an $\mathcal{L}$-ABox and $\varphi \in \mathcal{A}_2'$, we have that φ is an $\mathcal{L}$-assertion and $\mathcal{A}_2' \models \varphi$ which is a contradiction. The other case can be proved similarly.

3. Assume that $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$ and $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{L}} \mathcal{A}''$ such that $M(\mathcal{A}') \not\subseteq M(\mathcal{A}'')$. Thus, there is an interpretation $\mathcal{I}$ such that $\mathcal{I} \in M(\mathcal{A}')$ and $\mathcal{I} \notin M(\mathcal{A}'')$. $\mathcal{I} \notin M(\mathcal{A}'')$ implies that there exists an assertion $\varphi \in \mathcal{A}''$ such that $\mathcal{I} \not\models \varphi$. This, together with $\mathcal{I} \in M(\mathcal{A}')$, implies $\mathcal{A}' \not\models \varphi$. $\mathcal{A}'' \models \varphi$ and φ is an $\mathcal{L}$-assertion since $\varphi \in \mathcal{A}''$. Thus, $\mathcal{A}'$ and $\mathcal{A}''$ are not $\mathcal{L}$-indistinguishable. Point 1 of this lemma implies that $\mathcal{A}''$ is not an approximate update of $\mathcal{A}$ with $\mathcal{U}$ w.r.t. $\mathcal{L}$, which contradicts the assumption.

❑

Point 1 of the above lemma gives us another characterization of approximate updates. From Point 2, it is clear that approximate updates w.r.t. a fixed DL do not depend on the representation of the updated ABox. Moreover, approximate updates of equivalent $\mathcal{L}$-ABoxes with an update $\mathcal{U}$ w.r.t. the same DL $\mathcal{L}$ are not only $\mathcal{L}$-indistinguishable but also logically equivalent. Point 3 captures a property of models of approximate updates, which will be used in the proof of Theorem 59.

Returning to Example 20, we now show that there exists an ABox $\mathcal{A}''$ which is an approximate update of $\mathcal{A}$ with $\mathcal{U}$ w.r.t. $\mathcal{ALC}$.

Lemma 24. *Consider the ABox $\mathcal{A}$ in Example 7, the update $\mathcal{U} = \{\mathsf{MARY}{:}\neg\mathsf{Happy}\}$ and the following ABox $\mathcal{A}''$:*

$$\mathcal{A}'' = \{\mathsf{JOHN}{:}\exists\mathsf{hasChild}.(\mathsf{Happy} \sqcup \mathsf{Clever}), \mathsf{MARY}{:}\neg\mathsf{Happy} \sqcap \mathsf{Clever}\}.$$

Thus, we have $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{ALC}} \mathcal{A}''$.

Proof. By Lemma 23 Point 1, it is sufficient to show that $\mathcal{A}'$ in Example 20 and $\mathcal{A}''$ are $\mathcal{ALC}$-indistinguishable, i.e., for all $\mathcal{ALC}$-assertions φ, $\mathcal{A}' \models \varphi$ iff $\mathcal{A}'' \models \varphi$. Both directions are shown by contraposition.

"$\Leftarrow$": Assume that $\mathcal{A}' \not\models \varphi$ for some $\mathcal{ALC}$-assertion φ. Thus, there is a model $\mathcal{I}$ of $\mathcal{A}'$ such that $\mathcal{I} \not\models \varphi$. $\mathcal{I} \models \mathcal{A}'$ implies that $\mathcal{I} \models \mathsf{JOHN}{:}\exists\mathsf{hasChild}.(\mathsf{Happy} \sqcup \{\mathsf{MARY}\})$ and $\mathcal{I} \models \mathsf{MARY}{:}\neg\mathsf{Happy} \sqcap \mathsf{Clever}$. The latter implies $\mathsf{MARY}^{\mathcal{I}} \in \mathsf{Clever}^{\mathcal{I}}$, which, together with the former, yields $\mathcal{I} \models \mathsf{JOHN}{:}\exists\mathsf{hasChild}.(\mathsf{Happy} \sqcup \mathsf{Clever})$. Thus, $\mathcal{I} \models \mathcal{A}''$. Hence, $\mathcal{A}'' \not\models \varphi$ since $\mathcal{I} \models \mathcal{A}''$ and $\mathcal{I} \not\models \varphi$.

"$\Rightarrow$": Assume that $\mathcal{A}'' \not\models \varphi$ for some $\mathcal{ALC}$-assertion φ. If φ is a (possibly negated) role assertion, then $\mathcal{A}' \not\models \varphi$ since it is not hard to see that $\mathcal{A}' \not\models \varphi$ for all role assertions φ. Hence, φ is a concept assertion. Suppose $\varphi = C(a)$ and take a model $\mathcal{I}$ of $\mathcal{A}''$ with $a^{\mathcal{I}} \notin C^{\mathcal{I}}$. By standard unravelling [BdRV01], we can convert $\mathcal{I}$ into a model $\mathcal{J}$ of $\mathcal{A}''$ such that (still) $a^{\mathcal{J}} \notin C^{\mathcal{J}}$ and the directed graph $G = (\Delta^{\mathcal{J}}, \bigcup_{r \in \mathsf{N_R}} r^{\mathcal{J}})$ is such that $\mathsf{JOHN}^{\mathcal{J}}$ is not reachable from $\mathsf{MARY}^{\mathcal{J}}$.

- $\Delta^{\mathcal{J}} = \{d_0 \cdots d_k \mid k \geq 0 \wedge \forall i \in \{0, \ldots, k\} : d_i \in \Delta^{\mathcal{I}}\}$,
- for all $A \in \mathsf{N_C}$, $A^{\mathcal{J}} = \{d_0 \cdots d_k \mid d_0 \cdots d_k \in \Delta^{\mathcal{J}} \wedge d_k \in A^{\mathcal{I}}\}$,
- for all $r \in \mathsf{N_R}$, $r^{\mathcal{J}} = \{(d_0 \cdots d_k, d_0 \cdots d_{k+1}) \mid (d_0 \cdots d_k, d_0 \cdots d_{k+1}) \in \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} \wedge (d_k, d_{k+1}) \in r^{\mathcal{I}}\}$, and
- for all $a \in \mathsf{N_I}$, $a^{\mathcal{J}} = a^{\mathcal{I}}$.

By induction on the structure of C, it is not hard to show that for all $\mathcal{ALC}$-concepts C and all $w = d_0 \cdots d_k \in \Delta^{\mathcal{J}}$, $d_k \in C^{\mathcal{I}}$ iff $w \in C^{\mathcal{J}}$. Thus, we have

- $\mathcal{J} \not\models C(a)$ since $\mathcal{I} \not\models C(a)$, and
- $\mathcal{J} \models \mathcal{A}''$ since $\mathcal{I} \models \mathcal{A}''$.

Moreover, from the construction of $\mathcal{J}$, $a^{\mathcal{J}}$ is not reachable in G from $b^{\mathcal{J}}$ for all $a, b \in \mathsf{N_I}$ with $a \neq b$.

If $\mathcal{J}$ is a model of $\mathcal{A}'$, then $\mathcal{A}' \not\models C(a)$ since $\mathcal{J} \not\models C(a)$. If $\mathcal{J} \not\models \mathcal{A}'$, then $\mathcal{J} \models \mathcal{A}''$ implies

- $\mathcal{J} \not\models \mathsf{JOHN}{:}\exists\mathsf{hasChild}.(\mathsf{Happy} \sqcup \{\mathsf{MARY}\})$, and
- $\mathcal{J} \models \mathsf{JOHN}{:}\exists\mathsf{hasChild}.(\mathsf{Happy} \sqcup \mathsf{Clever})$.

Thus, there is a $d^* \in (\neg\mathsf{Happy} \sqcap \mathsf{Clever})^{\mathcal{J}}$ such that $(\mathsf{JOHN}^{\mathcal{J}}, d^*) \in \mathsf{hasChild}^{\mathcal{J}}$. We now manipulate $\mathcal{J}$ into a new interpretation $\mathcal{J}'$ and show that $\mathcal{J}' \models \mathcal{A}'$ and $\mathcal{J}' \not\models C(a)$ based on the fact that for all $\mathcal{ALC}$-concepts C, for all interpretation $\mathcal{I}$ and for all $d \in \Delta^{\mathcal{I}}$, $d \in C^{\mathcal{I}}$ is determined by the interpretations of concept and role names on the elements those are reachable from d in G while it is not up to the interpretations of individual names. There are two cases:

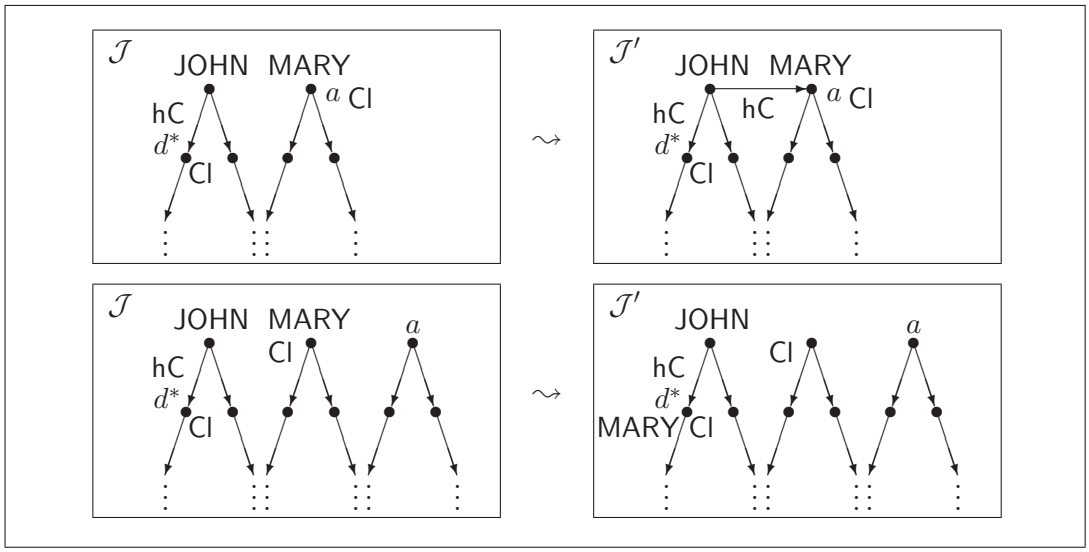


Figure 1: The interpretations $\mathcal{J}$ and $\mathcal{J}'$ in the proof of Lemma 24.

- If $a = \mathsf{MARY}$, then as depicted in $\mathcal{J}$ and $\mathcal{J}'$ in the first row in Figure 1, $\mathcal{J}'$ is obtained from $\mathcal{J}$ by setting $\mathsf{hasChild}^{\mathcal{J}'} = \mathsf{hasChild}^{\mathcal{J}} \cup \{(\mathsf{JOHN}^{\mathcal{J}}, \mathsf{MARY}^{\mathcal{J}})\}$. By the UNA, $a^{\mathcal{J}'} \neq \mathsf{JOHN}^{\mathcal{J}'}$. By the forementioned fact, $\mathcal{J}' \models \mathsf{MARY}{:}\neg\mathsf{Happy} \sqcap \mathsf{Clever}$ and $\mathcal{J}' \not\models C(a)$ still hold. $\mathcal{J}' \models \mathsf{JOHN}{:}\exists\mathsf{hasChild}.(\mathsf{Happy} \sqcup \{\mathsf{MARY}\})$ is a direct consequence of adding $(\mathsf{JOHN}^{\mathcal{J}}, \mathsf{MARY}^{\mathcal{J}})$ into $\mathsf{hasChild}^{\mathcal{J}}$.

- If $a \neq \mathsf{MARY}$, then as depicted in $\mathcal{J}$ and $\mathcal{J}'$ in the second row in Figure 1, $\mathcal{J}'$ is obtained from $\mathcal{J}$ by setting $\mathsf{MARY}^{\mathcal{J}'} = d^*$. From the construction of $\mathcal{J}'$, $\mathcal{J}' \models \mathsf{JOHN}{:}\exists\mathsf{hasChild}.(\mathsf{Happy} \sqcup \{\mathsf{MARY}\})$ and $\mathcal{J}' \not\models C(a)$ still hold. $\mathcal{J}' \models \mathsf{MARY}{:}\neg\mathsf{Happy} \sqcap \mathsf{Clever}$ since $d^* \in (\neg\mathsf{Happy} \sqcap \mathsf{Clever})^{\mathcal{J}'}$.

In both cases, $\mathcal{J}'$ is a model of $\mathcal{A}'$ and $\mathcal{J}' \not\models C(a)$. Thus, $\mathcal{A}' \not\models C(a)$. ❑

We are ready to show that an approximate update of an ABox $\mathcal{A}$ with an update $\mathcal{U}$ w.r.t. a given DL is not necessarily an update of $\mathcal{A}$ with $\mathcal{U}$ w.r.t. its superlanguage.

Lemma 25. *There exist ABoxes $\mathcal{A}$, $\mathcal{A}''$, and an update $\mathcal{U}$ such that $\mathcal{A}''$ is an approximate update of $\mathcal{U}$ w.r.t. $\mathcal{ALC}$ but $\mathcal{A}''$ is not an approximate update of $\mathcal{U}$ w.r.t. $\mathcal{ALCO}$.*

Proof. Consider the ABox $\mathcal{A}$, $\mathcal{A}'$ and the update $\mathcal{U}$ in Example 20 and the ABox $\mathcal{A}''$ in Lemma 24. We know $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{ALC}} \mathcal{A}''$ by Lemma 24. Since $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$ and $\mathcal{A}'$ is an $\mathcal{ALCO}$-ABox, $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{ALCO}} \mathcal{A}'$. By Lemma 23 Point 2, it remains to show that $\mathcal{A}' \not\equiv \mathcal{A}''$.

Consider the interpretation $\mathcal{I}$ displayed in Figure 2. We assume that in addition to the points depicted there is an infinite set of points interpreting the individual names $a \in (\mathsf{N_I} \setminus \{\mathsf{JOHN}, \mathsf{MARY}\})$. On these additional points the concept and role names are interpreted as empty set. The additional points are required to define interpretations

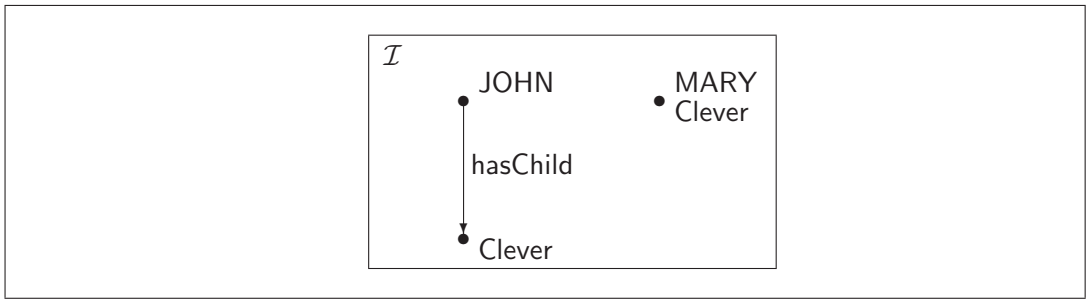


Figure 2: The interpretation $\mathcal{I}$ in the proof of Lemma 25.

satisfying the UNA. It is easy to see that $\mathcal{I} \not\models \mathcal{A}'$ (since $\mathcal{I} \not\models \mathsf{JOHN}{:}\exists\mathsf{hasChild}.(\mathsf{Happy} \sqcup \{\mathsf{MARY}\})$) and $\mathcal{I} \models \mathcal{A}''$. Thus, $\mathcal{A}' \not\equiv \mathcal{A}''$. ❑

Logical updates capture exactly the set of updated models while approximate updates preserve the logical consequences of updated models. Two other formalisms for describing updated ABoxes relativize those two properties to a fixed signature.

Definition 26 (Signature and Reduct)**.** A *signature* S is a subset of $\mathsf{N_C} \cup \mathsf{N_R} \cup \mathsf{N_I}$. The *signature of a concept* C, denoted with $\mathsf{Sig}(C)$, is the set of concept, role, and individual names which occur in C. The *signature of an ABox* $\mathcal{A}$, denoted with $\mathsf{Sig}(\mathcal{A})$, is the set of concept, role, and individual names which occur in $\mathcal{A}$.

Let $\mathcal{I}$ be an interpretation, S a signature, and M a set of interpretations. The *reduct of* $\mathcal{I}$ *to* S, denoted with $\mathcal{I}_{|\mathsf{S}}$, is the interpretation which interprets only the concept, role, and individual names from S and satisfies the following conditions:

- $\Delta^{\mathcal{I}_{|\mathsf{S}}} = \Delta^{\mathcal{I}}$, and
- for all names $x \in \mathsf{S}$, $x^{\mathcal{I}_{|\mathsf{S}}} = x^{\mathcal{I}}$.

The *reduct of* M *to* S is denoted with $M_{|\mathsf{S}}$ and defined by

$$M_{|\mathsf{S}} = \{\mathcal{I}_{|\mathsf{S}} \mid \mathcal{I} \in M\}.$$

△

Based on the above definition, two other formalisms about updating ABoxes can be introduced.

Definition 27 (Projective Update and Approximate Projective Update)**.** Let $\mathcal{L}$ be a DL, $\mathcal{A}$ an $\mathcal{L}$-ABox, $\mathcal{U}$ an update, and S the following signature:

$$\mathsf{S} = \big((\mathsf{N_C} \cup \mathsf{N_R} \cup \mathsf{N_I}) \setminus \mathsf{Sig}(\mathcal{A}')\big) \cup \mathsf{Sig}(\mathcal{A}) \cup \mathsf{Sig}(\mathcal{U}).$$

An $\mathcal{L}$-ABox $\mathcal{A}'$ is

- a *projective update of* $\mathcal{A}$ *with* $\mathcal{U}$, in symbols $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}} \mathcal{A}'$, if

$$M(\mathcal{A} * \mathcal{U})_{|\mathsf{S}} = M(\mathcal{A}')_{|\mathsf{S}}.$$

  The symbols in $\mathsf{Sig}(\mathcal{A}') \setminus (\mathsf{Sig}(\mathcal{A}) \cup \mathsf{Sig}(\mathcal{U}))$ are called the *fresh symbols of the projective update* $\mathcal{A}'$ *of* $\mathcal{A}$ *with* $\mathcal{U}$.

- an *approximate projective update of* $\mathcal{A}$ *with* $\mathcal{U}$ *w.r.t.* $\mathcal{L}$, in symbols $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}}_{\mathcal{L}} \mathcal{A}'$, if for all $\mathcal{L}$-assertions φ with $\mathsf{Sig}(\varphi) \subseteq \mathsf{S}$, we have

$$M(\mathcal{A} * \mathcal{U}) \models \varphi \text{ iff } M(\mathcal{A}') \models \varphi.$$

△

It follows immediately from Definition 27 that a projective update of an ABox $\mathcal{A}$ with an update $\mathcal{U}$ is also an approximate projective update of $\mathcal{A}$ with $\mathcal{U}$ w.r.t. any DL $\mathcal{L}$. Considering the four kinds of updates, we have the following lemma, which is a direct consequence of Definition 19 and Definition 27.

Lemma 28. *Let* $\mathcal{L}$ *be a DL,* $\mathcal{A}$ *and* $\mathcal{A}'$ $\mathcal{L}$*-ABoxes, and* $\mathcal{U}$ *an update. Then,*

- *if* $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$*, then* $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{L}} \mathcal{A}'$ *and* $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}} \mathcal{A}'$*;*
- *if* $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{L}} \mathcal{A}'$ *or* $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}} \mathcal{A}'$*, then* $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}}_{\mathcal{L}} \mathcal{A}'$*.*

Projective updates are not uniquely determined up to logical equivalence because different sets of fresh symbols might be chosen. Nevertheless, similar to logical updates they depend neither on the syntactic presentation of the ABox nor on the underlying DL.

In the context of propositional updates, projective updates have been investigated in detail in [CDLS99]. There, projective updates are discussed w.r.t. a "query equivalence" property. In the setting of updating ABoxes, we call the resulting ABox satisfying this property an approximate projective update. According to their definition, it is only required that

$$\mathsf{S} = \mathsf{Sig}(\mathcal{A}) \cup \mathsf{Sig}(\mathcal{U}).$$

This definition is equivalent to Definition 27 for the logics that have the interpolation property [Cra57], e.g., propositional logic.

In Example 20, the logical update $\mathcal{A}'$ of an $\mathcal{ALC}$-ABox is expressed with the help of the nominal constructor and its approximate update w.r.t. $\mathcal{ALC}$ exists. Is the nominal constructor essential to express the logical update for an $\mathcal{ALC}$-ABox? Or, more generally, which DLs are closed under logical updates? Which constructors are not necessary any more by considering weaker updates than logical ones? Those problems are addressed in this thesis. The related notion of a DL having updates is introduced in the next definition.

Definition 29. Let $\mathcal{L}$ be a DL. An update $\mathcal{U}$ is an $\mathcal{L}$*-update* if φ is an $\mathcal{L}$-assertion for all $\varphi/\psi \in \mathcal{U}$. The DL $\mathcal{L}$

- *has logical updates* iff, for every $\mathcal{L}$-ABox $\mathcal{A}$ and every $\mathcal{L}$-update $\mathcal{U}$, there exists an $\mathcal{L}$-ABox $\mathcal{A}'$ such that $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$.
- *has approximate updates* iff, for every $\mathcal{L}$-ABox $\mathcal{A}$ and every $\mathcal{L}$-update $\mathcal{U}$, there exists an $\mathcal{L}$-ABox $\mathcal{A}'$ such that $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{L}} \mathcal{A}'$.
- *has projective updates* iff, for every $\mathcal{L}$-ABox $\mathcal{A}$ and every $\mathcal{L}$-update $\mathcal{U}$, there exists an $\mathcal{L}$-ABox $\mathcal{A}'$ such that $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}} \mathcal{A}'$.

- *has approximate projective updates* iff, for every $\mathcal{L}$-ABox $\mathcal{A}$ and every $\mathcal{L}$-update $\mathcal{U}$, there exists an $\mathcal{L}$-ABox $\mathcal{A}'$ such that $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}}_{\mathcal{L}} \mathcal{A}'$.

△

Logical updates are obviously the most desirable form of updates in the sense that they describe exactly the set of updated models without using additional non-logical symbols. When the expressivity of the DL under consideration is inadequate, one might choose other weaker kinds of updates. Projective updates preserve the logical consequences of logical updates in a restricted signature in any DL, while approximate updates are a reasonable alternative if we are interested in logical consequences in a fixed DL but in an unrestricted signature. If neither of even those two updates exists, then one may resort to approximate projective update as they completely describe consequences in a restricted signature of logical updates in a given DL.

Another reason for choosing projective updates over logical updates is the size of the updated ABoxes. As we will see in Chapter 3, the construction of logical updates is exponential in the size of the original ABox together with the update, and this blowup cannot be avoided unless every PTime algorithm is LogTime-parallelizable. The projective update computed as described in Chapter 4 is polynomial both in the size of the original ABox and in the size of the update. The size of resulting ABoxes not only makes an impact on the space to store them but also usually influences the time to do reasoning with them. We will show this by experimental results presented in Chapter 6.

A fundamental problem in reasoning about action is *projection*, which is to determine whether or not some effect that we usually want to make true really holds after applying a given finite sequence of updates to the initial world description. In action formalisms based on DLs, the projection problem was investigated in [BLM+05].

Definition 30 (Projection)**.** Let $\mathcal{A}$ be an ABox and $\mathcal{U}_1 \cdots \mathcal{U}_n$ a finite sequence of updates. An assertion φ is a *consequence of applying* $\mathcal{U}_1 \cdots \mathcal{U}_n$ *to* $\mathcal{A}$ (denoted by $\mathcal{A}^{\mathcal{U}_1 \cdots \mathcal{U}_n} \models \varphi$) iff for all models $\mathcal{I}$ of $\mathcal{A}$, and all interpretations $\mathcal{I}'$ with $\mathcal{I} \Longrightarrow_{\mathcal{U}_1 \cdots \mathcal{U}_n} \mathcal{I}'$, we have $\mathcal{I}' \models \varphi$. △

There are basically two mechanisms in action theory to solve the projection problem: *regression* and *progression*. Roughly speaking, regression reduces checking whether the desired effect φ holds after the application of the updates $\mathcal{U}_1 \cdots \mathcal{U}_n$ ($\mathcal{A}^{\mathcal{U}_1 \cdots \mathcal{U}_n} \models \varphi$) to verifying whether a rewritten effect φ' is a logical consequence of the initial world description $\mathcal{A}$ ($\mathcal{A} \models \varphi'$). Regression is goal-oriented: Different effects often require checking different reduced logical consequences. The action programming language Golog [LRL+97] which is based on the Situation Calculus [Rei01] adopts regression to solve the projection problem. In [BLM+05], projection is solved by an approach similar to regression. More specifically, for an ABox $\mathcal{A}$, a finite sequence of updates $\mathcal{U}_1 \cdots \mathcal{U}_n$ and an ABox assertion φ (an input of the projection problem) in a DL $\mathcal{L}$ between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$, we construct an $\mathcal{LO}$-ABox $\mathcal{A}_{\mathsf{red}}$, an acyclic $\mathcal{LO}$-TBox $\mathcal{T}_{\mathsf{red}}$, and an $\mathcal{L}$-ABox assertion φ_{red} such that φ is a consequence of applying $\mathcal{U}_1 \cdots \mathcal{U}_n$ to $\mathcal{A}$ iff φ_{red} is a logical consequence of $\mathcal{A}_{\mathsf{red}}$ w.r.t. $\mathcal{T}_{\mathsf{red}}$. One observation is that the constructed $\mathcal{T}_{\mathsf{red}}$ and φ_{red} are goal-oriented, i.e., they depend on φ.

In contrast to regression, progression changes the initial world description according to updates and then checks whether the desired effect holds in the resulting world description. One advantage of progression is that different effects may be checked in the same resulting world without any extra overhead. The action programming language FLUX [Thi05a] which is based on the Fluent Calculus [Thi05b] uses progression to do reasoning about action. Computing updates of ABoxes integrates progression into DLs and it provides a solution to the projection problem. To check whether φ is a consequence of applying $\mathcal{U}_1 \cdots \mathcal{U}_n$ to $\mathcal{A}$ ($\mathcal{A}^{\mathcal{U}_1 \cdots \mathcal{U}_n} \models \varphi$), we can compute the updated ABox $\mathcal{A}'$ by iteratively updating $\mathcal{A}$ with the updates $\mathcal{U}_1 \cdots \mathcal{U}_n$ and then we check whether φ is a logical consequence of $\mathcal{A}'$ ($\mathcal{A}' \models \varphi$).

Some experimental results will be presented in Section 6.3 on implementations that can solve the projection problem in DLs. They are respectively based on regression and progression.

Chapter 3

Logical Updates

We focus on the computation of logical updates and the analysis of the size of the computed logical updates in this chapter. An algorithm for computing the logical update of an $\mathcal{ALCQIO}^{@}$-ABox with a single update is presented in Section 3.1. The size of the logical update achieved this way is exponential both in the size of the original ABox and in the size of the update in the worst case. In Section 3.2, it is shown that the exponential blowup cannot be entirely avoided unless the complexity classes PTIME and NC are identical, which is believed to be similarly unlikely as PTIME = NP [Pap94]. Two ways to avoid the exponential blowup in the size of the ABox are exhibited in Section 3.3: to allow only for concept literals as effects of updates and to compute logical updates in a more expressive DL with more role constructors than any a DL introduced so far. In Section 3.4, we show that the blowup produced by iterated updates is not worse than the blowup produced by a single update.

3.1 Computing Logical Updates in $\mathcal{ALCQIO}^{@}$

In this section, we show that the expressive DL $\mathcal{ALCQIO}^{@}$ has logical updates. Moreover, the proof is easily adapted to the fragments of $\mathcal{ALCQIO}^{@}$ obtained by dropping qualified number restrictions, inverse roles, or both. In Chapter 5, we will see that the @ constructor is necessary for expressing even approximate updates and that approximate updates and projective updates may not exist without nominals. This yields that for the DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}^{@}$, $\mathcal{ALCO}^{@}$ is the smallest DL which has logical updates.

Our construction of updated ABoxes is an extension of the corresponding construction for propositional logic described in [Win90]. In what follows, we start with the simple case, computing the logical update of an ABox with an *unconditional* update, and continue with extending the computation to assemble updated ABoxes by conditional updates. Computing the logical update of an ABox $\mathcal{A}$ with an unconditional update $\mathcal{U}$ is reduced to updating assertions in $\mathcal{A}$ with $\mathcal{U}$.

Let us start from coping with concept assertions. We proceeds as follows. First, we consider *updates of concepts* on the level of interpretations. More precisely, we

show how to convert a concept C and an unconditional update $\mathcal{U}$ into a concept $C^{\mathcal{U}}$ such that the following property holds: for all interpretations $\mathcal{I}$ and $\mathcal{I}'$ such that $\mathcal{I}$ satisfies *no* assertions in $\mathcal{U}$ and $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$, we have $C^{\mathcal{I}} = (C^{\mathcal{U}})^{\mathcal{I}'}$.

Intuitively, $C^{\mathcal{U}}$ can be used after the update to describe exactly those objects that have been in the extension of C before the update. The aim is to use the translation $C^{\mathcal{U}}$ to update concept assertions in ABoxes.

The limitation that $C^{\mathcal{U}}$ satisfies this property only if $\mathcal{I}$ satisfies no assertion in $\mathcal{U}$ can be overcome by replacing $C^{\mathcal{U}}$ with $C^{\mathcal{U}'}$, where $\mathcal{U}'$ is the set of those assertions in $\mathcal{U}$ that are violated in $\mathcal{I}$. However, we are confronted with the problem that ABoxes have many different models, and these models can violate *different* subsets of the update $\mathcal{U}$. Hence, there is no unique way of moving from $C^{\mathcal{U}}$ to $C^{\mathcal{U}'}$ as described above. The solution is to produce an updated ABox for each subset $\mathcal{U}' \subseteq \mathcal{U}$ separately, and then simply take the disjunction. This idea can also be adopted to the case of conditional updates. Because it is not clear which effects in an update are triggered in different interpretations, we update the original ABox with all subsets of effects and take the disjunction.

We first introduce a bit of notation. Throughout this thesis, we use $\mathsf{Sub}(\mathcal{A})$ to denote the set of the subconcepts occurring in $\mathcal{A}$, i.e.,

$$\mathsf{Sub}(\mathcal{A}) = \bigcup_{C(a) \in \mathcal{A}} \mathsf{Sub}(C).$$

We use $\mathsf{Obj}(\mathcal{A})$ to denote the set of individual names in the ABox $\mathcal{A}$. For an unconditional update $\mathcal{U}$, we use $\neg\mathcal{U}$ to denote $\{\dot{\neg}\varphi \mid \varphi \in \mathcal{U}\}$, where $\dot{\neg}\varphi$ denotes the assertion obtained by eliminating double negation in $\neg\varphi$. Let r be a role and a, b two individual names. Then, we define

$$r(a,b)\dot{\in}\mathcal{U} = \begin{cases} r(a,b) \in \mathcal{U} & \text{if } r \in \mathsf{N_R} \\ s(b,a) \in \mathcal{U} & \text{if } r = s^- \text{ with } s \in \mathsf{N_R} \end{cases}$$

and

$$\neg r(a,b)\dot{\in}\mathcal{U} = \begin{cases} \neg r(a,b) \in \mathcal{U} & \text{if } r \in \mathsf{N_R} \\ \neg s(b,a) \in \mathcal{U} & \text{if } r = s^- \text{ with } s \in \mathsf{N_R} \end{cases}$$

Finally, remember that (cf. Definition 17), for an interpretation $\mathcal{I}$, $\mathcal{I}^{\mathcal{U}}$ denotes the unique interpretation satisfying $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}^{\mathcal{U}}$. The inductive translation that takes a concept C and an unconditional update $\mathcal{U}$ to a concept $C^{\mathcal{U}}$ as explained above is given in Figure 3. Using this translation, we can show the following lemma:

Lemma 31. *Let $\mathcal{U}$ be an unconditional update and C an $\mathcal{ALCQIO}^{@}$-concept. For every interpretation $\mathcal{I}$ with $\mathcal{I} \models \neg\mathcal{U}$ and every individual name a, we have $\mathcal{I} \models C(a)$ iff $\mathcal{I}^{\mathcal{U}} \models C^{\mathcal{U}}(a)$.*

Proof. We first show the following claim:

Claim. If $\mathcal{I} \models \neg\mathcal{U}$, then, for all $x, y \in \Delta^{\mathcal{I}}$ and for all role names $r \in \mathsf{N_R}$, we have $(x,y) \in r^{\mathcal{I}}$ iff one of the following holds:

1. $x \neq a^{\mathcal{I}}$ for all $a \in \mathsf{Obj}(\mathcal{U})$ and $(x,y) \in r^{\mathcal{I}^{\mathcal{U}}}$;

$$
\begin{array}{rcl}
A^{\mathcal{U}} & = & \big(A \sqcup \bigsqcup_{\neg A(a)\in\mathcal{U}} \{a\}\big) \sqcap \neg\big(\bigsqcup_{A(a)\in\mathcal{U}} \{a\}\big), \text{ if } A \in \mathsf{N_C} \\
\{a\}^{\mathcal{U}} & = & \{a\} \qquad\qquad \top^{\mathcal{U}} = \top \qquad\qquad \bot^{\mathcal{U}} = \bot \\
(@_a C)^{\mathcal{U}} & = & @_a(C^{\mathcal{U}}) \qquad (\neg C)^{\mathcal{U}} = \neg C^{\mathcal{U}} \\
(C \sqcap D)^{\mathcal{U}} & = & C^{\mathcal{U}} \sqcap D^{\mathcal{U}} \qquad (C \sqcup D)^{\mathcal{U}} = C^{\mathcal{U}} \sqcup D^{\mathcal{U}} \\
(\geqslant n\ r\ C)^{\mathcal{U}} & = & \big(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\} \sqcap (\geqslant n\ r\ C^{\mathcal{U}})\big) \\
& & \sqcup \bigsqcup_{a\in\mathsf{Obj}(\mathcal{U})} \Big(\{a\} \sqcap \bigsqcup_{\substack{n_1+n_2+n_3=n \\ n_2,n_3\leq \#\mathsf{Obj}(\mathcal{U})}} \big((\geqslant n_1\ r\ ((\bigsqcap_{b\in\mathsf{Obj}(\mathcal{U})} \neg\{b\}) \sqcap C^{\mathcal{U}})) \\
& & \qquad \sqcap (\geqslant n_2\ r\ ((\bigsqcup_{b\in\mathsf{Obj}(\mathcal{U}), r(a,b)\dot{\notin}\mathcal{U}} \{b\}) \sqcap C^{\mathcal{U}})) \\
& & \qquad \sqcap \bigsqcup_{S\subseteq\{b\mid \neg r(a,b)\dot{\in}\mathcal{U}\}, \#S=n_3} \bigsqcap_{b\in S} @_b C^{\mathcal{U}}\big)\Big) \\
(\leqslant n\ r\ C)^{\mathcal{U}} & = & \big(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\} \rightarrow (\leqslant n\ r\ C^{\mathcal{U}})\big) \\
& & \sqcap \bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \Big(\{a\} \rightarrow \bigsqcap_{\substack{n_1+n_2+n_3=n+1 \\ n_2,n_3\leq \#\mathsf{Obj}(\mathcal{U})}} \big(\neg(\geqslant n_1\ r\ ((\bigsqcap_{b\in\mathsf{Obj}(\mathcal{U})} \neg\{b\}) \sqcap C^{\mathcal{U}})) \\
& & \qquad \sqcup \neg(\geqslant n_2\ r\ ((\bigsqcup_{b\in\mathsf{Obj}(\mathcal{U}), r(a,b)\dot{\notin}\mathcal{U}} \{b\}) \sqcap C^{\mathcal{U}})) \\
& & \qquad \sqcup \bigsqcap_{S\subseteq\{b\mid \neg r(a,b)\dot{\in}\mathcal{U}\}, \#S=n_3} \bigsqcup_{b\in S} \neg @_b C^{\mathcal{U}}\big)\Big)
\end{array}
$$

Figure 3: Constructing $C^{\mathcal{U}}$.

2. $x = a^{\mathcal{I}}$ for some $a \in \mathsf{Obj}(\mathcal{U})$ and
   a. $y \neq b^{\mathcal{I}}$ for all $b \in \mathsf{Obj}(\mathcal{U})$ and $(x, y) \in r^{\mathcal{I}^{\mathcal{U}}}$,
   b. or $y = b^{\mathcal{I}}$ for some $b \in \mathsf{Obj}(\mathcal{U})$ such that $r(a, b) \notin \mathcal{U}$ and $(x, y) \in r^{\mathcal{I}^{\mathcal{U}}}$,
   c. or $y = b^{\mathcal{I}}$ for some $b \in \mathsf{Obj}(\mathcal{U})$ such that $\neg r(a, b) \in \mathcal{U}$.

Proof of the claim: "⇒": In cases 1 and 2a, at least one of x, y is an anonymous object of $\mathcal{I}$. 1 and 2a follow from the fact that updating an interpretation $\mathcal{I}$ can make changes only on the named objects of $\mathcal{I}$. If both x and y are named objects of $\mathcal{I}$ (as in the cases 2b and 2c), then $r(a, b) \notin \mathcal{U}$ since $\mathcal{I} \models \neg\mathcal{U}$ and $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$. It is enough to show that 2c does not hold implies that 2b holds. If $\neg r(a, b) \notin \mathcal{U}$, then it follows from Definition 17 that $(x, y) \in r^{\mathcal{I}}$ implies $(x, y) \in r^{\mathcal{I}^{\mathcal{U}}}$. Thus, 2b holds.

"⇐": In cases 1 and 2a, at least one of x, y is an anonymous object of $\mathcal{I}$. Thus, either of 1 and 2a implies $(x, y) \in r^{\mathcal{I}}$ since updating an interpretation $\mathcal{I}$ can make

changes only on the named objects of $\mathcal{I}$. For the case 2b, $(x, y) \in r^{\mathcal{I}}$ is a direct consequence of Definition 17. For the case 2c, since $\neg r(a, b) \in \mathcal{U}$ and $\mathcal{I} \models \neg\mathcal{U}$, we have $(x, y) \in r^{\mathcal{I}}$. This finishes the proof of the claim.

Let $\mathcal{I}$ be an interpretation such that $\mathcal{I} \models \neg\mathcal{U}$ and let $E \in \mathsf{Sub}(\mathcal{A})$. By structural induction on E, we show that $(E^{\mathcal{U}})^{\mathcal{I}^{\mathcal{U}}} = E^{\mathcal{I}}$. As $\mathcal{I}$ and $\mathcal{I}^{\mathcal{U}}$ interpret all individuals in the same way, this implies Lemma 31.

- The cases $E = \{a\}$, $E = \top$, and $E = \bot$ are trivial since $\mathcal{I}$ and $\mathcal{I}^{\mathcal{U}}$ interpret all individual names, $\top$, and $\bot$ in the same way.
- $E = A$, for A a concept name: then

$$\begin{aligned}(A^{\mathcal{U}})^{\mathcal{I}^{\mathcal{U}}} &= \Big(A^{\mathcal{I}^{\mathcal{U}}} \cup \bigcup_{\neg A(a)\in\mathcal{U}} \{a^{\mathcal{I}^{\mathcal{U}}}\}\Big) \setminus \bigcup_{A(a)\in\mathcal{U}} \{a^{\mathcal{I}^{\mathcal{U}}}\} \\ &= \Big(\Big(\Big(A^{\mathcal{I}} \cup \bigcup_{A(a)\in\mathcal{U}} \{a^{\mathcal{I}}\}\Big) \setminus \bigcup_{\neg A(a)\in\mathcal{U}} \{a^{\mathcal{I}}\}\Big) \cup \bigcup_{\neg A(a)\in\mathcal{U}} \{a^{\mathcal{I}^{\mathcal{U}}}\}\Big) \setminus \bigcup_{A(a)\in\mathcal{U}} \{a^{\mathcal{I}^{\mathcal{U}}}\} \\ &= A^{\mathcal{I}}\end{aligned}$$

  since $A^{\mathcal{I}} \cap \bigcup_{A(a)\in\mathcal{U}} \{a^{\mathcal{I}}\} = \emptyset$ and $\bigcup_{\neg A(a)\in\mathcal{U}} \{a^{\mathcal{I}}\} \subseteq A^{\mathcal{I}}$ due to $\mathcal{I} \models \neg\mathcal{U}$.
- $E = @_a C$: $((@_a C)^{\mathcal{U}})^{\mathcal{I}^{\mathcal{U}}} = (@_a (C^{\mathcal{U}}))^{\mathcal{I}^{\mathcal{U}}} = (@_a C)^{\mathcal{I}}$ since $(C^{\mathcal{U}})^{\mathcal{I}^{\mathcal{U}}} = C^{\mathcal{I}}$ and $\mathcal{I}$ and $\mathcal{I}^{\mathcal{U}}$ interpret individuals in the same way.
- The cases $E = \neg C$, $E = C \sqcap D$, and $E = C \sqcup D$ follow immediately from I.H. and the definition of $E^{\mathcal{U}}$.
- $E = (\geqslant n\ r\ C)$: Here we show in detail the proof only in the case that r is a role name. The case that r is an inverse role can be proved analogously. The intuition of the construction of $(\geqslant n\ r\ C)^{\mathcal{U}}$ is depicted in Figure 4, where a solid circle, a hollow circle, and a star respectively stand for a named, an anonymous, and an arbitrary object of $\mathcal{I}$.

  $x \in ((\geqslant n\ r\ C)^{\mathcal{U}})^{\mathcal{I}^{\mathcal{U}}}$ iff (by the definition of $(\geqslant n\ r\ C)^{\mathcal{U}}$) one of the following holds:

  1. $x \in \big(\neg \bigsqcup_{a\in\mathsf{Obj}(\mathcal{U})} \{a\}\big)^{\mathcal{I}^{\mathcal{U}}}$ and $\#\{y \mid (x, y) \in r^{\mathcal{I}^{\mathcal{U}}} \wedge y \in (C^{\mathcal{U}})^{\mathcal{I}^{\mathcal{U}}}\} \geq n$;
  2. $x = a^{\mathcal{I}^{\mathcal{U}}}$, for some $a \in \mathsf{Obj}(\mathcal{U})$ and there are $n_1, n_2, n_3 \geq 0$ such that $n_1 + n_2 + n_3 = n$, $n_2, n_3 \leq \#\mathsf{Obj}(\mathcal{U})$, and
     a. $\#\{y \mid (x, y) \in r^{\mathcal{I}^{\mathcal{U}}} \wedge y \in \big(\bigsqcap_{b\in\mathsf{Obj}(\mathcal{U})} \neg\{b\}\big)^{\mathcal{I}^{\mathcal{U}}} \cap (C^{\mathcal{U}})^{\mathcal{I}^{\mathcal{U}}}\} \geq n_1$,
     b. $\#\{y \mid (x, y) \in r^{\mathcal{I}^{\mathcal{U}}} \wedge y \in (\bigcup_{b\in\mathsf{Obj}(\mathcal{U}), r(a,b)\notin\mathcal{U}} \{b\}^{\mathcal{I}^{\mathcal{U}}}) \cap (C^{\mathcal{U}})^{\mathcal{I}^{\mathcal{U}}}\} \geq n_2$, and
     c. $\#\{b \mid \neg r(a, b) \in \mathcal{U} \wedge b^{\mathcal{I}^{\mathcal{U}}} \in (C^{\mathcal{U}})^{\mathcal{I}^{\mathcal{U}}}\} \geq n_3$.

  Note that three sets in 2a, 2b, and 2c are pairwise disjoint. By I.H., we have that $(C^{\mathcal{U}})^{\mathcal{I}^{\mathcal{U}}} = C^{\mathcal{I}}$. Thus, using the claim above, we obtain that $x \in ((\geqslant n\ r\ C)^{\mathcal{U}})^{\mathcal{I}^{\mathcal{U}}}$ iff one of the following holds:

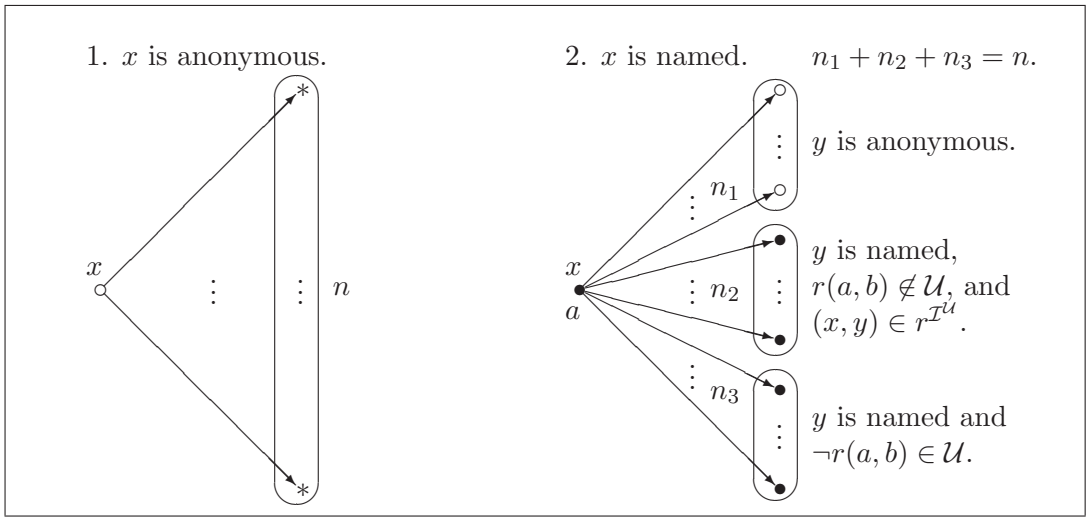


Figure 4: The case $E = (\geqslant n\ r\ C)$ in the proof of Lemma 31.

1. $x \in \left(\neg \bigsqcup_{a \in \mathsf{Obj}(\mathcal{U})} \{a\}\right)^{\mathcal{I}}$ and $\#\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n$;
2. $x = a^{\mathcal{I}}$, for some $a \in \mathsf{Obj}(\mathcal{U})$ and there are $n_1, n_2, n_3 \geq 0$ such that $n_1 + n_2 + n_3 = n$, $n_2, n_3 \leq \#\mathsf{Obj}(\mathcal{U})$, and
   a. $\#\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in \left(\bigsqcap_{b \in \mathsf{Obj}(\mathcal{U})} \neg\{b\}\right)^{\mathcal{I}} \cap C^{\mathcal{I}}\} \geq n_1$,
   b. $\#\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in (\bigcup_{b \in \mathsf{Obj}(\mathcal{U}), r(a,b) \notin \mathcal{U}} \{b\}^{\mathcal{I}}) \cap C^{\mathcal{I}}\} \geq n_2$, and
   c. $\#\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in (\bigcup_{\neg r(a,b) \in \mathcal{U}} \{b\}^{\mathcal{I}}) \cap C^{\mathcal{I}}\} \geq n_3$.

Note that $n_2, n_3 \leq \#\mathsf{Obj}(\mathcal{U})$ since for every y in the sets in 2b and 2c there exists some $b \in \mathsf{Obj}(\mathcal{U})$ such that $y = b^{\mathcal{I}}$. Further, by the semantics of qualified number restrictions, this is equivalent to

1. $x \in \left(\neg \bigsqcup_{a \in \mathsf{Obj}(\mathcal{U})} \{a\}\right)^{\mathcal{I}}$ and $\#\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n$, or
2. $x \in \left(\bigsqcup_{a \in \mathsf{Obj}(\mathcal{U})} \{a\}\right)^{\mathcal{I}}$ and $\#\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n$.

which is equivalent to $x \in (\geqslant n\ r\ C)^{\mathcal{I}}$.

- The case $E = (\leqslant n\ r\ C)$ is proved similarly to the previous case.

❑

We now extend the update of concepts to the update of ABoxes, while still remaining on the level of interpretations. We have seen that the translation $C^{\mathcal{U}}$ is used to update concept assertions. Additionally, we need to deal with role assertions. Intuitively, each role assertion remains the same if it does not contradict the update. Let $\mathcal{A}$ be an ABox and $\mathcal{U}$ an unconditional update. If we have $\varphi \dot{\in} \mathcal{U}$ for some role assertion $\varphi \in \mathcal{A}$, then $\mathcal{A}^{\mathcal{U}}$ is defined as $\{\bot(a)\}$ for an arbitrary $a \in \mathsf{N_I}$. Otherwise, we

define the ABox $\mathcal{A}^{\mathcal{U}}$ by setting

$$\begin{aligned}\mathcal{A}^{\mathcal{U}} \;=\; & \{C^{\mathcal{U}}(a) \mid C(a) \in \mathcal{A}\} \cup \\ & \{r(a,b) \mid r(a,b) \in \mathcal{A} \wedge \neg r(a,b) \dot{\notin} \mathcal{U}\} \cup \\ & \{\neg r(a,b) \mid \neg r(a,b) \in \mathcal{A} \wedge r(a,b) \dot{\notin} \mathcal{U}\}.\end{aligned}$$

We show the following lemma:

Lemma 32. *Let $\mathcal{A}$ be an $\mathcal{ALCQIO}^{@}$-ABox and $\mathcal{U}$ an unconditional update. If $\mathcal{A}^{\mathcal{U}} \neq \{\bot(a)\}$ for all $a \in \mathsf{N_I}$, then for every interpretation $\mathcal{I}$ with $\mathcal{I} \models \neg\mathcal{U}$, we have $\mathcal{I} \models \mathcal{A}$ iff $\mathcal{I}^{\mathcal{U}} \models \mathcal{A}^{\mathcal{U}}$.*

Proof. "⇒": Let $\mathcal{I}$ be a model of $\mathcal{A}$. We show that $\mathcal{I}^{\mathcal{U}} \models \mathcal{A}^{\mathcal{U}}$, i.e., for every $\varphi \in \mathcal{A}^{\mathcal{U}}$, $\mathcal{I}^{\mathcal{U}} \models \varphi$. Let $\varphi = r(a,b)$. By the construction of $\mathcal{A}^{\mathcal{U}}$, $r(a,b) \in \mathcal{A}^{\mathcal{U}}$ implies that $r(a,b) \in \mathcal{A}$ and that $\neg r(a,b) \dot{\in} \mathcal{U}$ does not hold. Thus, we know that $\mathcal{I} \models r(a,b)$ since $\mathcal{I} \models \mathcal{A}$. Moreover, by the definition of $\mathcal{I}^{\mathcal{U}}$, $\mathcal{I}^{\mathcal{U}} \models r(a,b)$. The case $\varphi = \neg r(a,b)$ can be proved analogously. If φ is a concept assertion $E^{\mathcal{U}}(a)$ for some $E(a) \in \mathcal{A}$, it follows from Lemma 31 that $\mathcal{I}^{\mathcal{U}} \models E^{\mathcal{U}}(a)$.

"⇐": Let $\mathcal{I}^{\mathcal{U}} \models \mathcal{A}^{\mathcal{U}}$. We show that $\mathcal{I} \models \mathcal{A}$. Let $\varphi \in \mathcal{A}$. If $\varphi = r(a,b)$, there are two cases to consider:

1. $\neg r(a,b) \dot{\in} \mathcal{U}$. Then $r(a,b) \in \neg\mathcal{U}$, and since $\mathcal{I} \models \neg\mathcal{U}$, we obtain that $\mathcal{I} \models r(a,b)$.
2. $\neg r(a,b) \dot{\notin} \mathcal{U}$. Then $r(a,b) \in \mathcal{A}^{\mathcal{U}}$, and thus $\mathcal{I}^{\mathcal{U}} \models r(a,b)$. Since $\mathcal{I}^{\mathcal{U}} \models \mathcal{A}^{\mathcal{U}}$ and $r(a,b)$ is in $\mathcal{A}$, $r(a,b) \dot{\in} \mathcal{U}$ does not hold. By definition of $\mathcal{I}^{\mathcal{U}}$, we obtain $\mathcal{I} \models r(a,b)$.

The case $\varphi = \neg r(a,b)$ is analogous and the case $\varphi = E(a)$ follows from Lemma 31. ❑

We are now in the position to lift updates from the level of interpretations to the level of ABoxes and to extend it to the case of conditional updates. For a conditional update $\mathcal{U}$, we use $\mathsf{rhs}(\mathcal{U})$ to denote the set of the effects of $\mathcal{U}$, i.e., $\mathsf{rhs}(\mathcal{U}) = \{\psi \mid \varphi/\psi \in \mathcal{U}\}$. The set of *literals* over $\mathsf{rhs}(\mathcal{U})$ is defined as $L_{\mathcal{U}} = \{\psi, \dot{\neg}\psi \mid \psi \in \mathsf{rhs}(\mathcal{U})\}$. A simple ABox $\mathcal{D}$ is called a *diagram for* $\mathsf{rhs}(\mathcal{U})$ if it is a maximal consistent subset of $L_{\mathcal{U}}$, i.e., there is no strict superset of $\mathcal{D}$ which is also a consistent subset of $L_{\mathcal{U}}$. Let $\mathfrak{D}$ be the set of all diagrams for $\mathsf{rhs}(\mathcal{U})$. Let $\mathcal{D} \in \mathfrak{D}$ and $\mathcal{U}' \subseteq \mathcal{U}$. We define the ABox $\mathcal{D}_{\mathcal{U}'}$ as

$$\mathcal{D}_{\mathcal{U}'} = \{\psi \mid \dot{\neg}\psi \in \mathcal{D} \text{ and } \varphi/\psi \in \mathcal{U}'\}.$$

It follows from the above definition that $\mathcal{D}_{\mathcal{U}'}$ is a simple ABox. The next lemma shows more properties of $\mathcal{D}_{\mathcal{U}'}$.

Lemma 33. *Let $\mathcal{U}$ be an update and $\mathfrak{D}$ the set of all diagrams for $\mathsf{rhs}(\mathcal{U})$. Then, for all $\mathcal{D} \in \mathfrak{D}$ and all $\mathcal{U}' \subseteq \mathcal{U}$,*

1. *$\mathcal{D}_{\mathcal{U}'}$ is consistent.*

2. *for all interpretations* $\mathcal{I}$*, if* $\mathcal{I} \models \mathcal{D}$ *and* $\mathcal{I} \models \neg\mathcal{D}_{\mathcal{U}'}$*, where* $\mathcal{U}' = \{\varphi/\psi \in \mathcal{U} \mid \mathcal{I} \models \varphi\}$*, then* $\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}} = \mathcal{I}^{\mathcal{U}}$*.*

3. *for all interpretations* $\mathcal{I}$*, if* $\mathcal{I} \models \mathcal{D}_{\mathcal{U}'}$*, then* $(\mathcal{I}^{\neg\mathcal{D}_{\mathcal{U}'}})^{\mathcal{D}_{\mathcal{U}'}} = \mathcal{I}$*.*

4. *for all interpretations* $\mathcal{I}$*, if* $\mathcal{I} \models \neg\mathcal{D}_{\mathcal{U}'}$ *and* $\mathcal{I} \models \mathcal{D}$*, then for all ABoxes* $\mathcal{A}$*,* $\mathcal{I} \models \mathcal{A}$ *implies that* $\mathcal{A}^{\mathcal{D}_{\mathcal{U}'}} \neq \{\bot(a)\}$ *for all* $a \in \mathsf{N_I}$*.*

Proof.

1. Assume that there exist a $\mathcal{D} \in \mathfrak{D}$ and a $\mathcal{U}' \subseteq \mathcal{U}$ such that $\mathcal{D}_{\mathcal{U}'}$ is inconsistent. Since $\mathcal{D}_{\mathcal{U}'}$ is a simple ABox, its inconsistency implies that there exists a ψ such that $\{\psi, \dot{\neg}\psi\} \subseteq \mathcal{D}_{\mathcal{U}'}$. Thus, by the definition of $\mathcal{D}_{\mathcal{U}'}$, we have $\{\psi, \dot{\neg}\psi\} \subseteq \mathcal{D}$, which implies that $\mathcal{D}$ is inconsistent. However, the assumption that $\mathcal{D}$ is a diagram for $\mathsf{rhs}(\mathcal{U})$ requires that $\mathcal{D}$ is consistent.

2. Since $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}^{\mathcal{U}}$ and $\mathcal{I} \Longrightarrow_{\mathcal{D}_{\mathcal{U}'}} \mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}$, it follows from Definition 17 that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{I}^{\mathcal{U}}} = \Delta^{\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}}$ and $a^{\mathcal{I}} = a^{\mathcal{I}^{\mathcal{U}}} = a^{\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}}$ for all $a \in \mathsf{N_I}$. It remains to show that for all $X \in \mathsf{N_C} \cup \mathsf{N_R}$, $X^{\mathcal{I}^{\mathcal{U}}} = X^{\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}}$. Here, we only show this in the case that X is a concept name. The case that X is a role name can be shown similarly. Let A be a concept name. From Definition 17, we know the following:

$$\begin{aligned} A^{\mathcal{I}^{\mathcal{U}}} &= (A^{\mathcal{I}} \cup \{a^{\mathcal{I}} \mid \varphi/A(a) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}) \setminus \{a^{\mathcal{I}} \mid \varphi/\neg A(a) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}. \\ A^{\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}} &= (A^{\mathcal{I}} \cup \{a^{\mathcal{I}} \mid A(a) \in \mathcal{D}_{\mathcal{U}'}\}) \setminus \{a^{\mathcal{I}} \mid \neg A(a) \in \mathcal{D}_{\mathcal{U}'}\}. \end{aligned}$$

   "$\subseteq$": Assume that there exists an $x \in \Delta^{\mathcal{I}}$ such that $x \in A^{\mathcal{I}^{\mathcal{U}}} \setminus A^{\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}}$. Thus, $x \in A^{\mathcal{I}^{\mathcal{U}}}$ implies that

   - $x \in A^{\mathcal{I}}$ and $x \notin \{a^{\mathcal{I}} \mid \varphi/\neg A(a) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}$, or
   - $x \notin A^{\mathcal{I}}$, $x \in \{a^{\mathcal{I}} \mid \varphi/A(a) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}$ and $x \notin \{a^{\mathcal{I}} \mid \varphi/\neg A(a) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}$.

   In the former case, $x \notin A^{\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}}$ implies that there exists some $\neg A(a) \in \mathcal{D}_{\mathcal{U}'}$ with $x = a^{\mathcal{I}}$. By the definition of $\mathcal{D}_{\mathcal{U}'}$, we know that $A(a) \in \mathcal{D}$ and there exists a φ such that $\varphi/\neg A(a) \in \mathcal{U}'$. By the definition of $\mathcal{U}'$, we get $\mathcal{I} \models \varphi$ and $\varphi/\neg A(a) \in \mathcal{U}$. Thus, $a^{\mathcal{I}} \notin A^{\mathcal{I}^{\mathcal{U}}}$, which contradicts the assumption. In the latter case, $x \in \{a^{\mathcal{I}} \mid \varphi/A(a) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}$ implies that there exists some $\varphi/A(a) \in \mathcal{U}$ such that $a^{\mathcal{I}} = x$ and $\mathcal{I} \models \varphi$. Thus, $\varphi/A(a) \in \mathcal{U}'$. Moreover, $\mathcal{I} \models \neg A(a)$ since $x \notin A^{\mathcal{I}}$. Since $\mathcal{D}$ is maximal and $\mathcal{I} \models \mathcal{D}$, we know that $\neg A(a) \in \mathcal{D}$, which, together with $\varphi/A(a) \in \mathcal{U}'$, yields $A(a) \in \mathcal{D}_{\mathcal{U}'}$. By 1 of this lemma, we know that $\mathcal{D}_{\mathcal{U}'}$ is consistent, which implies $\neg A(a) \notin \mathcal{D}_{\mathcal{U}'}$. Thus, $a^{\mathcal{I}} \in A^{\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}}$, which is a contradiction.

   "$\supseteq$": Assume that there exists an $x \in \Delta^{\mathcal{I}}$ such that $x \in A^{\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}} \setminus A^{\mathcal{I}^{\mathcal{U}}}$. Then, $x \in A^{\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}}$ implies that

   - $x \in A^{\mathcal{I}}$ and $x \notin \{a^{\mathcal{I}} \mid \neg A(a) \in \mathcal{D}_{\mathcal{U}'}\}$, or

- $x \notin A^{\mathcal{I}}$, $x \in \{a^{\mathcal{I}} \mid A(a) \in \mathcal{D}_{\mathcal{U}'}\}$ and $x \notin \{a^{\mathcal{I}} \mid \neg A(a) \in \mathcal{D}_{\mathcal{U}'}\}$.

In the former case, $x \notin A^{\mathcal{I}^{\mathcal{U}}}$ implies that there exists some $\varphi/\neg A(a) \in \mathcal{U}$ with $x = a^{\mathcal{I}}$ and $\mathcal{I} \models \varphi$. Then, $\varphi/\neg A(a) \in \mathcal{U}'$. Since $\mathcal{D}$ is maximal and $\mathcal{I} \models \mathcal{D}$, $a^{\mathcal{I}} \in A^{\mathcal{I}}$ yields $A(a) \in \mathcal{D}$, which, together with $\varphi/\neg A(a) \in \mathcal{U}'$, implies $\neg A(a) \in \mathcal{D}_{\mathcal{U}'}$. Thus, $a^{\mathcal{I}} \notin A^{\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}}$, which contradicts the assumption. In the latter case, $x \in \{a^{\mathcal{I}} \mid A(a) \in \mathcal{D}_{\mathcal{U}'}\}$ implies that there exists some $A(a) \in \mathcal{D}_{\mathcal{U}'}$ with $a^{\mathcal{I}} = x$. From the definition of $\mathcal{D}_{\mathcal{U}'}$, it follows that there exists some φ such that $\varphi/A(a) \in \mathcal{U}'$, which implies that $\varphi/A(a) \in \mathcal{U}$ and $\mathcal{I} \models \varphi$. Since we consider only consistent updates, $\mathcal{I} \not\models \varphi$ for all $\varphi/\neg A(a)$. Thus, $a^{\mathcal{I}} \in A^{\mathcal{I}^{\mathcal{U}}}$, which is a contradiction.

3. Following from the construction of $\mathcal{D}_{\mathcal{U}'}$, $\mathcal{D}_{\mathcal{U}'}$ is an unconditional update. From Definition 17, we know that $(\mathcal{I}^{\neg\mathcal{D}_{\mathcal{U}'}})^{\mathcal{D}_{\mathcal{U}'}}$ and $\mathcal{I}$ share the domain and the interpretations of all individual names. For every $A \in \mathsf{N_C}$,

$$\begin{aligned} A^{(\mathcal{I}^{\neg\mathcal{D}_{\mathcal{U}'}})^{\mathcal{D}_{\mathcal{U}'}}} &= ((A^{\mathcal{I}} \cup \{a^{\mathcal{I}} \mid A(a) \in \neg\mathcal{D}_{\mathcal{U}'}\}) \setminus \{a^{\mathcal{I}} \mid \neg A(a) \in \neg\mathcal{D}_{\mathcal{U}'}\} \cup \\ &\quad \{a^{\mathcal{I}} \mid A(a) \in \mathcal{D}_{\mathcal{U}'}\}) \setminus \{a^{\mathcal{I}} \mid \neg A(a) \in \mathcal{D}_{\mathcal{U}'}\} \\ &= ((A^{\mathcal{I}} \cup \{a^{\mathcal{I}} \mid \neg A(a) \in \mathcal{D}_{\mathcal{U}'}\}) \setminus \{a^{\mathcal{I}} \mid A(a) \in \mathcal{D}_{\mathcal{U}'}\} \cup \\ &\quad \{a^{\mathcal{I}} \mid A(a) \in \mathcal{D}_{\mathcal{U}'}\}) \setminus \{a^{\mathcal{I}} \mid \neg A(a) \in \mathcal{D}_{\mathcal{U}'}\}. \end{aligned}$$

Since $\mathcal{I} \models \mathcal{D}_{\mathcal{U}'}$, $\{a^{\mathcal{I}} \mid A(a) \in \mathcal{D}_{\mathcal{U}'}\} \subseteq A^{\mathcal{I}}$ and $A^{\mathcal{I}} \cap \{a^{\mathcal{I}} \mid \neg A(a) \in \mathcal{D}_{\mathcal{U}'}\} = \emptyset$. Hence, $A^{(\mathcal{I}^{\neg\mathcal{D}_{\mathcal{U}'}})^{\mathcal{D}_{\mathcal{U}'}}} = A^{\mathcal{I}}$. Analogously, it can be shown that for every $r \in \mathsf{N_R}$, $r^{(\mathcal{I}^{\neg\mathcal{D}_{\mathcal{U}'}})^{\mathcal{D}_{\mathcal{U}'}}} = r^{\mathcal{I}}$. Thus, we obtain that $(\mathcal{I}^{\neg\mathcal{D}_{\mathcal{U}'}})^{\mathcal{D}_{\mathcal{U}'}} = \mathcal{I}$.

4. Assume that $\mathcal{A}^{\mathcal{D}_{\mathcal{U}'}} = \{\bot(a)\}$ for some $a \in \mathsf{N_I}$. By the construction of $\mathcal{A}^{\mathcal{D}_{\mathcal{U}'}}$, either $\mathcal{A} = \{\bot(a)\}$ or there exists some role assertion $\varphi \in \mathcal{A}$ such that $\varphi \dot{\in} \mathcal{D}_{\mathcal{U}'}$. The fact $\mathcal{I} \models \mathcal{A}$ rules out the former case. Since $\varphi \dot{\in} \mathcal{D}_{\mathcal{U}'}$, we know that $\dot{\neg}\varphi$ is in $\mathcal{D}$. Since $\mathcal{I} \models \mathcal{A}$, $\mathcal{I} \models \mathcal{D}$, $\varphi \in \mathcal{A}$, and $\dot{\neg}\varphi \in \mathcal{D}$, we get $\mathcal{I} \models \varphi$ and $\mathcal{I} \models \dot{\neg}\varphi$.

❑

As a direct consequence of 1 of Lemma 33, a diagram $\mathcal{D}$ for $\mathsf{rhs}(\mathcal{U})$ and a subset $\mathcal{U}'$ of $\mathcal{U}$ uniquely determine an unconditional update $\mathcal{D}_{\mathcal{U}'}$ for a given conditional update $\mathcal{U}$. In what follows, we will see how to employ $\mathcal{D}_{\mathcal{U}'}$ to compute updated ABoxes with conditional updates.

Intuitively, a diagram gives a complete description of the part of an interpretation that can be changed by performing an update. Each $\mathcal{U}'$ is a set of possibly violated effects of $\mathcal{U}$ whose preconditions are satisfied in the current state of the world. Given a diagram $\mathcal{D}$ and a subset $\mathcal{U}'$ of $\mathcal{U}$, $\mathcal{D}_{\mathcal{U}'}$ determines all effects of $\mathcal{U}'$ that are violated by interpretations whose relevant part is described by $\mathcal{D}$.

We use $\bigwedge \mathcal{A}$ as an abbreviation for $\bigwedge_{\varphi \in \mathcal{A}} \varphi$. Then, we assemble the logical update $\mathcal{A}'$ of $\mathcal{A}$ with $\mathcal{U}$ as follows:

$$\begin{aligned} \mathcal{A}' = \bigvee_{\mathcal{D} \in \mathfrak{D}} \bigvee_{\mathcal{U}' \subseteq \mathcal{U}} \bigwedge \mathcal{A}^{\mathcal{D}_{\mathcal{U}'}} \cup \mathcal{D}_{\mathcal{U}'} \cup \mathcal{D}^{\mathcal{D}_{\mathcal{U}'}} \cup \\ \{\varphi \mid \varphi/\psi \in \mathcal{U}'\}^{\mathcal{D}_{\mathcal{U}'}} \cup \{\dot{\neg}\varphi \mid \varphi/\psi \in \mathcal{U} \setminus \mathcal{U}'\}^{\mathcal{D}_{\mathcal{U}'}}. \end{aligned} \tag{2}$$

The component $\mathcal{A}^{\mathcal{D}_{\mathcal{U}'}}$ is the update of the original ABox $\mathcal{A}$, $\mathcal{D}_{\mathcal{U}'}$ asserts the effects of $\mathcal{U}'$ that are triggered and violated, and $\mathcal{D}^{\mathcal{D}_{\mathcal{U}'}}$ denotes the update of the diagram $\mathcal{D}$. The precondition of every effect of $\mathcal{U}$ is updated as well since an effect of $\mathcal{U}$ is triggered if and only if its preconditions are satisfied. This implies that the preconditions of triggered effects and the negation of the preconditions of untriggered effects hold before executing the update.

Lemma 34. *Let $\mathcal{A}$ be an $\mathcal{ALCQIO}^{@}$-ABox and $\mathcal{U}$ an update. Let $\mathcal{A}'$ be the $\mathcal{ALCQIO}^{@}$-ABox defined as (2). Then, $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$.*

Proof. We prove this lemma by showing $M(\mathcal{A} * \mathcal{U}) = M(\mathcal{A}')$.

"$\subseteq$": Let $\mathcal{I}$ and $\mathcal{I}'$ be two interpretations such that $\mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$. We have to show that $\mathcal{I}' \models \mathcal{A}'$. For this purpose, it is enough to find a diagram $\mathcal{D}$ and a subset $\mathcal{U}'$ of $\mathcal{U}$ such that $\mathcal{I}'$ is a model of the disjunct of $\mathcal{A}'$ which is determined by $\mathcal{D}_{\mathcal{U}'}$. Consider the following subset $\mathcal{D}$ of $L_{\mathcal{U}}$:

$$\mathcal{D} = \{l \in L_{\mathcal{U}} \mid \mathcal{I} \models l\}.$$

We show that $\mathcal{D}$ is a diagram for $\mathsf{rhs}(\mathcal{U})$:

- $\mathcal{D}$ is consistent since $\mathcal{I} \models \psi$ for all $\psi \in \mathcal{D}$.
- $\mathcal{D}$ is maximal: Assume that there exists a consistent subset $\mathcal{D}'$ of $L_{\mathcal{U}}$ such that $\mathcal{D} \subset \mathcal{D}'$. Then, there is a ψ such that $\psi \in \mathcal{D}'$ and $\psi \notin \mathcal{D}$. By the definition of $\mathcal{D}$, $\psi \notin \mathcal{D}$ implies $\mathcal{I} \not\models \psi$, which yields $\mathcal{I} \models \dot{\neg}\psi$. Hence, $\dot{\neg}\psi \in \mathcal{D}$. Together with $\mathcal{D} \subset \mathcal{D}'$, this implies $\dot{\neg}\psi \in \mathcal{D}'$. Since both ψ and $\dot{\neg}\psi$ are in $\mathcal{D}'$, $\mathcal{D}'$ is inconsistent which contradicts the assumption.

By the definition of $\mathcal{D}$, we have that $\mathcal{I} \models \mathcal{D}$. Consider the following subset $\mathcal{U}'$ of $\mathcal{U}$:

$$\mathcal{U}' = \{\varphi/\psi \in \mathcal{U} \mid \mathcal{I} \models \varphi\}.$$

Thus, we have $\mathcal{I} \models \neg\mathcal{D}_{\mathcal{U}'}$ since $\mathcal{I} \models \mathcal{D}$. It follows from 2 of Lemma 33 that $\mathcal{I}' = \mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}$. Thus, it suffices to show that $\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}$ is a model of the following disjunct $\mathcal{B}_{\mathcal{D}_{\mathcal{U}'}}$ of $\mathcal{A}'$:

$$\begin{aligned} \mathcal{B}_{\mathcal{D}_{\mathcal{U}'}} \quad = \quad & \bigwedge \mathcal{A}^{\mathcal{D}_{\mathcal{U}'}} \cup \mathcal{D}_{\mathcal{U}'} \cup \mathcal{D}^{\mathcal{D}_{\mathcal{U}'}} \cup \\ & \{\varphi \mid \varphi/\psi \in \mathcal{U}'\}^{\mathcal{D}_{\mathcal{U}'}} \cup \{\dot{\neg}\varphi \mid \varphi/\psi \in \mathcal{U} \setminus \mathcal{U}'\}^{\mathcal{D}_{\mathcal{U}'}}. \end{aligned} \tag{3}$$

By the definition of $\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}}$, $\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}} \models \mathcal{D}_{\mathcal{U}'}$. Since $\mathcal{I} \models \neg\mathcal{D}_{\mathcal{U}'}$, $\mathcal{I} \models \mathcal{A}$, and $\mathcal{I} \models \mathcal{D}$, it follows from 4 of Lemma 33 that neither of $\mathcal{A}^{\mathcal{D}_{\mathcal{U}'}}$ and $\mathcal{D}^{\mathcal{D}_{\mathcal{U}'}}$ is $\{\bot(a)\}$ for all $a \in \mathsf{N_I}$. Since $\mathcal{I} \models \neg\mathcal{D}_{\mathcal{U}'}$, $\mathcal{I} \models \mathcal{A}$, and $\mathcal{I} \models \mathcal{D}$, it follows from Lemma 32 that $\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}} \models \mathcal{A}^{\mathcal{D}_{\mathcal{U}'}}$ and $\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}} \models \mathcal{D}^{\mathcal{D}_{\mathcal{U}'}}$. By the definition of $\mathcal{U}'$, we know that for all φ/ψ in $\mathcal{U}$, $\varphi/\psi \in \mathcal{U}'$ implies $\mathcal{I} \models \varphi$ and $\varphi/\psi \notin \mathcal{U}'$ implies $\mathcal{I} \models \dot{\neg}\varphi$. Likewise, it follows from 4 of Lemma 33 that neither of $\{\varphi \mid \varphi/\psi \in \mathcal{U}'\}^{\mathcal{D}_{\mathcal{U}'}}$ and $\{\dot{\neg}\varphi \mid \varphi/\psi \in \mathcal{U} \setminus \mathcal{U}'\}^{\mathcal{D}_{\mathcal{U}'}}$ is $\{\bot(a)\}$ for all $a \in \mathsf{N_I}$. By Lemma 32, $\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}} \models \{\varphi \mid \varphi/\psi \in \mathcal{U}'\}^{\mathcal{D}_{\mathcal{U}'}} \cup \{\dot{\neg}\varphi \mid \varphi/\psi \in \mathcal{U} \setminus \mathcal{U}'\}^{\mathcal{D}_{\mathcal{U}'}}$.

"$\supseteq$": Let $\mathcal{I}' \models \mathcal{A}'$. We need to show that there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$. Since $\mathcal{I}' \models \mathcal{A}'$, there exist a $\mathcal{D} \in \mathfrak{D}$ and a $\mathcal{U}' \subseteq \mathcal{U}$ such that $\mathcal{I}' \models \mathcal{B}_{\mathcal{D}_{\mathcal{U}'}}$, where $\mathcal{B}_{\mathcal{D}_{\mathcal{U}'}}$ is defined as (3).

$$
\begin{aligned}
(\exists r.C)^{\mathcal{U}} &= \Big(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\} \sqcap \exists r.C^{\mathcal{U}}\Big) \sqcup \exists r.\Big(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\} \sqcap C^{\mathcal{U}}\Big) \\
&\quad \sqcup \bigsqcup_{a,b\in\mathsf{Obj}(\mathcal{U}),r(a,b)\notin\mathcal{U}} (\{a\} \sqcap \exists r.(\{b\} \sqcap C^{\mathcal{U}})) \sqcup \bigsqcup_{\neg r(a,b)\in\mathcal{U}} (\{a\} \sqcap @_b C^{\mathcal{U}}) \\
(\forall r.C)^{\mathcal{U}} &= \Big(\Big(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\}\Big) \to \forall r.C^{\mathcal{U}}\Big) \sqcap \forall r.\Big(\Big(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\}\Big) \to C^{\mathcal{U}}\Big) \\
&\quad \sqcap \bigsqcap_{a,b\in\mathsf{Obj}(\mathcal{U}),r(a,b)\notin\mathcal{U}} (\{a\} \to \forall r.(\{b\} \to C^{\mathcal{U}})) \sqcap \bigsqcap_{\neg r(a,b)\in\mathcal{U}} (\{a\} \to @_b C^{\mathcal{U}})
\end{aligned}
$$

Figure 5: Constructing $C^{\mathcal{U}}$ for existential and value restrictions.

Let $\mathcal{I} = (\mathcal{I}')^{\neg\mathcal{D}_{\mathcal{U}'}}$. Then it follows from Definition 17 that $\mathcal{I} \models \neg\mathcal{D}_{\mathcal{U}'}$. Since for each fixed update $\mathcal{U}$, the relation $\Longrightarrow_{\mathcal{U}}$ is functional, $\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}} = ((\mathcal{I}')^{\neg\mathcal{D}_{\mathcal{U}'}})^{\mathcal{D}_{\mathcal{U}'}}$. Since $\mathcal{I}' \models \mathcal{B}_{\mathcal{D}_{\mathcal{U}'}}$, we know that $\mathcal{I}' \models \mathcal{D}_{\mathcal{U}'}$. By 3 of Lemma 33, $\mathcal{I}' \models \mathcal{D}_{\mathcal{U}'}$ implies that $((\mathcal{I}')^{\neg\mathcal{D}_{\mathcal{U}'}})^{\mathcal{D}_{\mathcal{U}'}} = \mathcal{I}'$. Hence, $\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}} = \mathcal{I}'$. Since $\mathcal{I}' \models \mathcal{B}_{\mathcal{D}_{\mathcal{U}'}}$, none of $\mathcal{A}^{\mathcal{D}_{\mathcal{U}'}}$, $\mathcal{D}^{\mathcal{D}_{\mathcal{U}'}}$, $\{\varphi \mid \varphi/\psi \in \mathcal{U}'\}^{\mathcal{D}_{\mathcal{U}'}}$, and $\{\dot{\neg}\varphi \mid \varphi/\psi \in \mathcal{U} \setminus \mathcal{U}'\}^{\mathcal{D}_{\mathcal{U}'}}$ is $\{\bot(a)\}$ for all $a \in \mathsf{N_I}$. By Lemma 32 and since $\mathcal{I}' \models \mathcal{A}^{\mathcal{D}_{\mathcal{U}'}}$ and $\mathcal{I} \models \neg\mathcal{D}_{\mathcal{U}'}$, we obtain that $\mathcal{I} \models \mathcal{A}$. Likewise, since $\mathcal{I}' \models \mathcal{D}^{\mathcal{D}_{\mathcal{U}'}}$ and $\mathcal{I} \models \neg\mathcal{D}_{\mathcal{U}'}$, we obtain that $\mathcal{I} \models \mathcal{D}$.

It remains to show that $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$. To this end, we show that $\mathcal{U}' = \{\varphi/\psi \in \mathcal{U} \mid \mathcal{I} \models \varphi\}$. This, together with $\mathcal{I} \models \mathcal{D}$, $\mathcal{I}^{\mathcal{D}_{\mathcal{U}'}} = \mathcal{I}'$ and $\mathcal{I} \models \neg\mathcal{D}_{\mathcal{U}'}$, yields that $\mathcal{I}' = \mathcal{I}^{\mathcal{U}}$ by 2 of Lemma 33.

Assume that $\mathcal{U}' \neq \{\varphi/\psi \in \mathcal{U} \mid \mathcal{I} \models \varphi\}$. Then

- either there exists some $\varphi/\psi \in \mathcal{U}$ such that $\mathcal{I} \models \varphi$ and $\varphi/\psi \notin \mathcal{U}'$,
- or there exits some $\varphi/\psi \in \mathcal{U}'$ such that $\mathcal{I} \not\models \varphi$.

In the former case, $(\dot{\neg}\varphi)^{\mathcal{D}_{\mathcal{U}'}} \in \mathcal{B}_{\mathcal{D}_{\mathcal{U}'}}$ since $\varphi/\psi \in \mathcal{U}$ and $\varphi/\psi \notin \mathcal{U}'$. Thus, $\mathcal{I}' \models \mathcal{B}_{\mathcal{D}_{\mathcal{U}'}}$ implies $\mathcal{I}' \models (\dot{\neg}\varphi)^{\mathcal{D}_{\mathcal{U}'}}$. By Lemma 32, we get $\mathcal{I} \models \dot{\neg}\varphi$, which contradicts $\mathcal{I} \models \varphi$. In the latter case, $\varphi/\psi \in \mathcal{U}'$ implies $\varphi^{\mathcal{D}_{\mathcal{U}'}} \in \mathcal{B}_{\mathcal{D}_{\mathcal{U}'}}$. $\mathcal{I}' \models \mathcal{B}_{\mathcal{D}_{\mathcal{U}'}}$ implies $\mathcal{I}' \models \varphi^{\mathcal{D}_{\mathcal{U}'}}$. Thus, by Lemma 32, we get $\mathcal{I} \models \varphi$, which is a contradiction. ❑

The Boolean ABox operators are used only as an abbreviation for the "@" constructor. This can be safely done since the translation from Boolean ABoxes to non-Boolean ones described in the proof of Lemma 13 is polynomial.

It is easy to adapt the construction of updated ABoxes to the DLs $\mathcal{ALCO}^{@}$, $\mathcal{ALCIO}^{@}$, $\mathcal{ALCQO}^{@}$. For the former two, we have to treat existential and value restrictions in the $C^{\mathcal{U}}$ translation rather than number restrictions. The corresponding translation is shown in Figure 5. The lemmas proved above for $\mathcal{ALCQIO}^{@}$ are then easily adapted. As a direct consequence of Lemma 34 and the construction of logical updates, we achieve the following theorem:

Theorem 35. *All of the following DLs have logical updates: $\mathcal{ALCO}^{@}$, $\mathcal{ALCIO}^{@}$, $\mathcal{ALCQO}^{@}$, and $\mathcal{ALCQIO}^{@}$.*

Size of Logical Updates

We now analyze the size of constructed logical updates. A close inspection of the ABox $\mathcal{A}'$ computed above reveals the following:

- For a concept C, we use $d(C)$ to denote the maximal nesting depth of qualified number restrictions in C. The size of $C^{\mathcal{D}_{\mathcal{U}'}}$ is exponential in the size of the original ABox $\mathcal{A}$. The exponential blowup is caused by the recursive translation $C^{\mathcal{D}_{\mathcal{U}'}}$ and the duplication of $C^{\mathcal{D}_{\mathcal{U}'}}$ for qualified number restriction, existential and value restriction. The number of recursions is bounded by $d(C)$ and the number of duplications is polynomially bounded by $\#\mathsf{Obj}(\mathcal{D}_{\mathcal{U}'})$. It is clear that $\#\mathsf{Obj}(\mathcal{D}_{\mathcal{U}'}) \leq |\mathcal{D}_{\mathcal{U}'}|$. Thus, we can find a polynomial p such that, for every concept C and every $\mathcal{D}_{\mathcal{U}'}$,
$$|C^{\mathcal{D}_{\mathcal{U}'}}| \leq |C| \times (p(|\mathcal{D}_{\mathcal{U}'}|))^{d(C)}.$$
- The size of $\mathcal{D}$ is linear in the size of $\mathcal{U}$, and the size of $\mathcal{D}_{\mathcal{U}'}$ is linear in the size of $\mathcal{U}$. Thus, we can find a polynomial q such that, for every concept C and every $\mathcal{D}_{\mathcal{U}'}$,
$$|C^{\mathcal{D}_{\mathcal{U}'}}| \leq |C| \times (q(|\mathcal{U}|))^{d(C)}.$$
It follows that the size of $C^{\mathcal{D}_{\mathcal{U}'}}$ is polynomial in the size of the update $\mathcal{U}$.
- The number of disjuncts in $\mathcal{A}'$ depends only on the size of $\mathcal{U}$ and is exponential in $\mathcal{U}$ since there are exponentially many diagrams for $\mathsf{rhs}(\mathcal{U})$ and exponentially many subsets $\mathcal{U}'$ of $\mathcal{U}$. Thus, the size of $\mathcal{A}'$ is exponential both in the size of $\mathcal{A}$ and in the size of $\mathcal{U}$.

These bounds hold independent of whether the numbers inside number restrictions are coded in unary or in binary. Therefore, we obtain the following theorem:

Theorem 36. *Let $\mathcal{L} \in \{\mathcal{ALCO}^{@}, \mathcal{ALCIO}^{@}, \mathcal{ALCQO}^{@}, \mathcal{ALCQIO}^{@}\}$. Then there exist polynomials p_1, p_2, and q such that, for every $\mathcal{L}$-ABox $\mathcal{A}$ and every update $\mathcal{U}$, there exists an $\mathcal{L}$-ABox $\mathcal{A}'$ such that the following hold:*

- $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$;
- $|\mathcal{A}'| \leq 2^{p_1(|\mathcal{A}|)} \cdot 2^{p_2(|\mathcal{U}|)}$;
- $\mathcal{A}'$ *can be computed in time* $q(|\mathcal{A}'|)$.

In Section 3.2, we will show that the exponential blowup cannot be entirely avoided even for unconditional updates unless PTime = NC.

Logical Updates with Unconditional Updates

When we consider unconditional updates, the logical update $\mathcal{A}'$ of the ABox $\mathcal{A}$ with the update $\mathcal{U}$ is
$$\mathcal{A}' = \bigvee_{\mathcal{D} \in \mathfrak{D}} \bigwedge \mathcal{A}^{\mathcal{D}_{\mathcal{U}}} \cup \mathcal{D}_{\mathcal{U}} \cup \mathcal{D}^{\mathcal{D}_{\mathcal{U}}}. \tag{4}$$

This is because we know that all of effects of an unconditional update are always triggered, i.e., $\mathcal{U}' = \mathcal{U}$, and thus we do not need to combine the disjunction of all of the subsets of $\mathcal{U}$. Moreover, since the preconditions are of the form $\top(a)$ and updating $\top(a)$ with any update results in $\top(a)$, updated preconditions are not taken into account. Note that the size of $\mathcal{A}'$ is still exponential both in size of $\mathcal{A}$ and in the size of $\mathcal{U}$.

Logical Updates on Boolean ABoxes

In Section 2.2, Boolean ABoxes were introduced as a generalization of standard ABoxes, and a close connection between Boolean ABoxes and the @ constructor was established. We say that a DL $\mathcal{L}$ has *logical updates on Boolean ABoxes* if, for every Boolean $\mathcal{L}$-ABox $\mathcal{A}$ and update $\mathcal{U}$, there exists a Boolean $\mathcal{L}$-ABox $\mathcal{A}'$ such that $M(\mathcal{A} * \mathcal{U}) = M(\mathcal{A}')$. In fact, by using the arguments of Lemma 13, it is easy to see that the expressive power of Boolean $\mathcal{L}$-ABoxes is identical to the expressive power of non-Boolean $\mathcal{L}^{@}$-ABoxes, for any DL $\mathcal{L}$ in $\{\mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}, \mathcal{ALCQIO}\}$. Hence, Theorem 35 and Theorem 36 can also be understood in terms of Boolean ABoxes.

Due to the generalization of Lemma 13 to the relevant languages, the construction presented in Section 3.1 can also be used to compute logical updates on Boolean ABoxes: first convert the Boolean $\mathcal{L}$-ABox into a non-Boolean $\mathcal{L}^{@}$-ABox, apply the described construction, and then convert the resulting Boolean $\mathcal{L}^{@}$-ABox back into a Boolean $\mathcal{L}$-ABox.

Theorem 37. *All of the following DLs have logical updates on Boolean ABoxes: $\mathcal{ALCO}$, $\mathcal{ALCIO}$, $\mathcal{ALCQO}$, $\mathcal{ALCQIO}$, and their extensions with the @ constructor.*

What is the size of updated Boolean ABoxes computed by the above approach? The main observation is that, while the translation of Boolean $\mathcal{L}$-ABoxes into non-Boolean $\mathcal{L}^{@}$-ABoxes is polynomial, the reverse translation induces an exponential blowup. More precisely, this blowup is exponential in the nesting depth of the @ constructor. Since our translation $C^{\mathcal{D}_{\mathcal{U}'}}$ introduces nestings of the @ constructor whose depth is linear in the size of C, this approach now produces a double exponential blowup both in the size of the original ABox and in the size of the update for every $\mathcal{L}$ in Theorem 37 without the extension with the @ constructor.

Theorem 38. *Let $\mathcal{L} \in \{\mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}, \mathcal{ALCQIO}\}$. Then there exist polynomials p_1, p_2, and q such that, for every Boolean $\mathcal{L}$-ABox $\mathcal{A}$ and every update $\mathcal{U}$, there exists an Boolean $\mathcal{L}$-ABox $\mathcal{A}'$ such that the following hold:*

- $\mathcal{A} * \mathcal{U} = \mathcal{A}'$;
- $|\mathcal{A}'| \leq 2^{2^{p_1(|\mathcal{A}|)}} \cdot 2^{2^{p_2(|\mathcal{U}|)}}$;
- $\mathcal{A}'$ *can be computed in time* $q(|\mathcal{A}'|)$.

Note that for unconditional updates $\mathcal{U}$, the double exponential blowup is only in the size of the original ABox, i.e., $|\mathcal{A}'| \leq 2^{2^{p_1(|\mathcal{A}|)}} \cdot 2^{p_2(|\mathcal{U}|)}$, since we do not need to update the preconditions of $\mathcal{U}$ in this case.

For the DLs $\mathcal{L}^{@}$, with $\mathcal{L}$ as in Theorem 38, we have logical updates on Boolean ABoxes whose size is as described in Theorem 36, i.e., only single exponential blowup is induced: the final conversion step of Boolean $\mathcal{L}^{@}$-ABoxes into Boolean $\mathcal{L}$-ABoxes can simply be omitted. It is still an open question whether the upper bounds given in Theorem 38 can be improved.

In [DLB+09a], a construction of logical updates is introduced directly on Boolean ABoxes[1] and thus the forementioned first conversion step can be ignored. There, the logical update of a Boolean ABox $\mathcal{A}$ is obtained by updating inductively each non-Boolean assertion in $\mathcal{A}$.

Lemma 39 ([DLB+09a])**.** *Update distributes over conjunction and disjunction of Boolean assertions, i.e.,*

$$(\varphi_1 \boxtimes \varphi_2) * \mathcal{U} \equiv (\varphi_1 * \mathcal{U}) \boxtimes (\varphi_2 * \mathcal{U}),$$

where $\boxtimes$ denotes either $\wedge$ or $\vee$, φ_1, φ_2 are Boolean assertions, and $\mathcal{U}$ is an update.

3.2 A Lower Bound for the Size of Logical Updates

In this section, we establish a general lower bound on the size of the updated ABox: even in propositional logic, the logical update of an ABox even with an unconditional update can become exponential in the size of the whole input which consists of the original ABox and the update. At least, this holds unless every PTime algorithm is LogTime-parallelizable, i.e., unless the complexity classes PTime and NC are identical. As discussed by Papadimitriou in [Pap94], this is believed to be similarly unlikely as PTime = NP. This lower bound on the size of updated ABoxes transfers to all DLs considered in this paper. Note that this result complements the one from [CDLS99], where it is shown that an exponential blowup of propositional updates cannot be avoided if *arbitrary formulas* are allowed as updates unless the first levels of the polynomial hierarchy collapses. Our argument uses a much more restricted form of updates (conjunctions of literals) and refers to a different complexity-theoretic assumption.

We start with introducing the notions of concept, ABox, and update, restricted to propositional logic. After that, we introduce the notion of a uniform interpolant of a propositional concept and show that the size of any representation of the logical update is polynomially bounded by the size of the smallest uniform interpolant. Finally, by establishing the relationship between uniform interpolants and Boolean circuits, we prove that if an exponential blowup in the input of logical update could be avoided then for any given Boolean circuit, there would be a polynomially bounded propositional concept which computes the same Boolean function: If such a concept always exists, then PTime = NC.

[1] Techniques of optimizing the construction of logical updates are discussed in [DLB+09a] as well.

For the following definitions, we fix an individual name a. A *propositional ABox* $\mathcal{A}$ is of the form $\{C(a)\}$ with C a *propositional concept*, i.e., a concept that uses only the concept constructors $\neg$, $\sqcap$, and $\sqcup$ over $\mathsf{N_C}$. The top concept ($\top$) and the bottom concept ($\bot$) still stand for tautology and falsity, respectively. A *propositional update* $\mathcal{U}$ contains only assertions of the form $A(a)$ and $\neg A(a)$, where A is a concept name. Observe that propositional ABoxes and propositional updates are only allowed to refer to the single, fixed individual name a.

For the semantics, we fix a single object x. Since we are dealing with propositional ABoxes and updates, we assume that interpretations do not interpret role names, and that interpretation domains have only a single element x with $a^{\mathcal{I}} = x$.

Recall (cf. Definition 26) that $\mathsf{Sig}(C)$ denotes the set of concept names used in a concept C and that for a set S of concept names, $\mathcal{I}_{|\mathsf{S}}$ denotes the reduct of an interpretation $\mathcal{I}$ that interprets only the concept names in S. With this, we introduce the notion of a uniform interpolant:

Definition 40 (S-Uniform Interpolant)**.** Let C be a propositional concept and $\mathsf{S} \subseteq \mathsf{Sig}(C)$. Then a propositional concept D is called a *uniform* S*-interpolant* of C iff

- $\mathsf{Sig}(D) \subseteq \mathsf{S}$, and
- $\{\mathcal{I}_{|\mathsf{S}} \mid x \in C^{\mathcal{I}}\} = \{\mathcal{I}_{|\mathsf{S}} \mid x \in D^{\mathcal{I}}\}$.

△

It is easily seen that, for any propositional concept C and any $\mathsf{S} \subseteq \mathsf{Sig}(C)$, a uniform S-interpolant of C exists [D'A98]. One way to construct it is to use the "truth table" of C and make a disjunction of all reducts of models of C to S. Moreover, if D is a uniform S-interpolant of C and there exists a concept E such that $\mathsf{Sig}(E) \subseteq \mathsf{S}$ and $\{D(a)\} \equiv \{E(a)\}$, then E is also a uniform S-interpolant of C.

In the next lemma, we illustrate how to employ the shortest uniform S-interpolant of the propositional concept C to construct the smallest logical update $\mathcal{A}'$ of the propositional ABox $\{C(a)\}$. We show that the size of any logical update is polynomially bounded by the size of $\mathcal{A}'$.

Lemma 41. *Let $\mathcal{A} = \{C(a)\}$ be a propositional ABox, $\mathcal{U}$ a propositional update, S the set of concept names in C not occurring in $\mathcal{U}$, D the shortest uniform S-interpolant of C, and*

$$\mathcal{A}' = \{a : (D \sqcap \bigsqcap_{A(a) \in \mathcal{U}} A \sqcap \bigsqcap_{\neg A(a) \in \mathcal{U}} \neg A)\}.$$

Then we have the following:

1. $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$;
2. *if $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}''$, then $|\mathcal{A}'| \leq |\mathcal{U}| + |\mathcal{A}''|$.*

Proof.

1. To prove 1, we have to show that $M(\mathcal{A} * \mathcal{U}) = M(\mathcal{A}')$.

   "$\Rightarrow$": Let $\mathcal{I}, \mathcal{I}'$ be interpretations such that $\mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$. We show that $\mathcal{I}' \models \mathcal{A}'$, i.e., $\mathcal{I}' \models \mathcal{U}$ and $\mathcal{I}' \models D(a)$. Since $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$, $\mathcal{I}' \models \mathcal{U}$. Moreover, since $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$ and the concept names in S do not appear in $\mathcal{U}$, we have $\mathcal{I}_{|\mathsf{S}} = \mathcal{I}'_{|\mathsf{S}}$. This, together with $\mathcal{I} \models C(a)$ and the fact that D is the uniform S-interpolant of C, yields that $\mathcal{I}' \models D(a)$ as required.

   "$\Leftarrow$": Let $\mathcal{I}'$ be an interpretation such that $\mathcal{I}' \models \mathcal{A}'$. In particular, $\mathcal{I}' \models D(a)$. Since D is the uniform S-interpolant of C, there is thus an interpretation $\mathcal{I}$ such that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $\mathcal{I}_{|\mathsf{S}} = \mathcal{I}'_{|\mathsf{S}}$. W.l.o.g., we assume that $\mathcal{I}$ and $\mathcal{I}'$ interpret concept names occurring neither in C nor in $\mathcal{U}$ in the same way. We have to show that $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$ and $\mathcal{I} \models \mathcal{A}$. The latter is clear since $a^{\mathcal{I}} \in C^{\mathcal{I}}$. For the former, it is enough to show that for every concept name A,

   (a) $a^{\mathcal{I}} \in A^{\mathcal{I}'} \setminus A^{\mathcal{I}}$ implies $A(a) \in \mathcal{U}$, and
   (b) $a^{\mathcal{I}} \in A^{\mathcal{I}} \setminus A^{\mathcal{I}'}$ implies $\neg A(a) \in \mathcal{U}$.

   For (a), let $a^{\mathcal{I}} \in A^{\mathcal{I}'} \setminus A^{\mathcal{I}}$. As $\mathcal{I}_{|\mathsf{S}} = \mathcal{I}'_{|\mathsf{S}}$, we have $A \notin \mathsf{S}$. Therefore, A appears in $\mathcal{U}$. This can be either in the form $A(a)$ or $\neg A(a)$. As the second yields a contradiction to $a^{\mathcal{I}} \in A^{\mathcal{I}'}$ and $\mathcal{I}' \models \mathcal{A}'$, we are done. Case (b) is symmetric.

2. Now for 2. Suppose that $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}''$. Then $\mathcal{A}' \equiv \mathcal{A}''$ and $\mathcal{A}'' = \{E(a)\}$ for some concept E. Since $\mathcal{A}''$ is a logical update of $\mathcal{A}$ with $\mathcal{U}$, we know that all concept names occurring in E occur in $\mathcal{A} \cup \mathcal{U}$ as well. Now, for all concept names A such that $A(a) \in \mathcal{U}$, replace every occurrence of A in E by $\top$. For $\neg A(a) \in \mathcal{U}$, replace every occurrence of A in E by $\bot$. Denote the resulting concept by E'. Then $\mathcal{A}'' \equiv \{E'(a)\} \cup \mathcal{U}$, which, together with $\mathcal{A}' \equiv \mathcal{A}''$, yields $\mathcal{A}' \equiv \{E'(a)\} \cup \mathcal{U}$. Moreover, as E' and $\mathcal{U}$ do not have any concept names in common and $\mathcal{A}' \equiv \{E'(a)\} \cup \mathcal{U}$, we have $\{D(a)\} \equiv \{E'(a)\}$. It follows that E' is a S-interpolant of C. We derive $|D| \leq |E'|$ because D is the shortest S-interpolant of C. But then

$$|\mathcal{A}'| \leq |D| + |\mathcal{U}| + 1 \leq |E'| + |\mathcal{U}| + 1 \leq |E| + |\mathcal{U}| + 1 \leq |\mathcal{A}''| + |\mathcal{U}|.$$

❑

The size of uniform interpolants of propositional concepts is closely related to the relative succinctness of propositional concepts and Boolean circuits [Pap94].

Definition 42 (Boolean Circuit)**.** A *Boolean circuit* is a graph $c = (v, e)$ such that

- $v = \{1, \ldots, n\}$;
- there is no circle in c;
- all nodes in the graph have indegree (number of incoming edges) equal to 0, 1, or 2, and each node having indegree 0 (respectively 1, 2) is labeled with an element in $\{\top, \bot, X_1, \ldots, X_m\}$ (respectively $\{\neg\}$, $\{\sqcap, \sqcup\}$) and only the node n has outdegree (number of outgoing edges) equal to 0.

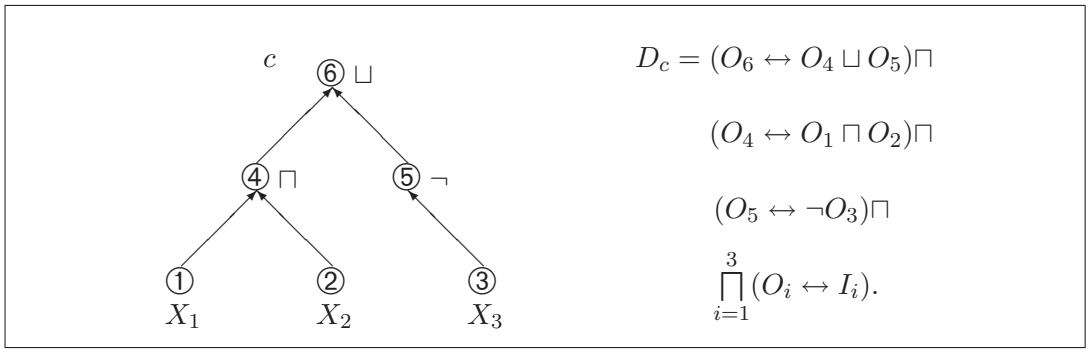


Figure 6: An example of Boolean Circuits.

Every node of c is called a *gate* of c. Gates with indegree 0 are called the *input gates* of c. The node n is called the *output gate* of c. The *size* of c, denoted with $|c|$, is the number of nodes in c. △

As shown in [Pap94], both propositional concepts and Boolean circuits can be used to compute Boolean functions. For example, the circuit c depicted in Figure 6 can be thought of as a representation of the propositional concept $(X_1 \sqcap X_2) \sqcup \neg X_3$. It is known that, unless PTIME = NC, there exists no polynomial p such that every Boolean circuit c can be converted into a propositional concept E_c that computes the same Boolean function as c and satisfies $|E_c| \leq p(|c|)$, see e.g., Exercise 15.5.4 of [Pap94]. In the following, we show that non-existence of such a polynomial p implies that an exponential blowup of the logical update in the size of the whole input cannot be avoided.

Theorem 43. *Unless* PTIME = NC*, there exists no polynomial p such that, for all propositional ABoxes $\mathcal{A}$ and propositional updates $\mathcal{U}$, there exists a propositional ABox $\mathcal{A}'$ such that*

- $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$ *and*
- $|\mathcal{A}'| \leq p(|\mathcal{A}| + |\mathcal{U}|)$.

Proof. Assume that such a p exists. Take a Boolean circuit c with k inputs. Then c can be translated into a propositional concept D_c by introducing concept names $I_1, \ldots, I_k$ for the inputs and, additionally, one auxiliary concept name for the output of every gate. D_c can be constructed with the conjunction of concepts determined by the labels of the nodes, like e.g., the concept D_c in Figure 6:

- If a node i is labeled with an element in $\{\top, \bot, X_1, \ldots, X_m\}$, then i is an input gate of c. Thus, there is a conjunct $I_i \leftrightarrow O_i$ in D_c;
- If a node i is labeled with $\neg$, then there is a conjunct $\neg O_j \leftrightarrow O_i$ in D_c, where O_j is the concept name corresponding to the output of gate j that has an outgoing edge to gate i;

- If a node i is labeled with $\sqcap$ ($\sqcup$, respectively), then there is a conjunct $O_{j_1} \sqcap O_{j_2} \leftrightarrow O_i$ ($O_{j_1} \sqcup O_{j_2} \leftrightarrow O_i$, respectively) in D_c, where O_{j_1} and O_{j_2} are the concept names respectively corresponding to the output of gate j_1 and gate j_2 both of which have an outgoing edge to gate i;

Let G be the set of concept names introduced for gate outputs, and let $O_n \in \mathsf{G}$ be the concept name for the output of the gate computing the final output of c. It follows from the construction of D_c that there exists a polynomial q such that, for all Boolean circuits c,

1. $|D_c| \leq q(|c|)$ and
2. for all interpretations $\mathcal{I}$, $x \in O_n^{\mathcal{I}}$ iff c outputs "$\top$" on input $b_1, \ldots, b_k$, where for all $j \in \{1, \ldots, k\}$, $b_j = \top$ if $x \in I_j^{\mathcal{I}}$ and $b_j = \bot$ otherwise.

Now, set $\mathsf{S} = (\mathsf{Sig}(D_c) \setminus \mathsf{G}) \cup \{O_n\}$. Then the shortest uniform S-interpolant E_c of D_c also satisfies Point 2. Thus, E_c is a (notational variant of a) propositional concept computing the same Boolean function as c.

Consider the ABox $\mathcal{A} = \{D_c(a)\}$ and any update $\mathcal{U}$ such that the set of concept names occurring in $\mathcal{U}$ is $\mathsf{G} \setminus \{O_n\}$ and the size of $\mathcal{U}$ is polynomial in the size of $\mathcal{A}$.[2] Thus, the smallest logical update $\mathcal{A}'$ of $\mathcal{A}$ with $\mathcal{U}$ defined in Lemma 41 is

$$\mathcal{A}' = \{a : (E_c \sqcap \bigsqcap_{A(a) \in \mathcal{U}} A \sqcap \bigsqcap_{\neg A(a) \in \mathcal{U}} \neg A)\}.$$

By the assumption, we know that $|\mathcal{A}'| \leq p(|\mathcal{A}| + |\mathcal{U}|)$, which, together with $|E_c| \leq |\mathcal{A}'|$, implies $|E_c| \leq p(|\mathcal{A}| + |\mathcal{U}|)$. Thus, there is a polynomial p' such that $|E_c| \leq p'(|c|)$ since $|E_c| \leq p(|\mathcal{A}| + |\mathcal{U}|)$, $\mathcal{A} = \{D_c(a)\}$, $|D_c| \leq q(|c|)$, and the size of $\mathcal{U}$ is polynomial in the size of $\mathcal{A}$. However, it is known that this would not happen unless PTIME = NC.

❑

Theorem 43 carries over to all DLs considered in this paper. In the terminology of Cadoli et al. [CDLS99], this result states that the common update operators for propositional theories are not logically compactable even for updates with conjunctions of literals (unless PTIME = NC).

An exponential blowup cannot be entirely avoided unless PTIME = NC. However, we should pay attention to whether the blowup occurs in the size of the original ABox $\mathcal{A}$ or in the size of the update $\mathcal{U}$. As the update will usually be rather small compared to the original ABox, an exponential blowup in the size of $\mathcal{U}$ is much more acceptable than an exponential blowup in the size of $\mathcal{A}$. The algorithm given in Section 3.1 produces an exponential in *both* $|\mathcal{A}|$ and $|\mathcal{U}|$. In the case of propositional logic, Winslett [Win90] gives an algorithm that blows up exponentially only in the size of $\mathcal{U}$, but not in the size of $\mathcal{A}$. We believe that, for the languages mentioned in Theorem 36, the exponential blowup in $|\mathcal{A}|$ can *not* be avoided in general, while the proof are left as an open problem. In Section 3.3 we exhibit two ways around the exponential blowup in the size of $\mathcal{A}$.

[2] Such an update $\mathcal{U}$ always exists, e.g., $\mathcal{U} = \{A(a) \mid A \in \mathsf{G} \setminus \{O_n\}\}$.

3.3 Smaller Logical Updates

The size of logical updates computed in Section 3.1 is exponential in the size of the original ABox. In this section, we explore different ways such that it becomes possible to compute logical updates that are only polynomial in the size of the original ABox (but still exponential in the size of the update).

The first, rather restrictive solution is to admit only concept assertions as effects of updates. As a result, for any concept C, every role remains intact when $C^{\mathcal{D}_{\mathcal{U}'}}$ is constructed since updates do not have effects on role names at all. Then, in all DLs captured by Theorem 35, computing the concepts $C^{\mathcal{D}_{\mathcal{U}'}}$ becomes a lot simpler: just replace every concept name A in C with

$$(A \sqcup \bigsqcup_{\neg A(a) \in \mathcal{D}_{\mathcal{U}'}} \{a\}) \sqcap \neg(\bigsqcup_{A(a) \in \mathcal{D}_{\mathcal{U}'}} \{a\}).$$

If modified in this way, the construction in Lemma 34 yields updated ABoxes that are only polynomial in the size of the original ABox (but still exponential in $|\mathcal{U}|$). The bound is independent of the coding of numbers.

The second solution is to consider computing logical updates in a more expressive DL. Intuitively, updates with only concept assertions do not lead to an exponential blowup because we have available the Boolean constructors on concepts, nominals, and the @ constructor. In standard DLs, none of these operators is available for roles: we can neither construct the union of roles, nor their complement, nor a "nominal role" $\{(a, b)\}$ with a and b individual names. In this section, we investigate updated ABoxes in a language in which such constructors are available. The language we consider is closely related to those languages introduced and investigated in [Bor96, LS01, LSW01], and is of almost the same expressive power as C^2, the two-variable fragment of first-order logic with counting quantifiers [GOR97].

Denote by $\mathcal{ALCQIO}^+$ the DL extending $\mathcal{ALCQIO}^@$ by means of the role constructors $\cap$ (role intersection), $\cup$ (role union), $\neg$ (negated roles), and $\{(a, b)\}$ (nominal roles). In this DL, complex roles are constructed starting from role names and nominal roles, and then applying $\cap$, $\cup$, $\neg$, and the inverse role constructor $\cdot^-$. The definition of the size of a role r is extended in a straightforward way:

- $|r| = 2$ if $r = \{(a, b)\}$ for some $a, b \in \mathsf{N_I}$;
- $|r| = |s| + 1$ if $r = s^-$ or $r = \neg s$;
- $|r| = |r_1| + |r_2| + 1$ if $r = r_1 \cap r_2$ or $r = r_1 \cup r_2$.

The interpretation of complex roles is as expected:

- $\{(a, b)\}^{\mathcal{I}} = \{(a^{\mathcal{I}}, b^{\mathcal{I}})\}$, for all $a, b \in \mathsf{N_I}$;
- $(r_1 \cap r_2)^{\mathcal{I}} = r_1^{\mathcal{I}} \cap r_2^{\mathcal{I}}$ and $(r_1 \cup r_2)^{\mathcal{I}} = r_1^{\mathcal{I}} \cup r_2^{\mathcal{I}}$;
- $(\neg r)^{\mathcal{I}} = (\Delta^{\mathcal{I}})^2 \setminus r^{\mathcal{I}}$.

We note that reasoning in $\mathcal{ALCQIO}^+$ is decidable: this DL can easily be embedded into C^2 and, therefore, ABox consistency is decidable in NExpTime even if the numbers inside number restrictions are coded in binary [GOR97, PST00, PH05]. This bound is tight as, already in $\mathcal{ALCQIO}$, reasoning is NExpTime-hard [Tob00]. We are now ready to show the following result which is independent of the coding of numbers in qualified number restrictions:

Theorem 44. *There exist polynomials p_1, p_2, and q such that, for every $\mathcal{ALCQIO}^+$-ABox $\mathcal{A}$ and every update $\mathcal{U}$, there is an $\mathcal{ALCQIO}^+$-ABox $\mathcal{A}'$ such that*

- $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$;
- $|\mathcal{A}'| \leq p_1(|\mathcal{A}|) \cdot 2^{p_2(|\mathcal{U}|)}$;
- $\mathcal{A}'$ *can be computed in time* $q(|\mathcal{A}'|)$.

Proof. We modify the proof of Theorem 36. For $\mathcal{ALCQIO}^+$, the construction of the concepts $C^{\mathcal{D}_{\mathcal{U}'}}$ is much simpler: it suffices to replace every concept name A in C with

$$(A \sqcup \bigsqcup_{\neg A(a) \in \mathcal{D}_{\mathcal{U}'}} \{a\}) \sqcap \neg(\bigsqcup_{A(a) \in \mathcal{D}_{\mathcal{U}'}} \{a\})$$

and every role name r in C with

$$(r \cup \bigcup_{\neg r(a,b) \in \mathcal{D}_{\mathcal{U}'}} \{(a,b)\}) \cap \neg(\bigcup_{r(a,b) \in \mathcal{D}_{\mathcal{U}'}} \{(a,b)\}).$$

The concepts $C^{\mathcal{D}_{\mathcal{U}'}}$ are therefore of size polynomial in the size of C. The ABox $\mathcal{A}'$ can then be constructed in the same way as in the proof of Theorem 36 and is polynomial in the size of $\mathcal{A}$, but exponential in the size of the update $\mathcal{U}$. ❏

The asymmetry w.r.t. concept and role constructors available in standard DLs leads to a more complicated construction of updated ABoxes. Computing logical updates in $\mathcal{ALCQIO}^+$ gives smaller logical updates. However, there exist no state-of-the-art DL reasoners which support $\mathcal{ALCQIO}^+$.[3] In principle, reasoning in such DLs can also be done with a first-order logic theorem prover. In [DLB+09a], some experimental results for computing logical updates in DLs with and without additional role constructors are presented.

An alternative to working with a DL such as $\mathcal{ALCQIO}^+$, is to work directly in C^2, the two-variable fragment with counting. Then, a result analogous to Theorem 44 is easily obtained.

3.4 Iterated Logical Updates

There are applications in which the domain of interest evolves continuously. In such an environment, it is necessary to update an ABox over and over again. It is clearly important that the exponential blowups of the individual updates do not add up. The following theorem shows that this is indeed not the case. It holds independently of the coding of numbers in qualified number restriction.

[3] The DL reasoner MetTel supports $\mathcal{ALCIO}^+$ [ST07].

Theorem 45. *There exist polynomials p_1, p_2 such that the following holds: for all ABoxes $\mathcal{A}_0, \ldots, \mathcal{A}_n$ and conditional updates $\mathcal{U}_1, \ldots, \mathcal{U}_n$, if $\mathcal{A}_i$ is the ABox computed by our algorithm when $\mathcal{A}_{i-1}$ is updated with $\mathcal{U}_i$, for $0 < i \leq n$, then*

$$|\mathcal{A}_n| \leq 2^{p_1(|\mathcal{A}_0|)} \cdot 2^{p_2(|\mathcal{U}_1|+\cdots+|\mathcal{U}_n|)}.$$

Proof. As argued in Section 3.1, for a concept C, when $C^{\mathcal{D}_{\mathcal{U}'}}$ is constructed, its size has the following upper bound:

$$|C^{\mathcal{D}_{\mathcal{U}'}}| \leq |C| \times (q(|\mathcal{U}|))^{d(C)},$$

where q is a polynomial and $d(C)$ is the maximal nesting depth of qualified number restrictions in C.

The crucial observation now is that, for every concept C and every $\mathcal{D}_{\mathcal{U}'}$, $d(C) = d(C^{\mathcal{D}_{\mathcal{U}'}})$. The maximal nesting depth of qualified number restrictions does not increase when forming $C^{\mathcal{D}_{\mathcal{U}'}}$. It follows that there exists a polynomial q' such that for every concept C and sequence of unconditional updates $\mathcal{U}'_1, \ldots, \mathcal{U}'_i$,

$$|(C^{\mathcal{U}'_1})^{\mathcal{U}'_2 \cdots \mathcal{U}'_i}| \leq |C| \times (q'(|\mathcal{U}'_1| + \cdots + |\mathcal{U}'_i|))^{d(C)}.$$

Note that for every conditional update $\mathcal{U}$, every diagram $\mathcal{D}$ of $\mathsf{rhs}(\mathcal{U})$, and every $\mathcal{U}' \subseteq \mathcal{U}$, $\mathcal{D}_{\mathcal{U}'}$ is linear in the size of $\mathcal{U}$. There are exponentially many disjuncts in the size of $\mathcal{U}$ when the logical update of an ABox $\mathcal{A}$ with $\mathcal{U}$ is constructed.

A close inspection of the construction of $\mathcal{A}_{i+1}$ from $\mathcal{A}_i$ using the concepts $(C^{\mathcal{U}'_1})^{\mathcal{U}'_2 \cdots \mathcal{U}'_i}$ shows that there exists an additional polynomial p such that, for all i,

$$\begin{aligned} |\mathcal{A}_{i+1}| \quad \leq \quad & 2^{p(|\mathcal{U}_1|+\cdots+|\mathcal{U}_i|)} \times \sum_{a:C \in \mathcal{A}_0} (|C| \times (q'(|\mathcal{U}_1| + \cdots + |\mathcal{U}_i|))^{d(C)}) + \\ & \sum_{j=1}^{i} \left(2^{p(|\mathcal{U}_j|+\cdots+|\mathcal{U}_i|)} \times \sum_{a:C/\psi \in \mathcal{U}_j} (|C| \times (q'(|\mathcal{U}_j| + \cdots + |\mathcal{U}_i|))^{d(C)})\right) \end{aligned}$$

The first part yields the upper bound on the size of assertions obtained by updating the assertions from $\mathcal{A}_0$ while the second part provides the upper bound on the size of assertions obtained by updating the assertions which are preconditions in $\mathcal{U}_j$. Note that the preconditions in $\mathcal{U}_j$ are only updated by the sequence of updates $\mathcal{U}_j, \ldots, \mathcal{U}_i$.

The upper bound claimed in the theorem follows immediately from the above inequation. ❑

Note that for an ABox $\mathcal{A}_1$ with the size $2^{p_1(|\mathcal{A}_0|)} \cdot 2^{p_2(|\mathcal{U}_1|)}$ for some ABox $\mathcal{A}_0$ and some update $\mathcal{U}_1$, the size of the logical update $\mathcal{A}_2$ of $\mathcal{A}_1$ with $\mathcal{U}_2$ is

$$2^{p_1(2^{p_1(|\mathcal{A}_0|)} \cdot 2^{p_2(|\mathcal{U}_1|)})} \cdot 2^{p_2(|\mathcal{U}_2|)}$$

in the worst case, which means that exponential blowups could add up. However, the above proof shows if $\mathcal{A}_1$ is the updated ABox then the worst case does not happen since for the updated ABox $\mathcal{A}_1$ computed in Section 3.1, the nesting depth of qualified number restrictions is polynomially bounded by $|\mathcal{A}_0|$.

If we consider only unconditional updates, alternatively, the updated ABox $\mathcal{A}_n$ can be constructed by updating $\mathcal{A}_0$ with a single update $\mathcal{U}$ which is determined by $\mathcal{U}_1, \ldots, \mathcal{U}_n$:

$$\mathcal{U} = (\ldots(\mathcal{U}_1 \setminus \neg\mathcal{U}_2) \cup \mathcal{U}_2 \cdots \setminus \neg\mathcal{U}_n) \cup \mathcal{U}_n. \tag{5}$$

It is not hard to see that for all $\mathcal{I}$ and $\mathcal{I}'$, if $\mathcal{I} \Longrightarrow_{\mathcal{U}_1 \cdots \mathcal{U}_n} \mathcal{I}'$ then $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$ and vice versa. Thus, $(\ldots(\mathcal{A} * \mathcal{U}_1) \ldots * \mathcal{U}_n) \equiv \mathcal{A} * \mathcal{U}$. If we compute $\mathcal{A}_n$ in this way, we need to store the original ABox $\mathcal{A}_0$ and the history of updates which have been used to update $\mathcal{A}_0$ so far. It is easy to see that $|\mathcal{U}| \leq |\mathcal{U}_1| + \cdots + |\mathcal{U}_n|$. It follows from Theorem 36 that the size of $\mathcal{A}_n$ is bounded by $2^{p_1(|\mathcal{A}_0|)} \cdot 2^{p_2(|\mathcal{U}_1|+\cdots+|\mathcal{U}_n|)}$ which coincides with the one in Theorem 45.

For the ways proposed in Section 3.3 to achieve smaller logical updates, the size of the updated ABox is still polynomially bounded by the size of the original ABox when we consider iterated updates. Moreover, the exponential blowups caused by individual updates do not add up either, which can be shown similarly to Theorem 45.

Chapter 4

Projective Updates

The logical updates constructed in Chapter 3 are of exponential size in both the original ABox and the update. The aim of the current chapter is to show that one can construct updates which are of polynomial size in the original ABox and the update when considering projective instead of logical updates. We will show that this holds for all DLs between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}^{@}$. In Section 4.1, such a polynomial construction of projective updates is presented for a given ABox $\mathcal{A}$ and an update $\mathcal{U}$. We show in Section 4.2 that the direct application of this construction to updating an ABox $\mathcal{A}$ iteratively leads to an exponential blowup in the size of $\mathcal{A}$. It is left as an open problem if this exponential blowup can be entirely avoided. Instead of computing projective updates iteratively, we exhibit a direct construction of a projective update of the original ABox with a finite sequence of updates. The updated ABox constructed in this way is bounded polynomially in the size of the input.

4.1 Computing Projective Updates in $\mathcal{ALCQIO}^{@}$

The central idea of achieving a more succinct construction of projective updates, compared to logical updates, is to introduce new (concept or role) names to describe the corresponding concepts and roles before the application of the update, where new means that the names occur neither in the original ABox nor in the update. First, we introduce some notions. Consider an ABox $\mathcal{A}$ and an update $\mathcal{U}$. Let $\mathsf{Sub}(\mathcal{U})$ be the set defined as follows:

$$\mathsf{Sub}(\mathcal{U}) = \bigcup_{\varphi/\psi \in \mathcal{U}} \mathsf{Sub}(\{\varphi, \psi\}).$$

We say that a concept name or a role name is *flexible in* $\mathcal{U}$ if it occurs in the effects of $\mathcal{U}$. We define the set $\mathsf{C}_{\mathsf{fle}}$ as follows:

$$\mathsf{C}_{\mathsf{fle}} = \{C \mid C \in \mathsf{Sub}(\mathcal{A}) \cup \mathsf{Sub}(\mathcal{U}) \text{ and } C \text{ contains a flexible name in } \mathcal{U}\}.$$

The set $\mathsf{C}_{\mathsf{fle}}$ contains all of the subconcepts in the input whose interpretations can be changed by $\mathcal{U}$. We use $\mathsf{R}_{\mathsf{fle}}$ to denote the set of flexible role names in $\mathcal{U}$. We introduce a new concept name $\mathsf{A}_C^{(0)}$ for each $C \in \mathsf{C}_{\mathsf{fle}}$ and a new role name $\mathsf{s}_r^{(0)}$ for each role name $r \in \mathsf{R}_{\mathsf{fle}}$. Intuitively, those new concept names (role names, respectively) are

used to denote the corresponding concepts (role names, respectively) before executing the update. More specifically,

- for every concept $C \in \mathsf{C_{fle}}$, $\mathsf{A}_C^{(0)}$ shall represent the interpretation of C before updating, and
- for every role name $r \in \mathsf{R_{fle}}$, $\mathsf{s}_r^{(0)}$ shall represent the interpretation of r before updating, but only with respect to named objects.

For a concept C and a role name r, we respectively define $C^{(0)}$ and $r^{(0)}$ as follows:

$$C^{(0)} = \begin{cases} \mathsf{A}_C^{(0)} & \text{if } C \in \mathsf{C_{fle}} \\ C & \text{otherwise} \end{cases} \qquad r^{(0)} = \begin{cases} \mathsf{s}_r^{(0)} & \text{if } r \in \mathsf{R_{fle}} \\ r & \text{otherwise} \end{cases}$$

We define $(r^-)^{(0)} = (r^{(0)})^-$ for a role name r. $\mathsf{A}_C^{(0)}$ and $\mathsf{s}_r^{(0)}$ define only on the concepts and role names containing flexible names while the function $\cdot^{(0)}$ defines on arbitrary concepts and roles. For example, for a concept $C \notin \mathsf{C_{fle}}$, the interpretation of C is the same before and after the application of the update since C does not contain any flexible name.

Let $\mathsf{rhs}(\mathcal{U})$ be the set of all effects of the update $\mathcal{U}$. We use $\mathsf{Obj}(\mathcal{U})$ to denote the set of all individual names occurring in $\mathsf{rhs}(\mathcal{U})$. Let φ be an assertion. We use $\varphi^{(0)}$ to denote the assertion defined as follows:

$$\varphi^{(0)} = \begin{cases} C^{(0)}(a) & \text{if } \varphi = C(a) \\ (\neg)r^{(0)}(a,b) & \text{if } \varphi = (\neg)r(a,b) \wedge \{a,b\} \subseteq \mathsf{Obj}(\mathcal{U}) \\ (\neg)r(a,b) & \text{if } \varphi = (\neg)r(a,b) \wedge \{a,b\} \not\subseteq \mathsf{Obj}(\mathcal{U}) \end{cases}$$

We define a projective update $\mathcal{A}'$ which is a union of a number of ABoxes. The following ABox $\mathcal{A}_{\mathsf{ini}}$ simulates the ABox $\mathcal{A}$ using the new concept names $\mathsf{A}_C^{(0)}$ and roles names $\mathsf{s}_r^{(0)}$ whenever the interpretation of C and r is possibly affected by the update $\mathcal{U}$:

$$\mathcal{A}_{\mathsf{ini}} = \{\varphi^{(0)} \mid \varphi \in \mathcal{A}\}.$$

The following ABox states when the interpretation of roles before and after the update definitely remains the same:

$$\mathcal{A}_{\mathsf{r}} = \{(\exists \mathsf{s}_r^{(0)}.\{b\} \leftrightarrow \exists r.\{b\})(a) \mid \{r(a,b), \neg r(a,b)\} \cap \mathsf{rhs}(\mathcal{U}) = \emptyset \wedge a, b \in \mathsf{Obj}(\mathcal{U}) \wedge r \in \mathsf{R_{fle}}\}.$$

Let Obj be the set of all individual names occurring in $\mathcal{A}$ or $\mathcal{U}$. Now we choose a new individual name a^* and a new role u and define the ABox $\mathcal{A}_{\mathsf{aux}}$ as follows:

$$\mathcal{A}_{\mathsf{aux}} = \{u(a^*, b) \mid b \in \mathsf{Obj}\}.$$

As stated in $\mathcal{A}_{\mathsf{aux}}$, the role name u connects the individual name a^* to all individual names occurring in the input. Let $\mathsf{p}(\varphi)$ be the abbreviation for an assertion φ defined as follows:

$$\mathsf{p}(\varphi) = \begin{cases} \exists u.(\{a\} \sqcap C) & \text{if } \varphi = C(a) \\ \exists u.(\{a\} \sqcap \exists r.\{b\}) & \text{if } \varphi = r(a,b) \\ \exists u.(\{a\} \sqcap \forall r.\neg\{b\}) & \text{if } \varphi = \neg r(a,b) \end{cases}$$

It follows from the definition of $\mathcal{A}_{\mathsf{aux}}$ that in every model of $\mathcal{A}_{\mathsf{aux}}$, $a^* : \mathsf{p}(\varphi)$ holds iff φ holds. With the help of $\mathsf{p}(\varphi)$, the ABox $\mathcal{A}_{\mathcal{U}}$ states that for all $\varphi/\psi \in \mathcal{U}$, if φ holds *before* then ψ holds *after* applying $\mathcal{U}$:

$$\mathcal{A}_{\mathcal{U}} = \{a^* : \bigsqcap_{\varphi/\psi \in \mathcal{U}} (\mathsf{p}(\varphi^{(0)}) \rightarrow \mathsf{p}(\psi))\}.$$

It is also necessary to ensure that untriggered effects do not make any changes. To this end, $\mathcal{A}_{\overline{\mathcal{U}}}$ is defined as the following ABox:

$$\mathcal{A}_{\overline{\mathcal{U}}} = \{a^* : \bigsqcap_{\psi \in \mathsf{rhs}(\mathcal{U})} \big((\bigsqcap_{\varphi/\psi \in \mathcal{U}} \neg \mathsf{p}(\varphi^{(0)})) \rightarrow (\mathsf{p}(\psi^{(0)}) \leftrightarrow \mathsf{p}(\psi))\big)\}.$$

Finally, we relate the interpretation of the new concept names $\mathsf{A}_C^{(0)}$ to the interpretation of the concepts C. Here we give the intuitions for two cases: C is a concept name or an at-least number restriction. All other cases can be understood in a similar way.

- For a concept name A, $\{a \mid A(a) \in \mathsf{rhs}(\mathcal{U}) \vee \neg A(a) \in \mathsf{rhs}(\mathcal{U})\}$ is the set of all individual names on which $\mathcal{U}$ can make changes about A. As a result, the interpretation of A will remains the same on the objects of the domain that are not assigned to any individual name in this set.

- For an at-least number restriction $(\geqslant n\ r\ C)$, the role r before applying the update is represented by different roles depending on the objects connected by r: if both of the objects are named, then it is represented by $r^{(0)}$; by r otherwise. Thus, the objects in $(\geqslant n\ r\ C)$ are divided into two parts: anonymous objects and named objects. For the latter ones, their role successors are divided accordingly as well.

The concept C_{bi} is a conjunction over all concepts from Figure 7, where the left hand side ranges over $\mathsf{C}_{\mathsf{fle}}$. We want to state that the concept C_{bi} holds on all relevant objects of the domain, i.e., the objects on which the changes of interpretation caused by an update may affect the interpretation of concepts in $\mathsf{C}_{\mathsf{fle}}$. To achieve this, we associate each concept C with a set P_C of words $r_1 \cdots r_n \in \mathsf{N_R}^*$ inductively defined as follows:

$$\begin{aligned}
\mathsf{P}_C &= \{\epsilon\} \text{ if } C \text{ is } \top, \bot, A \text{ or } \{a\}, \\
&\qquad \text{for some } A \in \mathsf{N_C} \text{ and some } a \in \mathsf{N_I} \\
\mathsf{P}_{C_1 \sqcap C_2} &= \mathsf{P}_{C_1} \cup \mathsf{P}_{C_2} \\
\mathsf{P}_{C_1 \sqcup C_2} &= \mathsf{P}_{C_1} \cup \mathsf{P}_{C_2} \\
\mathsf{P}_{\neg C} &= \mathsf{P}_C \\
\mathsf{P}_{@_a C} &= \mathsf{P}_C \\
\mathsf{P}_{(\geqslant m\ r\ C)} &= \{rw \mid w \in \mathsf{P}_C\} \cup \{\epsilon\} \\
\mathsf{P}_{(\leqslant m\ r\ C)} &= \{rw \mid w \in \mathsf{P}_C\} \cup \{\epsilon\}
\end{aligned}$$

$$\big(\mathsf{A}^{(0)}_{A} \sqcap \neg(\bigsqcup_{\{A(a),\neg A(a)\}\cap \mathsf{rhs}(\mathcal{U})\neq\emptyset} \{a\})\big) \leftrightarrow \big(A \sqcap \neg(\bigsqcup_{\{A(a),\neg A(a)\}\cap \mathsf{rhs}(\mathcal{U})\neq\emptyset} \{a\})\big),\ \text{if } A \in \mathsf{N_C}$$

$$\mathsf{A}^{(0)}_{\{a\}} \leftrightarrow \{a\} \qquad \mathsf{A}^{(0)}_{\top} \leftrightarrow \top \qquad \mathsf{A}^{(0)}_{\bot} \leftrightarrow \bot$$

$$\mathsf{A}^{(0)}_{@_a C} \leftrightarrow @_a \mathsf{A}^{(0)}_{C} \qquad \mathsf{A}^{(0)}_{\neg C} \leftrightarrow \neg \mathsf{A}^{(0)}_{C}$$

$$\mathsf{A}^{(0)}_{C\sqcap D} \leftrightarrow C^{(0)} \sqcap D^{(0)} \qquad \mathsf{A}^{(0)}_{C\sqcup D} \leftrightarrow C^{(0)} \sqcup D^{(0)}$$

$$\begin{aligned}\mathsf{A}^{(0)}_{(\geqslant n\ r\ C)} \leftrightarrow\ & \Big(\big(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\}\big) \sqcap (\geqslant n\ r\ C^{(0)})\Big) \sqcup \\ & \Big(\big(\bigsqcup_{a\in\mathsf{Obj}(\mathcal{U})} \{a\}\big) \sqcap \bigsqcup_{\substack{n_1+n_2=n\\ n_2\leq \#\mathsf{Obj}(\mathcal{U})}} \Big((\geqslant n_1\ r\ \big(\big(\bigsqcap_{b\in\mathsf{Obj}(\mathcal{U})} \neg\{b\}\big) \sqcap C^{(0)}\big)\Big) \\ & \qquad \sqcap (\geqslant n_2\ r^{(0)}\ \big(\big(\bigsqcup_{b\in\mathsf{Obj}(\mathcal{U})} \{b\}\big) \sqcap C^{(0)}\big))\Big)\Big)\end{aligned}$$

$$\begin{aligned}\mathsf{A}^{(0)}_{(\leqslant n\ r\ C)} \leftrightarrow\ & \Big(\big(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\}\big) \rightarrow (\leqslant r\ C^{(0)}\ )\Big) \sqcap \\ & \Big(\big(\bigsqcup_{a\in\mathsf{Obj}(\mathcal{U})} \{a\}\big) \rightarrow \bigsqcap_{\substack{n_1+n_2=n+1\\ n_2\leq \#\mathsf{Obj}(\mathcal{U})}} \Big(\neg(\geqslant n_1\ r\ \big(\big(\bigsqcap_{b\in\mathsf{Obj}(\mathcal{U})} \neg\{b\}\big) \sqcap C^{(0)}\big)\Big) \\ & \qquad \sqcup \neg(\geqslant n_2\ r^{(0)}\ \big(\big(\bigsqcup_{b\in\mathsf{Obj}(\mathcal{U})} \{b\}\big) \sqcap C^{(0)}\big))\Big)\Big)\end{aligned}$$

Figure 7: Bi-implications.

Intuitively, a word $r_1 \cdots r_n \in \mathsf{P}_C$ stands for a path composed of roles in C. For instance, $\mathsf{P}_{\forall r.(\forall s.A\sqcap\exists r^-.B)} = \{\epsilon, r, rs, rr^-\}$. We define

$$\mathsf{P}_{\mathsf{fle}} = \bigcup_{D\in\mathsf{Sub}(\mathcal{A})\cup\mathsf{Sub}(\mathcal{U})} \mathsf{P}_D.$$

Since updates can only change the interpretations of concept names and role names on named objects, for all interpretations $\mathcal{I}$ and for all $\mathcal{ALCQIO}^{@}$-concept assertions $C(a)$, whether $\mathcal{I} \models C(a)$ depends only on those $x \in \Delta^{\mathcal{I}}$ such that x is reachable from a named object in $\mathcal{I}$ via a path in $\mathsf{P}_{\mathsf{fle}}$. The following ABox $\mathcal{A}_{\mathsf{rel}}$ ensures that C_{bi} holds at all of such x. We use $\forall\epsilon.C$ to denote C and $\forall r_1 \cdots r_n.C$ to denote $\forall r_1.\ldots.\forall r_n.C$. Now we define the ABox $\mathcal{A}_{\mathsf{rel}}$ as follows:

$$\mathcal{A}_{\mathsf{rel}} = \{\forall uw.C_{\mathsf{bi}}(a^*) \mid w \in \mathsf{P}_{\mathsf{fle}}\}.$$

Finally, we define

$$\mathcal{A}' = \mathcal{A}_{\mathsf{ini}} \cup \mathcal{A}_{\mathsf{r}} \cup \mathcal{A}_{\mathsf{aux}} \cup \mathcal{A}_{\mathcal{U}} \cup \mathcal{A}_{\overline{\mathcal{U}}} \cup \mathcal{A}_{\mathsf{rel}}. \tag{6}$$

In the next lemma, we show that the above ABox $\mathcal{A}'$ is a projective update of $\mathcal{A}$ with $\mathcal{U}$:

Lemma 46. *Let $\mathcal{A}$ be an $\mathcal{ALCQIO}^{@}$-ABox and $\mathcal{U}$ an update. Let $\mathcal{A}'$ be the $\mathcal{ALCQIO}^{@}$-ABox defined as (6). Then, $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}} \mathcal{A}'$.*

Proof. We show that $M(\mathcal{A} * \mathcal{U})_{|\mathsf{S}} = M(\mathcal{A})_{|\mathsf{S}}$, where S is defined as in Definition 27.

"$\subseteq$":

Assume that there is an $\mathcal{I}$ in $M(\mathcal{A} * \mathcal{U})$. Then there exists a model $\mathcal{I}_0$ of $\mathcal{A}$ with $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$. We have to show that there exists a model $\mathcal{I}'$ of $\mathcal{A}'$ such that $\mathcal{I}_{|\mathsf{S}} = \mathcal{I}'_{|\mathsf{S}}$. We define $\mathcal{I}'$ by extending $\mathcal{I}_{|\mathsf{S}}$ as follows:

$$\begin{aligned}
(a^*)^{\mathcal{I}'} &= x, \text{ for some } x \in \Delta^{\mathcal{I}},\\
u^{\mathcal{I}'} &= \{((a^*)^{\mathcal{I}}, b^{\mathcal{I}}) \mid b \in \mathsf{Obj}\},\\
(r^{(0)})^{\mathcal{I}'} &= r^{\mathcal{I}_0}, \text{ for } r \in \mathsf{R}_{\mathsf{fle}},\\
(\mathsf{A}_C^{(0)})^{\mathcal{I}'} &= C^{\mathcal{I}_0}, \text{ for } C \in \mathsf{C}_{\mathsf{fle}}.
\end{aligned}$$

Claim 1. For all concepts $C \in \mathsf{Sub}(\mathcal{A}) \cup \mathsf{Sub}(\mathcal{U})$ and all objects $d \in \Delta^{\mathcal{I}} (= \Delta^{\mathcal{I}_0} = \Delta^{\mathcal{I}'})$, $d \in C^{\mathcal{I}_0}$ iff $d \in (C^{(0)})^{\mathcal{I}'}$.

The proof of the claim: If $C \notin \mathsf{C}_{\mathsf{fle}}$, then $C^{(0)} = C$. Thus, $d \in C^{\mathcal{I}_0}$ iff (since $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$ and $C \notin \mathsf{C}_{\mathsf{fle}}$) $d \in C^{\mathcal{I}}$ iff (since $\mathcal{I}'_{|\mathsf{S}} = \mathcal{I}_{|\mathsf{S}}$ and $\mathsf{Sig}(C) \subseteq \mathsf{S}$) $d \in C^{\mathcal{I}'}$ iff $d \in (C^{(0)})^{\mathcal{I}'}$. If $C \in \mathsf{C}_{\mathsf{fle}}$, then $C^{(0)} = \mathsf{A}_C^{(0)}$. Thus, $d \in C^{\mathcal{I}_0}$ iff (by the definition of $\mathcal{I}'$) $d \in (A_C^{(0)})^{\mathcal{I}'}$ iff $d \in (C^{(0)})^{\mathcal{I}'}$. This finishes the proof of Claim 1.

Claim 2. For all $x, y \in \Delta^{\mathcal{I}}$, if either $x \neq a^{\mathcal{I}}$ or $y \neq a^{\mathcal{I}}$ for all $a \in \mathsf{Obj}(\mathcal{U})$, then for all role names r in S, $(x, y) \in r^{\mathcal{I}_0}$ iff $(x, y) \in r^{\mathcal{I}'}$. Otherwise, $(x, y) \in r^{\mathcal{I}_0}$ iff $(x, y) \in (r^{(0)})^{\mathcal{I}'}$.

The proof of the claim: If either $x \neq a^{\mathcal{I}}$ or $y \neq a^{\mathcal{I}}$ for all $a \in \mathsf{Obj}(\mathcal{U})$, then $(x, y) \in r^{\mathcal{I}_0}$ iff $(x, y) \in r^{\mathcal{I}}$, since $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$. Moreover, we have $(x, y) \in r^{\mathcal{I}'}$ iff $(x, y) \in r^{\mathcal{I}}$ since $\mathcal{I}_{|\mathsf{S}} = \mathcal{I}'_{|\mathsf{S}}$ and r is in S. Thus, we obtain that $(x, y) \in r^{\mathcal{I}_0}$ iff $(x, y) \in r^{\mathcal{I}'}$.

If there are $a, b \in \mathsf{Obj}(\mathcal{U})$ such that $x = a^{\mathcal{I}}$ and $y = b^{\mathcal{I}}$, then $r \in \mathsf{R}_{\mathsf{fle}}$ implies $r^{(0)} = \mathsf{s}_r^{(0)}$. By the definition of $\mathcal{I}'$, we have $(x, y) \in (\mathsf{s}_r^{(0)})^{\mathcal{I}'}$ iff $(x, y) \in r^{\mathcal{I}_0}$. If $r \notin \mathsf{R}_{\mathsf{fle}}$, then $r^{(0)} = r$. Moreover, since $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$ and $r \notin \mathsf{R}_{\mathsf{fle}}$, we have $(x, y) \in r^{\mathcal{I}_0}$ iff $(x, y) \in r^{\mathcal{I}}$. By the definition of $\mathcal{I}'$, we have $(x, y) \in r^{\mathcal{I}'}$ iff $(x, y) \in r^{\mathcal{I}}$. Thus, we obtain that $(x, y) \in r^{\mathcal{I}_0}$ iff $(x, y) \in r^{\mathcal{I}'}$. This finishes the proof of Claim 2.

Note that it follows from Claim 2 that for all $x, y \in \Delta^{\mathcal{I}}$, if either $x \neq a^{\mathcal{I}}$ or $y \neq a^{\mathcal{I}}$ for all $a \in \mathsf{Obj}(\mathcal{U})$, then for all role names r in S, $(x, y) \in (r^-)^{\mathcal{I}_0}$ iff $(x, y) \in (r^-)^{\mathcal{I}'}$. Otherwise, $(x, y) \in (r^-)^{\mathcal{I}_0}$ iff $(x, y) \in ((r^-)^{(0)})^{\mathcal{I}'}$.

Claim 3. For all $x \in \Delta^{\mathcal{I}'}$, $x \in (C_{\mathsf{bi}})^{\mathcal{I}'}$.

Since C_{bi} is the conjunction of concepts from Figure 6, where the left hand side ranges over $E \in \mathsf{C}_{\mathsf{fle}}$, it is enough to show that for all $E \in \mathsf{C}_{\mathsf{fle}}$, x is in the interpretation of the corresponding concept to E. We prove this by induction on the structure of E.

$E = A$ for some $A \in \mathsf{N_C}$: We show that

$$x \in ((\mathsf{A}_A^{(0)} \sqcap \neg(\bigsqcup_{\{A(a), \neg A(a)\} \cap \mathsf{rhs}(\mathcal{U}) \neq \emptyset} \{a\})) \leftrightarrow (A \sqcap \neg(\bigsqcup_{\{A(a), \neg A(a)\} \cap \mathsf{rhs}(\mathcal{U}) \neq \emptyset} \{a\})))^{\mathcal{I}'}.$$

Let $X = \{a^{\mathcal{I}'} \mid A(a) \in \mathsf{rhs}(\mathcal{U}) \text{ or } \neg A(a) \in \mathsf{rhs}(\mathcal{U})\}$. It suffices to show that for all $x \notin X$, $x \in (\mathsf{A}_A^{(0)})^{\mathcal{I}'}$ iff $x \in A^{\mathcal{I}'}$. This holds since $x \in (\mathsf{A}_A^{(0)})^{\mathcal{I}'}$ iff (by the definition of $\mathcal{I}'$) $x \in A^{\mathcal{I}_0}$ iff (since $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$ and x is not in X) $x \in A^{\mathcal{I}}$ iff (since $\mathcal{I}_{\restriction \mathsf{S}} = \mathcal{I}'_{\restriction \mathsf{S}}$ and A is in S) $x \in A^{\mathcal{I}'}$.

The cases $E = \top$, $E = \bot$, $E = \{a\}$, $E = @_a C$, $E = \neg C$, $E = C \sqcap D$ and $E = C \sqcup D$ follow directly from the semantics of the concept and the induction hypothesis.

Now we show the case of $E = (\geqslant n\ r\ C)$. Let Y be the concept on the right-hand side of $\leftrightarrow$. It is enough to show that $x \in (\mathsf{A}^{(0)}_{(\geqslant n\ r\ C)})^{\mathcal{I}'}$ iff $x \in Y^{\mathcal{I}'}$.

For all $x \in \Delta^{\mathcal{I}'}$, $x \in (\mathsf{A}^{(0)}_{(\geqslant n\ r\ C)})^{\mathcal{I}'}$ iff (by the definition of $\mathcal{I}'$) $x \in (\geqslant n\ r\ C)^{\mathcal{I}_0}$ iff (by semantics of $(\geqslant n\ r\ C)$) iff there are pairwise different $d_1, \ldots, d_n$ such that $(x, d_i) \in r^{\mathcal{I}_0}$, $d_i \in C^{\mathcal{I}_0}$ for all $i \in \{1, \ldots, n\}$ iff

- $x \neq a^{\mathcal{I}_0}$ for all $a \in \mathsf{Obj}(\mathcal{U})$ and there are pairwise different $d_1, \ldots, d_n$ such that $(x, d_i) \in r^{\mathcal{I}_0}$, $d_i \in C^{\mathcal{I}_0}$ for all $i \in \{1, \ldots, n\}$, or
- $x = a^{\mathcal{I}_0}$ for some $a \in \mathsf{Obj}(\mathcal{U})$ and there are pairwise different

$$d_1, \ldots, d_{n_1}, e_1, \ldots, e_{n_2}$$

such that $n_1 + n_2 = n$, $n_2 \leq \#\mathsf{Obj}(\mathcal{U})$, for all $i \in \{1, \ldots, n_1\}$, we have $(x, d_i) \in r^{\mathcal{I}_0}$, $d_i \in C^{\mathcal{I}_0}$, and $d_i \neq b^{\mathcal{I}'}$ for all $b \in \mathsf{Obj}(\mathcal{U})$, and for all $i \in \{1, \ldots, n_2\}$, we have $(x, e_i) \in r^{\mathcal{I}_0}$, $e_i \in C^{\mathcal{I}_0}$, $d_i = b^{\mathcal{I}_0}$ for some $b \in \mathsf{Obj}(\mathcal{U})$.

iff

- $x \neq a^{\mathcal{I}'}$ for all $a \in \mathsf{Obj}(\mathcal{U})$ and there are pairwise different $d_1, \ldots, d_n$ such that $(x, d_i) \in r^{\mathcal{I}'}$ (by Claim 2), $d_i \in (C^{(0)})^{\mathcal{I}'}$ (by Claim 1) for all $i \in \{1, \ldots, n\}$, or
- $x = a^{\mathcal{I}'}$ for some $a \in \mathsf{Obj}(\mathcal{U})$ and there are pairwise different

$$d_1, \ldots, d_{n_1}, e_1, \ldots, e_{n_2}$$

such that $n_1 + n_2 = n$, $n_2 \leq \#\mathsf{Obj}(\mathcal{U})$, for all $i \in \{1, \ldots, n_1\}$, we have $(x, d_i) \in r^{\mathcal{I}'}$ (by Claim 2), $d_i \in (C^{(0)})^{\mathcal{I}'}$ (by Claim 1), and $d_i \neq b^{\mathcal{I}'}$ for all $b \in \mathsf{Obj}(\mathcal{U})$, and for all $i \in \{1, \ldots, n_2\}$, we have $(x, e_i) \in (r^{(0)})^{\mathcal{I}'}$ (by Claim 2), $e_i \in (C^{(0)})^{\mathcal{I}'}$ (by Claim 1), $d_i = b^{\mathcal{I}}$ for some $b \in \mathsf{Obj}(\mathcal{U})$.

iff (by the semantics of Y) $x \in Y^{\mathcal{I}'}$.

The case $E = (\leqslant n\ r\ C)$ can be shown similarly to the previous case. This completes the proof of Claim 3.

We are now ready to show $\mathcal{I}' \models \mathcal{A}'$, which implies $\mathcal{I}_{\restriction \mathsf{S}} \in M(\mathcal{A}')_{\restriction \mathsf{S}}$ since $\mathcal{I}_{\restriction \mathsf{S}} = \mathcal{I}'_{\restriction \mathsf{S}}$.

- $\mathcal{I}' \models \mathcal{A}_{\mathsf{ini}}$:
  - For all $C^{(0)}(a) \in \mathcal{A}_{\mathsf{ini}}$, we know that $C(a)$ is in $\mathcal{A}$. $\mathcal{I}_0 \models \mathcal{A}$ implies $\mathcal{I}_0 \models C(a)$ and thus $a^{\mathcal{I}_0} \in C^{\mathcal{I}_0}$. By Claim 1 and $a^{\mathcal{I}_0} = a^{\mathcal{I}'}$, we have $a^{\mathcal{I}'} \in (C^{(0)})^{\mathcal{I}'}$. Hence, $\mathcal{I}' \models C^{(0)}(a)$.

- For all $r(a,b) \in \mathcal{A}_{\mathsf{ini}}$, by the definition of $\mathcal{A}_{\mathsf{ini}}$, we know that $r(a,b) \in \mathcal{A}$ and $\{a,b\} \not\subseteq \mathsf{Obj}(\mathcal{U})$. $\mathcal{I}_0 \models \mathcal{A}$ implies $\mathcal{I}_0 \models r(a,b)$. By Claim 2, we get $\mathcal{I}' \models r(a,b)$.
- For all $r^{(0)}(a,b) \in \mathcal{A}_{\mathsf{ini}}$, we know that $r(a,b)$ is in $\mathcal{A}$ and $\{a,b\} \subseteq \mathsf{Obj}(\mathcal{U})$. $\mathcal{I}_0 \models \mathcal{A}$ implies $\mathcal{I}_0 \models r(a,b)$. By Claim 2, we get $\mathcal{I}' \models r^{(0)}(a,b)$.
- The cases $\neg r(a,b) \in \mathcal{A}_{\mathsf{ini}}$ and $\neg r^{(0)}(a,b) \in \mathcal{A}_{\mathsf{ini}}$ can be proved similarly.

- $\mathcal{I}' \models \mathcal{A}_{\mathsf{r}}$: It is enough to show that for all $r \in \mathsf{R}_{\mathsf{fle}}$, $a,b \in \mathsf{Obj}(\mathcal{U})$ such that $\{r(a,b), \neg r(a,b)\} \cap \mathsf{rhs}(\mathcal{U}) = \emptyset$, $\mathcal{I}' \models \exists \mathsf{s}_r^{(0)}.\{b\}(a)$ iff $\mathcal{I}' \models \exists r.\{b\}(a)$, which is equivalent to showing that $\mathcal{I}' \models \mathsf{s}_r^{(0)}(a,b)$ iff $\mathcal{I}' \models r(a,b)$.

  Since $\{r(a,b), \neg r(a,b)\} \cap \mathsf{rhs}(\mathcal{U}) = \emptyset$ and $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$, we have $\mathcal{I}_0 \models r(a,b)$ iff $\mathcal{I} \models r(a,b)$. By the definition of $\mathcal{I}'$, we know that $(\mathsf{s}_r^{(0)})^{\mathcal{I}'} = r^{\mathcal{I}_0}$ and $r^{\mathcal{I}'} = r^{\mathcal{I}}$. Hence, $\mathcal{I}' \models \mathsf{s}_r^{(0)}(a,b)$ iff $\mathcal{I}_0 \models r(a,b)$, and $\mathcal{I}' \models r(a,b)$ iff $\mathcal{I} \models r(a,b)$. Thus, $\mathcal{I}' \models \mathsf{s}_r^{(0)}(a,b)$ iff $\mathcal{I}' \models r(a,b)$.

- $\mathcal{I}' \models \mathcal{A}_{\mathsf{aux}}$: This is a direct consequence of the definition of $u^{\mathcal{I}'}$.

- $\mathcal{I}' \models \mathcal{A}_{\mathcal{U}}$: By the definition of $\mathcal{I}'$ and $\mathcal{A}_{\mathsf{aux}}$, it is enough to show that for all $\varphi/\psi \in \mathcal{U}$, $\mathcal{I}' \models \varphi^{(0)}$ implies $\mathcal{I}' \models \psi$. Consider $\varphi/\psi \in \mathcal{U}$. If φ is a concept assertion, then by Claim 1 we know that $\mathcal{I}' \models \varphi^{(0)}$ implies that $\mathcal{I}_0 \models \varphi$. If φ is a role assertion, then by Claim 2 we know that $\mathcal{I}' \models \varphi^{(0)}$ implies that $\mathcal{I}_0 \models \varphi$. Since $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$, we know that $\mathcal{I}_0 \models \varphi$ implies $\mathcal{I} \models \psi$. Since $\mathsf{Sig}(\psi) \subseteq \mathsf{S}$ and $\mathcal{I}_{\restriction \mathsf{S}} = \mathcal{I}'_{\restriction \mathsf{S}}$, we know that $\mathcal{I} \models \psi$ implies $\mathcal{I}' \models \psi$.

- $\mathcal{I}' \models \mathcal{A}_{\overline{\mathcal{U}}}$ can be proved similarly to $\mathcal{I}' \models \mathcal{A}_{\mathcal{U}}$.

- $\mathcal{I}' \models \mathcal{A}_{\mathsf{rel}}$: This follows from Claim 3 and the fact that $\mathcal{I}' \models \mathcal{A}_{\mathsf{aux}}$.

"$\supseteq$":

Assume that $\mathcal{I} \models \mathcal{A}'$. We construct $\mathcal{I}_0$ such that $\mathcal{I}_0 \models \mathcal{A}$ and $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$. Thus, $\mathcal{I}_{\restriction \mathsf{S}} \in M(\mathcal{A} * \mathcal{U})_{\restriction \mathsf{S}}$. Define $\mathcal{I}_0$ as follows: $\Delta^{\mathcal{I}_0} = \Delta^{\mathcal{I}}$; for all individual names a, $a^{\mathcal{I}_0} = a^{\mathcal{I}}$; and for all concept names $A \notin \mathsf{C}_{\mathsf{fle}}$, set $A^{\mathcal{I}_0} = A^{\mathcal{I}}$. For all concept names $A \in \mathsf{C}_{\mathsf{fle}}$, set $d \in A^{\mathcal{I}_0}$ iff:

1. $d \in A^{\mathcal{I}}$ and $d \notin \{a^{\mathcal{I}} \mid a \in \mathsf{Obj}(\mathcal{U})\}$; or
2. $d \in (\mathsf{A}_A^{(0)})^{\mathcal{I}}$ and $d \in \{a^{\mathcal{I}} \mid a \in \mathsf{Obj}(\mathcal{U})\}$.

Similarly, for all role names $r \notin \mathsf{R}_{\mathsf{fle}}$, set $r^{\mathcal{I}_0} = r^{\mathcal{I}}$. For all role names $r \in \mathsf{R}_{\mathsf{fle}}$, set $(d_1,d_2) \in r^{\mathcal{I}_0}$ iff:

1. $(d_1,d_2) \in r^{\mathcal{I}}$ and $\{d_1,d_2\} \not\subseteq \{a^{\mathcal{I}} \mid a \in \mathsf{Obj}(\mathcal{U})\}$; or
2. $(d_1,d_2) \in (\mathsf{s}_r^{(0)})^{\mathcal{I}}$ and $\{d_1,d_2\} \subseteq \{a^{\mathcal{I}} \mid a \in \mathsf{Obj}(\mathcal{U})\}$.

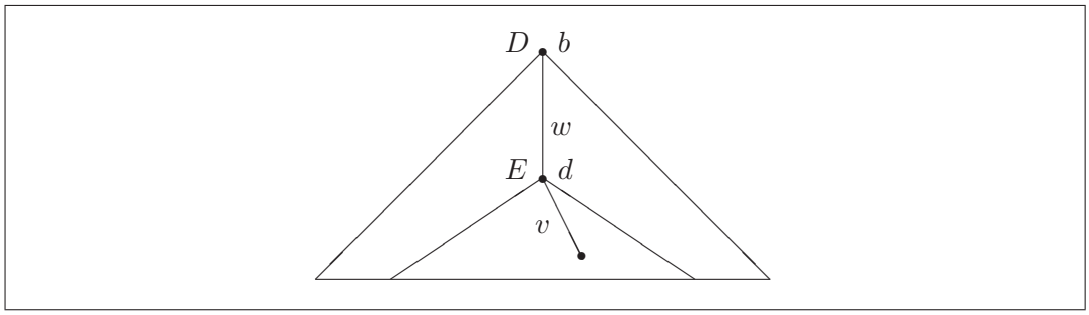


Figure 8: Concept descriptions D, E, paths w, v of roles, individual name b and domain element d in the proof of Lemma 46.

It follows from the definition of $\mathcal{I}_0$ that

> for all $x, y \in \Delta^{\mathcal{I}}$, if either $x \neq a^{\mathcal{I}}$ or $y \neq a^{\mathcal{I}}$ for all $a \in \mathsf{Obj}(\mathcal{U})$, then for all role names r in S, $(x, y) \in r^{\mathcal{I}_0}$ iff $(x, y) \in r^{\mathcal{I}}$. Otherwise, $(x, y) \in r^{\mathcal{I}_0}$ iff $(x, y) \in (r^{(0)})^{\mathcal{I}}$. (a)

As an immediate result of (a), we have that for all $x, y \in \Delta^{\mathcal{I}}$, if either $x \neq a^{\mathcal{I}}$ or $y \neq a^{\mathcal{I}}$ for all $a \in \mathsf{Obj}(\mathcal{U})$, then for all role names r in S, $(x, y) \in (r^-)^{\mathcal{I}_0}$ iff $(x, y) \in (r^-)^{\mathcal{I}}$. Otherwise, $(x, y) \in (r^-)^{\mathcal{I}_0}$ iff $(x, y) \in ((r^-)^{(0)})^{\mathcal{I}}$.

We show that $\mathcal{I}_0 \models \mathcal{A}$ and $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$.

Let $w = r_1 \cdots r_n$ be a path of roles and $\mathcal{I}$ an interpretation. We define $w^{\mathcal{I}}$ as follows:

$$w^{\mathcal{I}} = \begin{cases} \{(x, x) \mid x \in \Delta^{\mathcal{I}}\} & \text{if } n = 0 \\ \{(x_0, x_n) \mid \exists x_1, \ldots, x_{n-1} \in \Delta^{\mathcal{I}}. \forall i < n : (x_i, x_{i+1}) \in r_{i+1}^{\mathcal{I}}\} & \text{if } n > 0 \end{cases}$$

Let D be a concept such that $D \in \mathsf{Sub}(\mathcal{A}) \cup \mathsf{Sub}(\mathcal{U})$ and E a subconcept of D. Let $w \in \mathsf{P}_D$ be a path such that $wv \in \mathsf{P}_D$ for all $v \in \mathsf{P}_E$. The relationships between those concepts and paths of roles are displayed in Figure 8. Moreover, let $d \in \Delta^{\mathcal{I}}$ be such that $(b^{\mathcal{I}}, d) \in w^{\mathcal{I}}$ for some individual name b occurring in $\mathcal{A}$ or $\mathcal{U}$. We show by induction on the structure of E that:

$$d \in E^{\mathcal{I}_0} \text{ iff } d \in (E^{(0)})^{\mathcal{I}}. \tag{b}$$

$E = A$, a concept name. If $A \notin \mathsf{C}_{\mathsf{fle}}$, then $A^{(0)} = A$ and thus (b) is true by the definition of $A^{\mathcal{I}_0}$. If $A \in \mathsf{C}_{\mathsf{fle}}$, then $A^{(0)} = \mathsf{A}_A^{(0)}$. If $d \notin \{a^{\mathcal{I}} \mid a \in \mathsf{Obj}(\mathcal{U})\}$, then $d \in (\mathsf{A}_A^{(0)})^{\mathcal{I}}$ iff (since $\mathcal{I} \models \mathcal{A}_{\mathsf{rel}} \cup \mathcal{A}_{\mathsf{aux}}$) $d \in A^{\mathcal{I}}$ iff (by the definition of $A^{\mathcal{I}_0}$) $d \in A^{\mathcal{I}_0}$. If $d \in \{a^{\mathcal{I}} \mid a \in \mathsf{Obj}(\mathcal{U})\}$, then $d \in A^{\mathcal{I}_0}$ iff $d \in (\mathsf{A}_A^{(0)})^{\mathcal{I}}$ by the definition of $A^{\mathcal{I}_0}$.

The cases $E = \top$, $E = \bot$, $E = \{a\}$, $E = @_a F$, $E = \neg F$, $E = F \sqcap G$ and $E = F \sqcup G$ follow directly from the semantics of the concept and the induction hypothesis.

Now we show the case of $E = (\geqslant n\ r\ F)$. Then wr is in P_D, and $(wr)v$ is in P_D for all $v \in \mathsf{P}_F$. If $E \notin \mathsf{C}_{\mathsf{fle}}$, then $E^{(0)} = E$ and thus (b) is true by definition of $\mathcal{I}_0$. If $E \in \mathsf{C}_{\mathsf{fle}}$, then $E^{(0)} = \mathsf{A}_E^{(0)}$. We distingush the following cases:

- $d \notin \{a^{\mathcal{I}} \mid a \in \mathsf{Obj}(\mathcal{U})\}$. We have that $d \in (\mathsf{A}_E^{(0)})^{\mathcal{I}}$ iff (since $\mathcal{I} \models \mathcal{A}_{\mathsf{rel}} \cup \mathcal{A}_{\mathsf{aux}}$) $d \in (\geqslant n\ r\ F^{(0)})^{\mathcal{I}}$. By (a), for all $x \in \Delta^{\mathcal{I}}$, we have that $(d, x) \in r^{\mathcal{I}}$ iff $(d, x) \in r^{\mathcal{I}_0}$.

Moreover, $(d,x) \in r^{\mathcal{I}}$ implies that $(b^{\mathcal{I}},x) \in (wr)^{\mathcal{I}}$. By I.H., it holds that $x \in F^{\mathcal{I}_0}$ iff $x \in (F^{(0)})^{\mathcal{I}}$. Thus we obtain that $d \in (\mathsf{A}_E^{(0)})^{\mathcal{I}}$ iff $d \in (\geqslant n\ r\ F)^{\mathcal{I}_0}$, i.e., $d \in E^{\mathcal{I}_0}$.

- $d \in \{a^{\mathcal{I}} \mid a \in \mathsf{Obj}(\mathcal{U})\}$. We have that $d \in (\mathsf{A}_E^{(0)})^{\mathcal{I}}$ iff (since $\mathcal{I} \models \mathcal{A}_{\mathsf{rel}} \cup \mathcal{A}_{\mathsf{aux}}$) for some n_1, n_2 such that $n_1 + n_2 = n$ and $n_2 \leq \#\mathsf{Obj}(\mathcal{U})$ it holds that $d \in (\geqslant n_1\ r\ (\bigsqcap_{b \in \mathsf{Obj}(\mathcal{U})} \neg\{b\} \sqcap F^{(0)}))^{\mathcal{I}}$ and $d \in (\geqslant n_2\ r^{(0)}\ (\bigsqcup_{b \in \mathsf{Obj}(\mathcal{U})} \{b\} \sqcap F^{(0)}))^{\mathcal{I}}$. Analogously to the previous case it can be shown that

$$d \in (\geqslant n_1\ r\ (\bigsqcap_{b \in \mathsf{Obj}(\mathcal{U})} \neg\{b\} \sqcap F^{(0)}))^{\mathcal{I}} \text{ iff } d \in (\geqslant n_1\ r\ (\bigsqcap_{b \in \mathsf{Obj}(\mathcal{U})} \neg\{b\} \sqcap F))^{\mathcal{I}_0}.$$

Moreover, if $x = c^{\mathcal{I}}$, for some $c \in \mathsf{Obj}(\mathcal{U})$, by (a), we have $(d,x) \in r^{\mathcal{I}_0}$ iff $(d,x) \in (r^{(0)})^{\mathcal{I}}$. Since $F \in \mathsf{Sub}(\mathcal{A}) \cup \mathsf{Sub}(\mathcal{U})$, F is a subconcept of F, $\epsilon \in \mathsf{P}_F$, $(c^{\mathcal{I}},x) \in \epsilon^{\mathcal{I}}$ and $\epsilon v \in \mathsf{P}_F$ for all $v \in \mathsf{P}_F$, by I.H., we have that $c^{\mathcal{I}} \in F^{\mathcal{I}_0}$ iff $c^{\mathcal{I}} \in (F^{(0)})^{\mathcal{I}}$. Thus, we obtain that $d \in (\geqslant n_2\ r^{(0)}\ (\bigsqcup_{b \in \mathsf{Obj}(\mathcal{U})} \{b\} \sqcap F^{(0)}))^{\mathcal{I}}$ iff $d \in (\geqslant n_2\ r\ (\bigsqcup_{b \in \mathsf{Obj}(\mathcal{U})} \{b\} \sqcap F))^{\mathcal{I}_0}$.

Summing up the previous equivalences, we obtain that $d \in (\mathsf{A}_E^{(0)})^{\mathcal{I}}$ iff $d \in E^{\mathcal{I}_0}$.

The case $E = (\leqslant n\ r\ C)$ can be shown similarly to the previous case. This finishes the proof of (b).

We now show that $\mathcal{I}_0 \models \mathcal{A}$. Let $C(a)$ be a concept assertion in $\mathcal{A}$. Then, $\mathcal{I} \models C^{(0)}(a)$. Since C is in $\mathsf{Sub}(\mathcal{A}) \cup \mathsf{Sub}(\mathcal{U})$, C is a subconcept of itself, the empty word ϵ is in P_C, $\epsilon v \in \mathsf{P}_C$ for all $v \in \mathsf{P}_C$, and $(a^{\mathcal{I}}, a^{\mathcal{I}}) \in \epsilon^{\mathcal{I}}$, we obtain by (b) that $C^{\mathcal{I}_0}$ iff $(C^{(0)})^{\mathcal{I}}$. Thus, $\mathcal{I} \models C^{(0)}(a)$ implies $\mathcal{I}_0 \models C(a)$. For all role assertions $\varphi \in \mathcal{A}$, it follows from (a) that $\mathcal{I} \models \varphi^{(0)}$ implies that $\mathcal{I}_0 \models \varphi$.

It is not hard to see that $\mathcal{I} \models \mathcal{A}_{\mathsf{aux}} \cup \mathcal{A}_{\mathcal{U}} \cup \mathcal{A}_{\overline{\mathcal{U}}}$ implies

> that for all $\varphi/\psi \in \mathcal{U}$, $\mathcal{I}_0 \models \varphi$ implies $\mathcal{I} \models \psi$, and that for all $\psi \in \mathsf{rhs}(\mathcal{U})$, if $\mathcal{I}_0 \not\models \varphi$ for all φ with $\varphi/\psi \in \mathcal{U}$, then $\mathcal{I}_0 \models \psi$ iff $\mathcal{I} \models \psi$. (c)

It remains to show that $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$. First, interpretations of all concept names $A \notin \mathsf{C}_{\mathsf{fle}}$ and all role names $r \notin \mathsf{R}_{\mathsf{fle}}$ are identical in $\mathcal{I}_0$ and $\mathcal{I}$. Second, $\mathcal{I}$ and $\mathcal{I}_0$ interpret all concept names $A \in \mathsf{C}_{\mathsf{fle}}$ and all role names $r \in \mathsf{R}_{\mathsf{fle}}$ in the same way on the part of the domain $\Delta^{\mathcal{I}}$ unaffected by the update $\mathcal{U}$, i.e., on $\Delta^{\mathcal{I}} \setminus \{a^{\mathcal{I}} \mid a \in \mathsf{Obj}(\mathcal{U})\}$.

Consider $x = a^{\mathcal{I}}$ for some $a \in \mathsf{Obj}(\mathcal{U})$. If $\{A(a), \neg A(a)\} \cap \mathsf{rhs}(\mathcal{U}) \neq \emptyset$, then there exists some $\varphi/\psi \in \mathcal{U}$ such that $\psi = A(a)$ or $\psi = \neg A(a)$. Thus, (c) guarantees that the interpretations of A in $\mathcal{I}_0$ and $\mathcal{I}$ respect $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$ on x. If $\{A(a), \neg A(a)\} \cap \mathsf{rhs}(\mathcal{U}) = \emptyset$, then $\mathcal{I} \models \mathcal{A}_{\mathsf{rel}} \cup \mathcal{A}_{\mathsf{aux}}$ implies that $x \in (\mathsf{A}_A^{(0)})^{\mathcal{I}}$ iff $x \in A^{\mathcal{I}}$. By (c), $x \in (\mathsf{A}_A^{(0)})^{\mathcal{I}}$ iff $x \in A^{\mathcal{I}_0}$. Thus, we have $x \in A^{\mathcal{I}}$ iff $x \in A^{\mathcal{I}_0}$.

Likewise, consider $x = a^{\mathcal{I}}$ and $y = b^{\mathcal{I}}$ for some $a, b \in \mathsf{Obj}(\mathcal{U})$. If $\{r(a,b), \neg r(a,b)\} \cap \mathsf{rhs}(\mathcal{U}) \neq \emptyset$, then there exists some $\varphi/\psi \in \mathcal{U}$ such that $\psi = r(a,b)$ or $\psi = \neg r(a,b)$. Thus, (c) ensures that $r^{\mathcal{I}_0}$ and $r^{\mathcal{I}}$ respect $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$ on (x,y). If $\{r(a,b), \neg r(a,b)\} \cap \mathsf{rhs}(\mathcal{U}) = \emptyset$, then $\mathcal{I} \models \mathcal{A}_{\mathsf{r}}$ implies that $(x,y) \in (\mathsf{s}_r^{(0)})^{\mathcal{I}_0}$ iff $(x,y) \in r^{\mathcal{I}}$. By the definition of $r^{\mathcal{I}_0}$, $(x,y) \in (\mathsf{s}_r^{(0)})^{\mathcal{I}_0}$ iff $(x,y) \in r^{\mathcal{I}_0}$. Thus, we have $(x,y) \in r^{\mathcal{I}_0}$ iff $(x,y) \in r^{\mathcal{I}}$.

Overall, we obtain that $\mathcal{I}_0 \Longrightarrow_{\mathcal{U}} \mathcal{I}$. ❏

$$
\begin{aligned}
\mathsf{A}^{(0)}_{\exists r.C} \quad \leftrightarrow \quad & \Big(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\}\Big) \sqcap \exists r.C^{(0)} \sqcup \\
& \Big(\Big(\bigsqcup_{a\in\mathsf{Obj}(\mathcal{U})} \{a\}\Big) \sqcap \big(\exists r.\Big(\Big(\bigsqcap_{b\in\mathsf{Obj}(\mathcal{U})} \neg\{b\}\Big) \sqcap C^{(0)}\Big) \\
& \qquad \sqcup \exists r^{(0)}.\Big(\Big(\bigsqcup_{b\in\mathsf{Obj}(\mathcal{U})} \{b\}\Big) \sqcap C^{(0)}\Big)\big)\Big) \\
\mathsf{A}^{(0)}_{\forall r.C} \quad \leftrightarrow \quad & \Big(\Big(\bigsqcap_{a\in\mathsf{Obj}(\mathcal{U})} \neg\{a\}\Big) \rightarrow \forall r.C^{(0)}\Big) \sqcap \\
& \Big(\Big(\bigsqcup_{a\in\mathsf{Obj}(\mathcal{U})} \{a\}\Big) \rightarrow \big(\forall r.\Big(\Big(\bigsqcup_{b\in\mathsf{Obj}(\mathcal{U})} \{b\}\Big) \sqcup C^{(0)}\Big) \\
& \qquad \sqcap \forall r^{(0)}.\Big(\Big(\bigsqcap_{b\in\mathsf{Obj}(\mathcal{U})} \neg\{b\}\Big) \sqcup C^{(0)}\Big)\big)\Big)
\end{aligned}
$$

Figure 9: Bi-implications for existential and value restrictions.

For DLs $\mathcal{L}$ between $\mathcal{ALCO}$ and $\mathcal{ALCIO}^{@}$, the bi-implications of the concepts with existential and value restrictions in C_{bi} are displayed in Figure 9. As a direct consequence of Lemma 46 and the construction of projective updates, we achieve the following theorem:

Theorem 47. *All of the following DLs have projective updates: $\mathcal{ALCO}$, $\mathcal{ALCIO}$, $\mathcal{ALCQO}$, $\mathcal{ALCQIO}$, and their extensions with the @ constructor.*

Size of Projective Updates

It is not hard to see that $|\mathcal{A}_{\mathsf{ini}}| = O(|\mathcal{A}|)$, $|\mathcal{A}_{\mathsf{r}}| = O(|\mathcal{U}|^3)$, $|\mathcal{A}_{\mathsf{aux}}| = O(|\mathcal{A}| + |\mathcal{U}|)$, $|\mathcal{A}_{\mathcal{U}}| = O(|\mathcal{U}|)$, and $|\mathcal{A}_{\overline{\mathcal{U}}}| = O(|\mathcal{U}|)$. For every concept D, the size of bi-implication in Figure 7 is of size $O(|\mathcal{U}| \cdot (|\mathcal{U}| + |D|))$. It is clear that $\#\mathsf{C}_{\mathsf{fle}} \leq |\mathcal{A}| + |\mathcal{U}|$ and for every $D \in \mathsf{C}_{\mathsf{fle}}$, $|D| \leq |\mathcal{A}| + |\mathcal{U}|$. Thus, $|C_{\mathsf{bi}}| = O(|\mathcal{U}| \cdot (|\mathcal{U}| + |\mathcal{A}|)^2)$. For every $w = r_1 \cdots r_n \in \mathsf{P}_{\mathsf{fle}}$, $n \leq |\mathcal{A}| + |\mathcal{U}|$ and $\#\mathsf{P}_{\mathsf{fle}} \leq |\mathcal{A}| + |\mathcal{U}|$. Thus,

$$
\begin{aligned}
|\mathcal{A}_{\mathsf{rel}}| &= O((|\mathcal{A}| + |\mathcal{U}| + |\mathcal{U}| \cdot (|\mathcal{U}| + |\mathcal{A}|)^2) \cdot (|\mathcal{U}| + |\mathcal{A}|)) \\
&= O(|\mathcal{U}| \cdot (|\mathcal{A}| + |\mathcal{U}|)^3)
\end{aligned}
$$

Overall, we have $|\mathcal{A}'| = O(|\mathcal{U}| \cdot (|\mathcal{A}| + |\mathcal{U}|)^3)$.

We have shown that the size of $\mathcal{A}'$ is polynomial in the size of $\mathcal{A}$ and $\mathcal{U}$. More precisely, $|\mathcal{A}'|$ is cubic in $|\mathcal{A}|$ and biquadratic in $|\mathcal{U}|$. By a similar inspection of the sizes of those concepts constructed according to the bi-implications in Figure 9, we can obtain that for every $\mathcal{L}$-concept D with a DL $\mathcal{L}$ between $\mathcal{ALCO}$ and $\mathcal{ALCIO}^{@}$, the size of the bi-implication is $O(|\mathcal{U}| + |D|)$ and thus $|\mathcal{A}'| = O((|\mathcal{A}| + |\mathcal{U}|)^3)$. These bounds hold no matter how the numbers inside number restrictions are encoded.

Theorem 48. *Let $\mathcal{L}$ be a DL between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}^{@}$. Then for every $\mathcal{L}$-ABox $\mathcal{A}$ and every update $\mathcal{U}$, there exists an $\mathcal{L}$-ABox $\mathcal{A}'$ such that the following hold:*

- $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}} \mathcal{A}'$;
- $|\mathcal{A}'| = O(|\mathcal{U}| \cdot (|\mathcal{A}| + |\mathcal{U}|)^3)$;
- $\mathcal{A}'$ *can be computed in time* $O(|\mathcal{A}'|)$.

Projective Updates With Unconditional Updates

When we consider only unconditional updates $\mathcal{U}$, $\mathsf{Sub}(\mathcal{U})$ is defined as

$$\mathsf{Sub}(\mathcal{U}) = \bigcup_{C(a) \in \mathcal{U}} \mathsf{Sub}(C)$$

as usual since $\mathcal{U}$ is an ABox. Moreover, $\mathsf{P}_{\mathsf{fle}}$ is defined as

$$\mathsf{P}_{\mathsf{fle}} = \bigcup_{D \in \mathsf{Sub}(\mathcal{A})} \mathsf{P}_D,$$

since there are no complex concepts occurring in $\mathcal{U}$. As all of the effects are triggered in an unconditional update, $\mathcal{A}_{\mathcal{U}} = \mathcal{U}$, and $\mathcal{A}_{\overline{\mathcal{U}}}$ is now rendered unnecessary. Thus, we compute $\mathcal{A}'$ as follows:

$$\mathcal{A}' = \mathcal{A}_{\mathsf{ini}} \cup \mathcal{A}_{\mathsf{aux}} \cup \mathcal{A}_{\mathsf{r}} \cup \mathcal{U} \cup \mathcal{A}_{\mathsf{rel}}.$$

Accordingly, for every $w = r_1 \cdots r_n \in \mathsf{P}_{\mathsf{fle}}$, $n \leq |\mathcal{A}|$ and $\#\mathsf{P}_{\mathsf{fle}} \leq |\mathcal{A}|$ and we can obtain that $|\mathcal{A}'| = O(|\mathcal{U}| \cdot (|\mathcal{A}| + |\mathcal{U}|)^2 \cdot |\mathcal{A}|)$ if qualified number restriction is available in the DL under consideration; $|\mathcal{A}'| = O((|\mathcal{A}| + |\mathcal{U}|)^2 \cdot |\mathcal{A}| + |\mathcal{U}|^3)$ otherwise.

Projective Updates on Boolean ABoxes

Similar to logical updates on Boolean ABoxes, we say that a DL $\mathcal{L}$ has *projective updates on Boolean ABoxes* if, for every Boolean $\mathcal{L}$-ABox $\mathcal{A}$ and update $\mathcal{U}$, there exists a Boolean $\mathcal{L}$-ABox $\mathcal{A}'$ such that $M(\mathcal{A} * \mathcal{U})_{|\mathsf{S}} = M(\mathcal{A}')_{|\mathsf{S}}$, where S is defined as in Definition 27.

Let $\mathcal{L}$ be a DL in $\{\mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}, \mathcal{ALCQIO}\}$. It follows from Lemma 13 that $\mathcal{L}^{@}$ has projective updates on Boolean ABoxes: convert the original Boolean $\mathcal{L}^{@}$-ABox to an equivalent non-Boolean $\mathcal{L}^{@}$-ABox by the polynomial translation in the proof of Lemma 13[1] and then compute the projective update which is a non-Boolean $\mathcal{L}^{@}$-ABox of polynomial size of the input as in Section 4.1. Overall, the resulting ABox is polynomial in the size of the original ABox and the update.

$\mathcal{L}$ has also projective updates on Boolean ABoxes: by compiling away @ constructors in the resulting non-Boolean $\mathcal{L}^{@}$-ABox using the translation in the proof of

[1] In the proof of (ii) in Lemma 13, the mapping $\cdot^*$ can be easily extended to $\mathcal{L}^{@}$-ABox assertions. As a result, we can obtain that for every Boolean $\mathcal{L}^{@}$-ABox, there exists an equivalent non-Boolean $\mathcal{L}^{@}$-ABox.

Lemma 13, we obtain an equivalent Boolean $\mathcal{L}$-ABox. Although this translation is exponential in the worst case, the size of the Boolean $\mathcal{L}$-ABox constructed in this way is still polynomial in the size of the input since neither converting a Boolean $\mathcal{L}$-ABox to a non-Boolean $\mathcal{L}^{@}$-ABox nor the construction of projective updates induces nesting @ constructors.

Theorem 49. *All of the following DLs have projective updates on Boolean ABoxes which are of polynomial size in both the original ABox and the update: $\mathcal{ALCO}$, $\mathcal{ALCIO}$, $\mathcal{ALCQO}$, $\mathcal{ALCQIO}$, and their extensions with the @ constructor.*

The following lemma gives us another way to simulate the @ constructor by introducing an auxiliary role name.

Lemma 50. *Let $\mathcal{L}$ be a DL in $\{\mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}, \mathcal{ALCQIO}\}$. Then for every $\mathcal{L}^{@}$-ABox $\mathcal{A}$, there exists an $\mathcal{L}$-ABox $\mathcal{A}'$ such that*

$$M(\mathcal{A})_{\restriction(\mathsf{N_C}\cup\mathsf{N_R}\cup\mathsf{N_I})\setminus\{u\}} = M(\mathcal{A}')_{\restriction(\mathsf{N_C}\cup\mathsf{N_R}\cup\mathsf{N_I})\setminus\{u\}},$$

where u is a role name which does not occur in $\mathcal{A}$.

Proof. First convert all concepts occurring in $\mathcal{A}$ into the negation normal form (NNF), i.e., negation signs occurs only in front of concept names or nominals. This conversion can be done by the following rules:

$$\begin{array}{rclrcl} \neg(C \sqcap D) & \leadsto & \neg C \sqcup \neg D; & \neg(C \sqcup D) & \leadsto & \neg C \sqcap \neg D; \\ \neg(\geqslant (n+1)\ r\ C) & \leadsto & (\leqslant n\ r\ C); & \neg(\leqslant n\ r\ C) & \leadsto & (\geqslant (n+1)\ r\ C); \\ \neg(\geqslant 0\ r\ C) & \leadsto & \bot; & \neg@_a C & \leadsto & @_a(\neg C). \end{array}$$

Then, we obtain $\mathcal{A}'$ from $\mathcal{A}$ by replacing every concept of the form $@_a D$ in $\mathcal{A}$ with $\exists u.(\{a\} \sqcap C)$. It is easy to see that those two steps can be done in polynomial time of $|\mathcal{A}|$. Let $\mathsf{S} = (\mathsf{N_C} \cup \mathsf{N_R} \cup \mathsf{N_I}) \setminus \{u\}$. Now we show that $M(\mathcal{A})_{\restriction\mathsf{S}} = M(\mathcal{A}')_{\restriction\mathsf{S}}$.

"$\subseteq$": Let $\mathcal{I} \in M(\mathcal{A})$. Then we define $\mathcal{I}'$ by extending $\mathcal{I}_{\restriction\mathsf{S}}$ as follows:

$$u^{\mathcal{I}'} = \{(d, e) \mid \{d, e\} \subseteq \Delta^{\mathcal{I}}\}.$$

For an $\mathcal{ALCQIO}^{@}$-concept C, we use C' to denote the concept obtained from C by doing the forementioned replacement.

Claim 1. For all $d \in \Delta^{\mathcal{I}}$ and for all $\mathcal{ALCQIO}^{@}$-concepts C with $\mathsf{Sig}(C) \subseteq \mathsf{S}$, $d \in C^{\mathcal{I}}$ implies $d \in (C')^{\mathcal{I}'}$.

Proof of Claim 1. We prove Claim 1 by induction on the structure of C.

- The cases $C = A$, $C = \neg A$, $C = \{a\}$, $C = \neg\{a\}$, $C = D \sqcap E$, or $C = D \sqcup E$ follow from I.H., the definition of $\mathcal{I}'$, and the definition of C'.
- $C = (\geqslant n\ r\ D)$: Since $\mathsf{Sig}(C) \subseteq \mathsf{S}$, we know that $u \neq r$ and thus $r^{\mathcal{I}} = r^{\mathcal{I}'}$. It follows from $d \in C^{\mathcal{I}}$ that $\#\{e \mid e \in D^{\mathcal{I}} \wedge (d, e) \in r^{\mathcal{I}}\} \geq n$. By I.H., we get that $e \in (D')^{\mathcal{I}'}$ iff $e \in (D)^{\mathcal{I}}$. Thus, $\#\{e \mid e \in (D')^{\mathcal{I}'} \wedge (d, e) \in r^{\mathcal{I}'}\} \geq n$, which, together with $C' = (\geqslant n\ r\ D')$, implies that $d \in (C')^{\mathcal{I}'}$. The case $C = (\leqslant n\ r\ D)$ can be proved analogously.

- If $C = @_a D$, then $d \in (@_a D)^{\mathcal{I}}$ implies $a^{\mathcal{I}} \in D^{\mathcal{I}}$. By I.H., we know that $a^{\mathcal{I}} \in (D')^{\mathcal{I}'}$. It follows from the definition of $u^{\mathcal{I}'}$ that $(d, a^{\mathcal{I}}) \in u^{\mathcal{I}'}$. Moreover, since $a^{\mathcal{I}} = a^{\mathcal{I}'}$ and $C' = \exists u.(\{a\} \sqcap D')$, we get $d \in (C')^{\mathcal{I}'}$.

This finishes the proof of Claim 1.

We now show that $\mathcal{I}' \models \mathcal{A}'$. Let $\varphi \in \mathcal{A}'$. If φ is a role assertion, then $\varphi \in \mathcal{A}$. Since $\mathcal{I} \models \mathcal{A}$ and $\varphi \in \mathcal{A}$, $\mathcal{I} \models \varphi$. Moreover, since u does not occur in φ, it follows from the definition of $\mathcal{I}'$, $\mathcal{I}' \models \varphi$. If φ is a concept assertion $C'(a)$, then by Claim 1 we get $\mathcal{I}' \models C'(a)$ since $\mathcal{I} \models C(a)$ and $a^{\mathcal{I}} = a^{\mathcal{I}'}$.

"$\supseteq$": Let $\mathcal{I}' \in M(\mathcal{A}')$. We show that $\mathcal{I}' \models \mathcal{A}$. Similar to Claim 1, we can show that for all $d \in \Delta^{\mathcal{I}'}$ and all $\mathcal{ALCQIO}^{@}$-concepts C with $\mathsf{Sig}(C) \subseteq \mathsf{S}$, $d \in (C')^{\mathcal{I}'}$ implies $d \in C^{\mathcal{I}'}$ by induction on the structure of C (The proof is very similar to the one of Claim 1 and we do not show the details here). This yields that for all concept assertions $C(a) \in \mathcal{A}$, $\mathcal{I}' \models C(a)$ since $\mathcal{I}' \models C'(a)$. For all role assertions φ in $\mathcal{A}$, $\mathcal{I}' \models \varphi$ since $\mathcal{I}' \models \mathcal{A}'$ and $\varphi \in \mathcal{A}'$. ❑

This lemma generalizes Lemma 53 in [Bon07] which is used to show that $\mathcal{ALCO}$ has approximate projective updates. First, the DL under consideration is extended from $\mathcal{ALCO}$ to all DLs between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}$. Second, this lemma can be used to show existence of projective updates which are stronger than approximate projective updates.

Let $\mathcal{L}$ be a DL in Lemma 50. After we obtain a non-Boolean $\mathcal{L}^{@}$-ABox from the computation of the projective update, we use the construction in the proof of Lemma 50 to simulate the @ constructor and thus get a non-Boolean $\mathcal{L}$-ABox instead of using the construction in the proof of Lemma 13.

On the one hand, this construction is polynomial in the worst case, compared to the one in Lemma 13. On the other hand, it is not equivalence preserving because of the introduced role name u, but it preserves projective updates. It also follows from the proof of Lemma 50 that $\mathcal{A}$ is consistent iff $\mathcal{A}'$ is consistent, i.e., the construction of $\mathcal{A}'$ is consistency preserving. By induction on the Boolean structure of Boolean assertions, this claim can be easily extended to Boolean ABoxes:

Lemma 51. *Let $\mathcal{L}$ be a DL in $\{\mathcal{ALCO}, \mathcal{ALCIO}, \mathcal{ALCQO}, \mathcal{ALCQIO}\}$. Then for every Boolean $\mathcal{L}^{@}$-ABox $\mathcal{A}$, there exists a Boolean $\mathcal{L}$-ABox $\mathcal{A}'$ such that $\mathcal{A}$ is consistent iff $\mathcal{A}'$ is consistent.*

The above lemma will be used to decide consistency of Boolean ABoxes with the @ constructor in Section 6.2.

4.2 Iterated Projective Updates

In this section, we first show by an example that even for unconditional updates, the direct application of the construction in Section 4.1 to updating an ABox $\mathcal{A}$ iteratively leads to an exponential blowup in the size of $\mathcal{A}$.

Example 52. *Consider the ABox* $\mathcal{A}_0 = \{\exists r.A(a)\}$ *and the updates*

$$\mathcal{U}_1 = \cdots = \mathcal{U}_n = \{r(a,b)\}.$$

For all $i \in \{1,\ldots,n\}$*, we use* $C_{\mathsf{bi}}^{(i)}$ *to denote the concept* C_{bi} *if the update* $\mathcal{U}_i$ *is applied. When the projective update of* $\mathcal{A}_0$ *with* $\mathcal{U}_1$ *is constructed according to the construction in Section 4.1, the following concept*

$$\begin{aligned}\mathsf{A}_{\exists r.A}^{(0)} \quad \leftrightarrow \quad & (\neg\{a\} \sqcap \neg\{b\}) \sqcap \exists r.A \sqcup \\ & (\{a\} \sqcup \{b\}) \sqcap \big(\exists r.((\neg\{a\} \sqcap \neg\{b\}) \sqcap A) \sqcup \exists \mathsf{s}_r^{(0)}.((\{a\} \sqcup \{b\}) \sqcap A)\big)\end{aligned}$$

is a conjunct in $C_{\mathsf{bi}}^{(1)}$*. Thus, there are bi-implications for both* $\exists r.A$ *and* $\exists r.((\neg\{a\} \sqcap \neg\{b\}) \sqcap A)$ *when* $C_{\mathsf{bi}}^{(2)}$ *is computed since both concepts contain the flexible name* r *in* $\mathcal{U}_2$*. Similarly, the bi-implication for each of those concepts has two subconcepts containing* r*. Therefore, the size of* $C_{\mathsf{bi}}^{(i)}$ *is exponential in the size of* $\mathcal{A}_0$ *when* $\mathcal{A}_0$ *is updated iteratively.*

It is left as an open problem if this exponential blowup can be entirely avoided. Instead of computing projective updates iteratively, we exhibit a construction of a projective update of the original ABox with a finite sequence of updates. The updated ABox constructed in this way is bounded polynomially in the size of the input.

As introduced in Section 4.1, we use the name $\mathsf{A}_C^{(0)}$ ($\mathsf{s}_r^{(0)}$, respectively) to denote the concept C (role name r, respectively) before executing the update $\mathcal{U}_1$. Consider updates $\mathcal{U}_1,\ldots,\mathcal{U}_n$. For all $i \in \{1,\ldots,n\}$, we introduce a new name $\mathsf{A}_C^{(i-1)}$ ($\mathsf{s}_r^{(i-1)}$, respectively) to denote the concept C (role name r, respectively) before executing the update $\mathcal{U}_i$. Other notions need to be revised as follows:

- The set $\mathsf{Sub}(\mathcal{U})$ collects the subconcepts appearing in the sequence of updates:

$$\mathsf{Sub}(\mathcal{U}) = \bigcup_{i=1}^{n} \mathsf{Sub}(\mathcal{U}_i).$$

- We say that a concept C *contains a flexible name in* $\mathcal{U}_1,\ldots,\mathcal{U}_n$ iff there exists an $i \in \{1\ldots,n\}$ such that C contains a flexible name in $\mathcal{U}_i$.

- We use $\mathsf{C}_{\mathsf{fle}}$ to collect concepts which may be changed by some $\mathcal{U}_i$, which is defined as follows:

$$\mathsf{C}_{\mathsf{fle}} = \{C \mid C \in \mathsf{Sub}(\mathcal{A}) \cup \mathsf{Sub}(\mathcal{U}) \wedge C \text{ contains a flexible name in } \mathcal{U}_1,\ldots,\mathcal{U}_n\}.$$

- Similarly, we use $\mathsf{R}_{\mathsf{fle}}$ to denote the set of role names occurring in $\mathcal{U}_1,\ldots,\mathcal{U}_n$. For a concept C and a role name r, and an i with $i \in \{0,\ldots,n-1\}$, the mappings $C^{(i)}$ and $r^{(i)}$ are defined as follows:

$$C^{(i)} = \begin{cases} \mathsf{A}_C^{(i)} & \text{if } C \in \mathsf{C}_{\mathsf{fle}} \\ C & \text{otherwise} \end{cases} \qquad r^{(i)} = \begin{cases} \mathsf{s}_r^{(i)} & \text{if } r \in \mathsf{R}_{\mathsf{fle}} \\ r & \text{otherwise} \end{cases}$$

  Additionally, let $C^{(n)}$ and $r^{(n)}$ denote C and r, respectively, to streamline notation later on.

Let Obj be the set of all individual names occurring in $\mathcal{A}$ or $\mathcal{U}_1, \ldots, \mathcal{U}_n$. For an assertion φ and an $i \in \{0, \ldots, n\}$, the assertion $\varphi^{(i)}$ is defined as follows:

$$\varphi^{(i)} = \begin{cases} C^{(i)}(a) & \text{if } \varphi = C(a) \\ (\neg)r^{(i)}(a,b) & \text{if } \varphi = (\neg)r(a,b) \end{cases}$$

The ABox $\mathcal{A}_{\mathsf{ini}}$ is defined as before. For all $i \in \{1, \ldots, n\}$, we define an ABox $\mathcal{A}_{\mathsf{r}}^{(i)}$ as

$$\mathcal{A}_{\mathsf{r}}^{(i)} = \{(\exists r^{(i-1)}.\{b\} \leftrightarrow \exists r^{(i)}.\{b\})(a) \mid \{r(a,b), \neg r(a,b)\} \cap \mathsf{rhs}(\mathcal{U}_i) = \emptyset \wedge a, b \in \mathsf{Obj} \wedge r \in \mathsf{R}_{\mathsf{fle}}\}.$$

The ABox $\mathcal{A}_{\mathsf{aux}}$ and the abbreviation p are defined as before. For all $i \in \{1, \ldots, n\}$, we define an ABox $\mathcal{A}_{\mathcal{U}_i}$ as

$$\mathcal{A}_{\mathcal{U}_i} = \{a^* : \bigsqcap_{\varphi/\psi \in \mathcal{U}_i} (\mathsf{p}(\varphi^{(i-1)}) \rightarrow \mathsf{p}(\psi^{(i)}))\}.$$

Similarly, for all $i \in \{1, \ldots, n\}$, we define an ABox $\mathcal{A}_{\overline{\mathcal{U}_i}}$ as

$$\mathcal{A}_{\overline{\mathcal{U}_i}} = \{a^* : \bigsqcap_{\psi \in \mathsf{rhs}(\mathcal{U}_i)} ((\bigsqcap_{\varphi/\psi \in \mathcal{U}_i} \neg \mathsf{p}(\varphi^{(i-1)})) \rightarrow (\mathsf{p}(\psi^{(i-1)}) \leftrightarrow \mathsf{p}(\psi^{(i)})))\}.$$

The bi-implications in Figure 7 needs minor changes:

- replace every occurrence of (0) with $(i-1)$ and replace r with $r^{(i)}$;
- replace $\mathsf{Obj}(\mathcal{U})$ with Obj and replace $\mathsf{rhs}(\mathcal{U})$ with $\mathsf{rhs}(\mathcal{U}_i)$.

For all $i \in \{1, \ldots, n\}$, we use $C_{\mathsf{bi}}^{(i)}$ to denote the conjunction of bi-implications, where the left hand side ranges over $\mathsf{C}_{\mathsf{fle}}$. $\mathsf{P}_{\mathsf{fle}}$ is defined as before and for all $i \in \{1, \ldots, n\}$, the ABox $\mathcal{A}_{\mathsf{rel}}^{(i)}$ is defined as follows:

$$\mathcal{A}_{\mathsf{rel}}^{(i)} = \{\forall uw.C_{\mathsf{bi}}^{(i)}(a^*) \mid w \in \mathsf{P}_{\mathsf{fle}}\}.$$

Finally, we define

$$\mathcal{A}_n = \mathcal{A}_{\mathsf{ini}} \cup \mathcal{A}_{\mathsf{aux}} \cup \bigcup_{i=1}^{n} (\mathcal{A}_{\mathsf{r}}^{(i)} \cup \mathcal{A}_{\mathcal{U}_i} \cup \mathcal{A}_{\overline{\mathcal{U}_i}} \cup \mathcal{A}_{\mathsf{rel}}^{(i)}). \tag{7}$$

Similarly to the proof of Lemma 46, we can show that $(\cdots(\mathcal{A}_0 * \mathcal{U}_1) * \cdots * \mathcal{U}_n) \equiv^{\mathsf{P}} \mathcal{A}_n$. A close inspection of the construction of $\mathcal{A}_n$ shows the following upper bound of the computed projective update regardless of the coding of numbers:

Theorem 53. *For all ABoxes $\mathcal{A}_0$ and updates $\mathcal{U}_1, \ldots, \mathcal{U}_n$, if $\mathcal{A}_n$ is the ABox computed as in (7), then*

- $(\cdots(\mathcal{A}_0 * \mathcal{U}_1) * \cdots * \mathcal{U}_n) \equiv^{\mathsf{P}} \mathcal{A}_n$, *and*
- $|\mathcal{A}_n| = O((|\mathcal{U}_1| + \cdots + |\mathcal{U}_n|) \cdot (|\mathcal{A}| + |\mathcal{U}_1| + \cdots + |\mathcal{U}_n|)^3)$.

The above theorem gives a clue of how to compute projective updates of ABoxes in practice when new updates arise again. Suppose that $\mathcal{A}_n$ computed as in (7) is the result of updating $\mathcal{A}_0$ with $\mathcal{U}_1, \ldots, \mathcal{U}_n$. We assume that the structure of $\mathcal{A}_n$ is known, i.e., for every assertion φ in $\mathcal{A}_n$, we know which part of $\mathcal{A}_n$ (as specified in (7)) φ belongs to. Moreover, when $\mathcal{A}_n$ is computed, the sets $\mathsf{C}_{\mathsf{fle}}$, $\mathsf{R}_{\mathsf{fle}}$, and $\mathsf{P}_{\mathsf{fle}}$ are stored. If $\mathcal{A}_n$ needs to be updated with another update $\mathcal{U}_{n+1}$, then the new sets $\mathsf{C}'_{\mathsf{fle}}$, $\mathsf{R}'_{\mathsf{fle}}$, and $\mathsf{P}'_{\mathsf{fle}}$ are computed again according to $\mathcal{A}_0$ and $\mathcal{U}_1, \ldots, \mathcal{U}_{n+1}$. Based on the difference of the corresponding sets, we can construct $\mathcal{A}_{n+1}$ by modifying $\mathcal{A}_n$. Alternatively, $\mathcal{A}_{n+1}$ can be directly constructed from $\mathcal{A}_0$ and $\mathcal{U}_1, \ldots, \mathcal{U}_{n+1}$ as in (7). In both cases, the original ABox $\mathcal{A}_0$ and the history of updates which have been used to update $\mathcal{A}_0$ so far are stored.

If $\mathcal{A}_0$ is updated only with unconditional updates, then the idea of calculating the accumulated update $\mathcal{U}$ of $\mathcal{U}_1, \ldots, \mathcal{U}_n$ as in (5) can be applied to computing projective updates as well. The updated ABox obtained by updating the original ABox with $\mathcal{U}$ is then polynomial in the size of the input. As we have seen in Section 3.4, merging a sequence of unconditional updates into one update does not have an impact on the size of the computed logical updates whereas it does affect the size of the projective updates constructed as in Section 4.1. This is because in the construction of iterative projective updates, there are some subconcepts for which exponentially many bi-implications are introduced (cf. the subconcept $\exists r.C$ in Example 52), which is avoided by merging updates.

Chapter 5

DLs Having No Updates

In Chapter 3, we have shown how to compute logical updates with the help of nominals and the @ constructor. In Chapter 4, projective updates are constructed employing nominals. In this chapter, we show that those constructors are really necessary to build updated ABoxes, i.e., they cannot be built without them. More precisely, in Section 5.1, we show that projective updates are not expressible in DLs (between $\mathcal{ALC}$ and $\mathcal{ALCNI}^{@}$) without nominals.[1] The proof (with minor modifications) works also for approximate updates, i.e., describing approximate updates needs nominals as well. In Section 5.2, we show that extending DLs (between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}$) with nominals only is not sufficient to express approximate updates. Section 5.3 sums up the results about the expressivity of a DL and the existence of updates.

5.1 Nominals and Updates

In this section, we illustrate the relationship between nominals and existence of projective updates and approximate updates. In the constructions of both logical and projective updates, respectively presented in Chapter 3 and Chapter 4, nominals are explicitly employed, i.e., they are introduced no matter whether they appear in the original ABox. We now demonstrate that the introduction of nominals is somehow unavoidable. We start with showing that the DL $\mathcal{ALC}$ has no projective updates.

Lemma 54. *$\mathcal{ALC}$ does not have projective updates.*

Proof. Consider the ABoxes $\mathcal{A} = \{a : \exists r.A, r(b,a)\}$, $\mathcal{U} = \{\neg A(b)\}$ and

$$\mathcal{A}' = \{\neg A(b), r(b,a), a : \exists r.(A \sqcup \{b\})\}.$$

It is easy to see that $\mathcal{A}'$ is the logical update of $\mathcal{A}$ with $\mathcal{U}$, i.e., $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$. Assume now that there exists a projective update $\mathcal{B}$ of $\mathcal{A}$ with $\mathcal{U}$, i.e., $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}} \mathcal{B}$. We show that this leads to a contradiction.

Claim 1. There exists a model $\mathcal{I}$ of $\mathcal{B}$ such that $a^{\mathcal{I}} \notin (\exists r.A)^{\mathcal{I}}$.
The above claim holds since

[1] Here, $\mathcal{N}$ stands for *unqualified* number restriction. We will introduce its definition later on.

- there are models $\mathcal{I}$ of $\mathcal{A}'$ such that $a^{\mathcal{I}} \notin (\exists r.A)^{\mathcal{I}}$;
- for all interpretations $\mathcal{I}$, whether or not $a^{\mathcal{I}} \in (\exists r.A)^{\mathcal{I}}$ depends only on $\mathcal{I}_{\restriction \mathsf{S}}$, where $\mathsf{S} = ((\mathsf{N_C} \cup \mathsf{N_R} \cup \mathsf{N_I}) \setminus \mathsf{Sig}(\mathcal{B})) \cup \mathsf{Sig}(\mathcal{A}) \cup \mathsf{Sig}(\mathcal{U})$; and
- every $\mathcal{I}_{\restriction \mathsf{S}}$ can be extended to a model of $\mathcal{B}$ since $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}} \mathcal{B}$.

Claim 2.

(a) For all $n \geq 0$, $\mathcal{B} \models a : \exists r.(A \sqcup \exists r^{2n}.\top)$, where $\exists r^n.C$ denotes the n-fold nesting $\exists r.\cdots.\exists r.C$, with $\exists r^0.C = C$;

(b) For all $n \geq 0$, there does not exist a set of assertions

$$X = \{r(c_0, c_1), r(c_1, c_2), \ldots, r(c_k, c_{k+1}), \ldots, r(c_n, c_k)\}$$

such that $a = c_0$, $0 \leq k \leq n$, and $\mathcal{B} \models \varphi$ for all $\varphi \in X$.

To prove (a) observe that, for all $n \geq 0$, we have $\mathcal{A}' \models a : \exists r.(A \sqcup (\exists r^{2n}.\top))$. The crucial observation is that due to $a : \exists r.(A \sqcup \{b\}) \in \mathcal{A}'$, for every model $\mathcal{I}$ of $\mathcal{A}'$, there must be a $d \in \Delta^{\mathcal{I}}$ such that $(a^{\mathcal{I}}, d) \in r^{\mathcal{I}}$ and $d \in A^{\mathcal{I}}$, or $b^{\mathcal{I}} = d$. In the latter case, we get an r-cycle of length 2 between $a^{\mathcal{I}}$ and $b^{\mathcal{I}}$ since $r(b, a) \in \mathcal{A}'$. $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}} \mathcal{B}$ implies $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}}_{\mathcal{ALC}} \mathcal{B}$. Thus, together with $\mathcal{A}' \models a : \exists r.(A \sqcup (\exists r^{2n}.\top))$ and the fact that $a : \exists r.(A \sqcup \exists r^{2n}.\top)$ is an $\mathcal{ALC}$-assertion with $\mathsf{Sig}(a : \exists r.(A \sqcup \exists r^{2n}.\top)) \subseteq \mathsf{S}$, we have $\mathcal{B} \models a : \exists r.(A \sqcup \exists r^{2n}.\top)$.

For (b), assume that there exists such a set X. Observe that for all interpretations $\mathcal{I}$, $\mathcal{I} \models X$ if, and only if, the interpretation of X forms a "balloon" in $\mathcal{I}$ starting from $a^{\mathcal{I}}$ with an r-cycle starting at $c_k^{\mathcal{I}}$. Consider the concept $C = \exists r^k.A \sqcap \neg \exists r^{n+1}.A$. Based on this observation, it is not hard to see that $X \cup \{C(a)\}$ is inconsistent, which, together with that $\mathcal{B} \models \varphi$ for all $\varphi \in X$, yields that $\mathcal{B} \cup \{C(a)\}$ is also inconsistent. Thus, $M(\mathcal{B}) \models \neg C(a)$. Moreover, since $\mathsf{Sig}(C(a)) \subseteq \mathsf{S}$ and $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}}_{\mathcal{ALC}} \mathcal{B}$, and since $C(a)$ is an $\mathcal{ALC}$-assertion with $\mathsf{Sig}(C(a)) \subseteq \mathsf{S}$, we have $M(\mathcal{A}') \models \neg C(a)$, which implies that $\mathcal{A}' \cup \{C(a)\}$ is inconsistent. However, it is easy to construct a model of $\mathcal{A}' \cup \{C(a)\}$ and thus we have derived a contradiction.

Now take the model $\mathcal{I}$ of $\mathcal{B}$ in Claim 1. We unravel $\mathcal{I}$ into an interpretation $\mathcal{J}$ as follows:

- $\Delta^{\mathcal{J}} = \{d_0 \cdots d_k \mid k \geq 0 \wedge \forall i \in \{0, \ldots, k\} : d_i \in \Delta^{\mathcal{I}}\}$;
- for all $A \in \mathsf{N_C}$, $A^{\mathcal{J}} = \{d_0 \cdots d_k \in \Delta^{\mathcal{J}} \mid d_k \in A^{\mathcal{I}}\}$;
- for all role names s with $s \neq r$,

$$s^{\mathcal{J}} = s^{\mathcal{I}} \cup \{(d_0 \cdots d_k, d_0 \cdots d_{k+1}) \in \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} \mid (d_k, d_{k+1}) \in s^{\mathcal{I}}\},$$

and for the role name r,

$$\begin{aligned} r^{\mathcal{J}} \quad = \quad & \{(c_1^{\mathcal{I}}, c_2^{\mathcal{I}}) \mid \exists c_1, c_2 \in \mathsf{N_I} : \mathcal{B} \models r(c_1, c_2)\} \cup \\ & \{(d_0 \cdots d_k, d_0 \cdots d_{k+1}) \in \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} \mid (d_k, d_{k+1}) \in r^{\mathcal{I}}\}; \end{aligned}$$

- for all $a \in \mathsf{N_I}$, $a^{\mathcal{J}} = a^{\mathcal{I}}$.

Notice that $r^{\mathcal{J}}$ is well-defined since given an ABox $\mathcal{B}$ and a role name r, the set $\{r(c_1, c_2) \mid \exists c_1, c_2 \in \mathsf{N_I} : \mathcal{B} \models r(c_1, c_2)\}$ is decidable. By induction on the structure of C, it is easy to see that the following claim holds:

Claim 3. For all $\mathcal{ALC}$-concepts C and all $w = d_0 \cdots d_k \in \Delta^{\mathcal{J}}$, we have $w \in C^{\mathcal{J}}$ iff $d_k \in C^{\mathcal{I}}$.

By Claim 3 and since $\mathcal{I} \not\models a : \exists r.A$, we have $\mathcal{J} \not\models a : \exists r.A$. Claim 3 also implies that $\mathcal{J}$ is a model of $\mathcal{B}$:

- For all concept assertions $C(c) \in \mathcal{B}$, we have $\mathcal{I} \models C(c)$. Thus, $\mathcal{J} \models C(c)$ by Claim 3.
- For all positive role assertions $\varphi \in \mathcal{B}$, we have, by definition of $\mathcal{J}$, $\mathcal{J} \models \varphi$.
- Let $\neg r(c_1, c_2) \in \mathcal{B}$ be a negated role assertion. Then, $\mathcal{I} \models \mathcal{B}$ implies $\mathcal{I} \models \neg r(c_1, c_2)$. Assume that $\mathcal{J} \not\models \neg r(c_1, c_2)$, i.e., $\mathcal{J} \models r(c_1, c_2)$. It follows from the definition of $\mathcal{J}$ that $\mathcal{J} \models \varphi$ implies $\mathcal{I} \models \varphi$, for all positive role assertions φ. Thus, $\mathcal{I} \models r(c_1, c_2)$, which is a contradiction.

We define the depth $d(C)$ of an $\mathcal{ALC}$-concept C as the nesting depth of existential and value restrictions in C. The depth $d(\mathcal{B})$ of an ABox $\mathcal{B}$ is defined as $\max\{d(C) \mid C(c) \in \mathcal{B}\}$ if $\mathcal{B}$ contains a concept assertion; $d(\mathcal{B}) = 0$ otherwise. In the next step, we further modify $\mathcal{J}$ by "cutting off" all paths in $\Delta^{\mathcal{J}}$ at length $d(\mathcal{B})$. Thus, let $\Delta^{\mathcal{J}'} = \{d_0 \cdots d_k \in \Delta^{\mathcal{J}} \mid k \leq d(\mathcal{B})\}$, let $B^{\mathcal{J}'}$ and $s^{\mathcal{J}'}$ be the restrictions of $B^{\mathcal{J}}$ and $s^{\mathcal{J}}$ to $\Delta^{\mathcal{J}'}$ for all $B \in \mathsf{N_C}$ and $s \in \mathsf{N_R}$, and let $c^{\mathcal{J}'} = c^{\mathcal{J}}$ for all $c \in \mathsf{N_I}$. The interpretation of individual names is well-defined since for all $c \in \mathsf{N_I}$, $c^{\mathcal{J}}$ is of length 0 (and thus not dropped). It is not hard to prove the following claim by induction on the structure of C:

Claim 4. For all $\mathcal{ALC}$-concepts C with $d(C) = i \leq d(\mathcal{B})$, and all $w = d_0 \cdots d_k \in \Delta^{\mathcal{J}}$ with $k \leq d(\mathcal{B}) - i$, we have $w \in C^{\mathcal{J}}$ iff $w \in C^{\mathcal{J}'}$.

It follows from the definition of $\mathcal{J}'$ that $a^{\mathcal{J}'} = a^{\mathcal{J}}$, $A^{\mathcal{J}'} \subseteq A^{\mathcal{J}}$ and $r^{\mathcal{J}'} \subseteq r^{\mathcal{J}}$. Thus, $\mathcal{J} \not\models a : \exists r.A$ implies that $\mathcal{J}' \not\models a : \exists r.A$. Additionally, $\mathcal{J} \models \mathcal{B}$ implies that $\mathcal{J}'$ is a model of $\mathcal{B}$:

- As an immediate consequence of the construction of $\mathcal{J}'$, all (possibly negated) role assertions $\varphi \in \mathcal{B}$ are not invalidated.
- Let $C(a) \in \mathcal{B}$ be a concept assertion. Then, $d(C) \leq d(\mathcal{B})$ and there exists a $d_0 \in \Delta^{\mathcal{J}'}$ such that $a^{\mathcal{I}} = d_0$. Hence, $\mathcal{J}' \models C(a)$ by Claim 4 and since $\mathcal{J} \models C(a)$.

Since $\mathcal{J}' \models \mathcal{B}$, it follows from Claim 2(a) that $\mathcal{J}' \models a : \exists r.(A \sqcup \exists r^{2n}.\top)$ for all $n \geq 0$. Additionally, $\mathcal{J}' \not\models a : \exists r.A$ implies that $\mathcal{J}' \models a : \exists r^{2n+1}.\top$ for all $n \geq 0$. Thus, by the construction of $\mathcal{J}'$ from $\mathcal{J}$, there must exist $d_0, d_1, \ldots, d_n \in \Delta^{\mathcal{I}}$ such that $a^{\mathcal{J}'} = d_0$, $(d_i, d_{i+1}) \in r^{\mathcal{J}'}$ for all $i < n$, and $(d_n, d_k) \in r^{\mathcal{J}'}$ for some $k \leq n$. However, this yields that a set X with the properties of Claim 2(b) exists. ❑

The above proof can be extended to show that every DLs between $\mathcal{ALC}$ to $\mathcal{ALCNI}^{@}$ does not have projective updates, where the letter $\mathcal{N}$ in the name of a DL stands for unqualified number restrictions. Compared to qualified number restrictions, every at-least (at-most, respectively) unqualified number restriction is of the form $(\geqslant n\ r\ \top)$ $((\leqslant n\ r\ \top)$, respectively) with n a natural number and r a role, i.e., only the top concept can appear in number restrictions.

The crucial step in making the proof work for the DL $\mathcal{ALCNI}^{@}$ (and its fragments) is to construct $\mathcal{J}$ by unraveling $\mathcal{I}$ such that Claim 3 holds for all $\mathcal{ALCNI}^{@}$-concepts. In addition, by extending the notion of the depth $d(C)$ to $\mathcal{ALCNI}^{@}$-concepts C as the nesting depth of number, existential, and value restrictions in C,[2] the rest of the proof can be easily gone through.

In order to extend the unraveling technique to $\mathcal{ALCNI}^{@}$, we modify the proof in two places:

1. The model $\mathcal{I}$ of $\mathcal{B}$ in Claim 1 requires additionally that for all $d \in \Delta^{\mathcal{I}}$, the sets $\{(d, d') \mid (d, d') \in r^{\mathcal{I}}\}$ and $\{(d', d) \mid (d', d) \in r^{\mathcal{I}}\}$ are infinite. Such a model exists for the same reason as before: $\mathcal{A}'$ has such models $\mathcal{I}$ and the above requirement only restricts on $\mathcal{I}_{\restriction \mathsf{S}}$.

2. $\mathcal{J}$ is built from $\mathcal{I}$ as follows:

   - Let $\Delta^{\mathcal{J}}$ be the set of all words $w = d_0 s_0 d_1 s_1 \cdots s_{k-1} d_k$ such that
     - $k \geq 0$;
     - $d_0, \ldots, d_k \in \Delta^{\mathcal{I}}$;
     - $s_0 \in \{r, r^-\}$;
     - $s_1, \ldots, s_{k-1}$ are roles;
     - for $i < k$, $(d_i, d_{i+1}) \in s_i^{\mathcal{I}}$ and if $s_i = s_{i+1}^-$, then $d_i \neq d_{i+2}$.
   - $B^{\mathcal{J}} = \{d_0 \cdots d_k \in \Delta^{\mathcal{J}} \mid d_k \in B^{\mathcal{I}}\}$ for all $B \in \mathsf{N_C}$;
   - for every $s \in \mathsf{N_R}$ with $s \neq r$,

     $$\begin{aligned} s^{\mathcal{J}} \quad = \quad & s^{\mathcal{I}} \cup \{(d_0 \cdots d_k, d_0 \cdots s d_{k+1}) \in \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} \mid (d_k, d_{k+1}) \in s^{\mathcal{I}}\} \cup \\ & \{(d_0 \cdots s^- d_{k+1}, d_0 \cdots d_k \in \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} \mid (d_{k+1}, d_k) \in s^{\mathcal{I}}\}. \end{aligned}$$

   - for the role name r,

     $$\begin{aligned} r^{\mathcal{J}} \quad = \quad & \{(c_1^{\mathcal{I}}, c_2^{\mathcal{I}}) \mid \exists c_1, c_2 \in \mathsf{N_I} : \mathcal{B} \models r(c_1, c_2)\} \cup \\ & \{(d_0 \cdots d_k, d_0 \cdots r d_{k+1}) \in \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} \mid (d_k, d_{k+1}) \in r^{\mathcal{I}}\} \cup \\ & \{(d_0 \cdots r^- d_{k+1}, d_0 \cdots d_k) \in \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}} \mid (d_{k+1}, d_k) \in r^{\mathcal{I}}\}. \end{aligned}$$

   - $a^{\mathcal{J}} = a^{\mathcal{I}}$ for all $c \in \mathsf{N_I}$.

The next lemma shows that Claim 3 indeed holds for all $\mathcal{ALCNI}^{@}$-concepts:

[2] A more careful definition would "reset" the depth when encountering the "@" operator, but this is not necessary for our purposes.

Lemma 55. *Consider the ABox $\mathcal{B}$ in the proof of Lemma 54. Let $\mathcal{I}$ be a model of $\mathcal{B}$ such that for all $d \in \Delta^{\mathcal{I}}$, the sets $\{(d,d') \mid (d,d') \in r^{\mathcal{I}}\}$ and $\{(d',d) \mid (d',d) \in r^{\mathcal{I}}\}$ are infinite and $\mathcal{J}$ the interpretation obtained from $\mathcal{I}$ by the above construction. Then, for all $\mathcal{ALCNI}^{@}$-concepts C and all $w = d_0 \cdots d_k \in \Delta^{\mathcal{J}}$, we have $w \in C^{\mathcal{J}}$ iff $d_k \in C^{\mathcal{I}}$.*

Proof. We proof this lemma by induction on the structure of C. If $C = A$ for some concept name A, then it follows from the definition of $A^{\mathcal{J}}$ that $w \in A^{\mathcal{J}}$ iff $d_k \in A^{\mathcal{I}}$. The cases $C = \top$ and $C = \bot$ hold trivially. By I.H., the cases $C = \neg D$, $C = @_a D$, $C = D \sqcap E$, $C = D \sqcup E$ hold.

Consider the role name r. Then $(\geqslant n\ r\ \top)^{\mathcal{J}} = \Delta^{\mathcal{J}}$ and $(\geqslant n\ r\ \top)^{\mathcal{I}} = \Delta^{\mathcal{I}}$ since for all $d \in \Delta^{\mathcal{I}}$, the sets $\{(d,d') \mid (d,d') \in r^{\mathcal{I}}\}$ and $\{(d',d) \mid (d',d) \in r^{\mathcal{I}}\}$ are infinite. Thus, $w \in (\geqslant n\ r\ \top)^{\mathcal{J}}$ iff $d_k \in (\geqslant n\ r\ \top)^{\mathcal{I}}$.

Consider $C = (\geqslant n\ s\ \top)$ with a role name s and $s \neq r$. "$\Rightarrow$": $w \in (\geqslant n\ s\ \top)^{\mathcal{J}}$ implies that there are pairwise distinct $w_1, \ldots, w_n$ such that $(w, w_i) \in s^{\mathcal{J}}$ for all i with $1 \leq i \leq n$.

- If $k = 0$, then for all i with $1 \leq i \leq n$, $w_i \in \Delta^{\mathcal{I}}$ and $(d_k, w_i) \in r^{\mathcal{I}}$. Thus, $d_k \in (\geqslant n\ s\ \top)^{\mathcal{I}}$.

- If $k \geq 1$, then w is of the form $d_0 \cdots d_{k-1} s_{k-1} d_k$. For all i with $1 \leq i \leq n$, w_i is either of the form $d_0 r \cdots d_k s d_{p_i}$ with $(d_k, d_{p_i}) \in s^{\mathcal{I}}$ or of the form $d_0 \cdots d_{k-1}$ with $(d_k, d_{k-1}) \in s^{\mathcal{I}}$. Since $w_1, \ldots, w_n$ are pairwise distinct, for all w_i, w_j such that $1 \leq i, j \leq n$, $i \neq j$, and w_i and w_j are of the first form, $d_{p_i} \neq d_{p_j}$. If some w_i with $1 \leq i \leq n$ is of the second form, then $s_{k-1}^{-} = s$ and for all w_i those are of the second form, $d_{p_i} \neq d_{k-1}$. Therefore, $d_k \in (\geqslant n\ s\ \top)^{\mathcal{I}}$.

"$\Leftarrow$": $d_k \in (\geqslant n\ s\ \top)^{\mathcal{I}}$ implies that there are pairwise distinct $e_1, \ldots, e_n$ such that $(d_k, e_i) \in s^{\mathcal{I}}$ for all i with $1 \leq i \leq n$.

- If $k = 0$, then for all i with $1 \leq i \leq n$, $(d_k, e_i) \in r^{\mathcal{J}}$. Thus, $d_k \in (\geqslant n\ s\ \top)^{\mathcal{J}}$.

- If $k \geq 1$, then w is of the form $d_0 \ldots d_{k-1} s_{k-1} d_k$. For all i with $1 \leq i \leq n$, if $s_{k-1}^{-} \neq s$ or $d_{k-1} \neq e_i$, then $(w, w s e_i) \in r^{\mathcal{J}}$; Otherwise, $(w, d_0 \cdots d_{k-1}) \in r^{\mathcal{J}}$. In addition, for all i with $1 \leq i \leq n$, there is at most one e_i with $e_i = d_{k-1}$. Therefore, $w \in (\geqslant n\ s\ \top)^{\mathcal{J}}$.

The cases $C = (\geqslant n\ s\ \top)$ and $C = (\leqslant n\ s\ \top)$ with an inverse role s can be proved analogously.

Consider $C = \exists s.D$.[3] "$\Rightarrow$": $w \in (\exists s.D)^{\mathcal{J}}$ implies that there exists w' such that $w' \in D^{\mathcal{J}}$ and $(w, w') \in s^{\mathcal{J}}$.

If s is a role name with $s \neq r$ then

- if $k = 0$, then $w' = d'$ for some $d' \in \Delta^{\mathcal{I}}$ and $(d_k, d') \in s^{\mathcal{I}}$. By I.H., $w' \in D^{\mathcal{J}}$ implies $d' \in D^{\mathcal{I}}$. Thus, $d_k \in (\exists s.D)^{\mathcal{I}}$.

[3] Existential and value restrictions cannot be expressed by unqualified number restrictions. We have to treat them explicitly.

- If $k \geq 1$, then w is of the form $d_0 \cdots d_{k-1} s_{k-1} d_k$. Thus, w' is either of the form $d_0 r \cdots d_k s d_{k+1}$ with $(d_k, d_{k+1}) \in s^{\mathcal{I}}$ or of the form $d_0 \cdots d_{k-1}$ with $(d_k, d_{k-1}) \in s^{\mathcal{I}}$. If w' is of the first form, then by I.H., $w' \in D^{\mathcal{J}}$ implies $d_{k+1} \in D^{\mathcal{I}}$. If w' is of the second form, then by I.H., $w' \in D^{\mathcal{J}}$ implies $d_{k-1} \in D^{\mathcal{I}}$. Overall, $d_k \in (\exists s.D)^{\mathcal{I}}$.

Consider $C = \exists r.D$.

- If $k = 0$, then w' is either of the form d' such that $d_k = c_1^{\mathcal{I}}$, $d' = c_2^{\mathcal{I}}$ for some $d' \in \Delta^{\mathcal{I}}$, $c_1, c_2 \in \mathsf{N_I}$, and $\mathcal{B} \models r(c_1, c_2)$, or of the form $d_k r d'$ with $(d_k, d') \in r^{\mathcal{I}}$. In the first case, $\mathcal{I} \models \mathcal{B}$ implies that $(d_k, d') \in r^{\mathcal{I}}$. In both cases, by I.H., $w' \in D^{\mathcal{J}}$ implies $d' \in D^{\mathcal{I}}$. Thus, $d_k \in (\exists r.D)^{\mathcal{I}}$.

- $k \geq 1$: This case can be shown similarly to the case that $s \neq r$ and $k \geq 1$.

The case that s is an inverse role can be shown analogously.

"$\Leftarrow$": $d_k \in (\exists s.D)^{\mathcal{I}}$ implies that there exists $e_j \in \Delta^{\mathcal{I}}$ such that $(d_k, e_j) \in s^{\mathcal{I}}$ and $e_j \in D^{\mathcal{I}}$.

Suppose that s is a role name with $s \neq r$.

- If $k = 0$, then $(d_k, e_j) \in s^{\mathcal{J}}$. By I.H., $e_j \in D^{\mathcal{I}}$ implies $e_j \in D^{\mathcal{J}}$. Thus, $d_k \in (\exists s.D)^{\mathcal{J}}$.

- If $k \geq 1$, then w is of the form $d_0 \dots d_{k-1} s_{k-1} d_k$. If $s_{k-1}^- \neq s$ or $d_{k-1} \neq e_j$, then $(w, w s e_j) \in s^{\mathcal{J}}$. Otherwise, $(w, d_0 \cdots d_{k-1}) \in s^{\mathcal{J}}$. In the first case, by I.H., $e_j \in D^{\mathcal{I}}$ implies $w s e_j \in D^{\mathcal{J}}$. In the second case, by I.H., $e_j \in D^{\mathcal{I}}$ implies $d_0 \cdots d_{k-1} \in D^{\mathcal{J}}$. Therefore, $w \in (\exists s.D)^{\mathcal{J}}$.

Consider $C = \exists r.D$.

- If $k = 0$, then $(d_k, d_k r e_j) \in r^{\mathcal{J}}$. By I.H., $e_j \in \Delta^{\mathcal{I}}$ implies $d_k r e_j \in \Delta^{\mathcal{J}}$. Thus, $d_k \in (\exists r.D)^{\mathcal{J}}$.

- $k \geq 1$: This case can be shown similarly to the case $s \neq r$ and $k \geq 1$.

The case that s is an inverse role can be shown analogously. The case $C = \forall s.D$ can be proved similarly to the case $C = \exists s.D$. ❑

The unraveling of a model of an ABox described above does not preserve the number of r-successors. The enhanced Claim 1 which requires that every object in the domain has infinitely many r-successors ensures correctness of Claim 3 for unqualified number restriction. Nevertheless, this does not suffice to make the claim true for qualified number restrictions since those r-successors also have to satisfy the respective qualifications. Following the methodology in the proof of Lemma 55, either an enhanced claim or a different example of $\mathcal{A}$ and $\mathcal{U}$ is necessary to show the non-existence of projective updates for arbitrary $\mathcal{ALCQI}^{@}$-ABoxes. This is left as an open problem, although we believe that nominals are necessary for expressing projective updates.

Theorem 56. *Let $\mathcal{L}$ be a DL between $\mathcal{ALC}$ and $\mathcal{ALCNI}^{@}$. Then $\mathcal{L}$ does not have projective updates.*

It follows from Lemma 55 that for some DLs, Boolean ABoxes are strictly more expressive than the @ constructor, e.g., there are Boolean $\mathcal{ALC}$-ABoxes for which no equivalent non-Boolean $\mathcal{ALC}^{@}$-ABox exists.

Corollary 57. *There exists no non-Boolean $\mathcal{ALCNI}^{@}$-ABox that is equivalent to the Boolean $\mathcal{ALC}$-ABox $\mathcal{B} = \{\neg A(b), r(b,a), a : \exists r.A \vee r(a,b)\}$.*

Proof. We can see that $\mathcal{B}$ is equivalent to the ABox $\mathcal{A}'$ in Lemma 56 since the Boolean assertion $a : \exists r.A \vee r(a,b)$ apparently has the same set of models as the ABox assertion $a : \exists r.(A \sqcup \{b\})$. Assume that there is a non-Boolean $\mathcal{ALCNI}^{@}$-ABox $\mathcal{B}'$ which is equivalent to $\mathcal{B}$. Then, we have $\mathcal{A}' \equiv \mathcal{B}'$, which, together with $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$, yields $\mathcal{A} * \mathcal{U} \equiv \mathcal{B}'$. Hence, $\mathcal{A} * \mathcal{U} \equiv^{\mathsf{P}} \mathcal{B}'$, which leads to a contradiction. ❑

As we have seen, the ABox $\mathcal{A}$ and the update $\mathcal{U}$ in Lemma 54 are used to show that $\mathcal{L}$ does not have projective updates for any $\mathcal{L}$ between $\mathcal{ALC}$ and $\mathcal{ALCNI}^{@}$. In the proof, only Claim 1 and Claim 2 exploit the assumption that $\mathcal{B}$ is a projective update of $\mathcal{A}$ with $\mathcal{U}$. In particular, Claim 1 also holds if $\mathcal{B}$ is an approximate update since by 3 of Lemma 23, every model of the logical update is also a model of an approximate update. Claim 2 also holds even if $\mathcal{B}$ is an approximate projective update. Thus, we obtain the following theorem:

Theorem 58. *Let $\mathcal{L}$ be a DL between $\mathcal{ALC}$ and $\mathcal{ALCNI}^{@}$. Then $\mathcal{L}$ does not have approximate updates.*

We have shown that nominals are crucial to describe both projective updates and approximate updates. The necessity of nominals for expressing approximate projective updates is still open. The main difficulty of extending the proof of Lemma 54 is to prove Claim 1 if $\mathcal{B}$ is an approximate projective update of $\mathcal{A}$ with $\mathcal{U}$.

5.2 The @ Constructor and Updates

In the last section, we investigated the relationship between nominals and the existence of updates. More specifically, every DL $\mathcal{L}$ between $\mathcal{ALC}$ and $\mathcal{ALCNI}^{@}$ has neither projective updates nor approximate updates. In Chapter 4, we have seen that adding nominals to $\mathcal{L}$ leads to existence of projective updates. This section is dedicated to showing that only adding nominals is not enough to express approximate updates.

Theorem 59. *Let $\mathcal{L}$ be a DL between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}$. Then $\mathcal{L}$ does not have approximate updates.*

Proof. We start the proof by exhibiting an ABox $\mathcal{A}$ and an update $\mathcal{U}$ such that no approximate update of $\mathcal{A}$ with $\mathcal{U}$ w.r.t. $\mathcal{L}$ exists. Let $\mathcal{A} = \{a : \exists r.A\}$, $\mathcal{U} = \{\neg r(a,b)\}$ and

$$\mathcal{A}' = \{a : \exists r.A \sqcup @_b A, \neg r(a,b)\}.$$

Then, it is easy to see that $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$.

Note that the logical update $\mathcal{A}'$ is an $\mathcal{ALC}^{@}$-ABox, but not an $\mathcal{ALCO}$-ABox. Our aim is to show that there is no $\mathcal{L}$-ABox $\mathcal{B}$ with $\mathcal{B} \equiv_{\mathcal{L}} \mathcal{A}'$, which implies that there is no $\mathcal{L}$-ABox $\mathcal{B}$ such that $\mathcal{A} * \mathcal{U} \equiv_{\mathcal{L}} \mathcal{B}$ by 1 of Lemma 23.

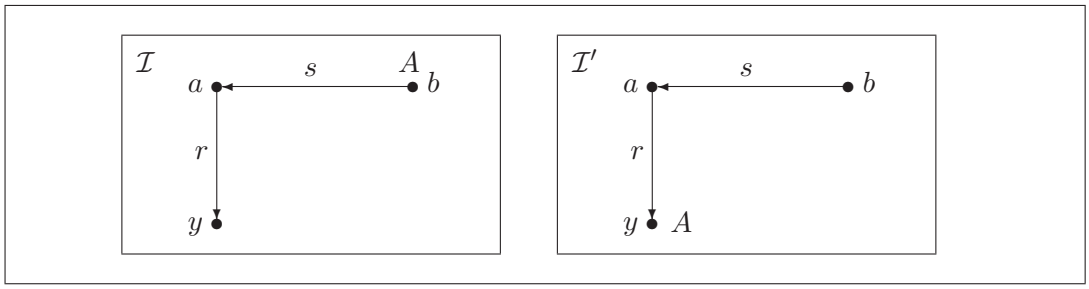


Figure 10: The interpretations $\mathcal{I}$ and $\mathcal{I}'$.

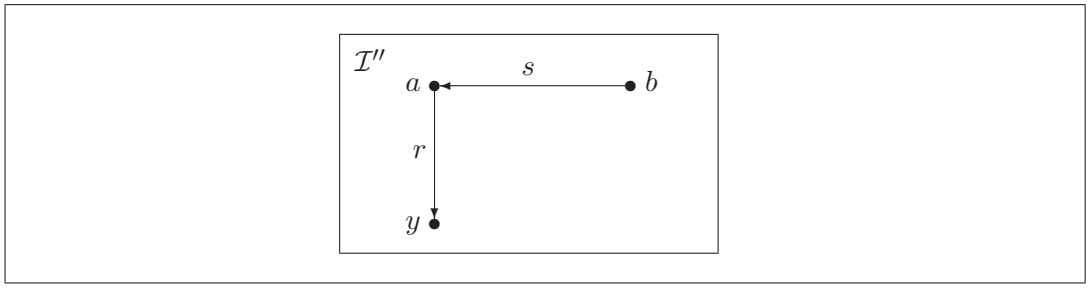


Figure 11: The interpretation $\mathcal{I}''$.

Assume to the contrary that such a $\mathcal{B}$ exists. Choose a role name s that does not occur in $\mathcal{A}'$ and $\mathcal{B}$. Now consider the interpretations $\mathcal{I}$ and $\mathcal{I}'$ displayed in Figure 10. We assume that the individual names a and b are mapped to the objects of the same name as shown in the figure. We also assume that in addition to the points depicted there is an infinite set of points interpreting the individual names distinct from a and b. On these additional points the concept and role names are interpreted as the empty set. (The additional points are required to define interpretations satisfying the UNA.) Clearly, $\mathcal{I}$ and $\mathcal{I}'$ are models of $\mathcal{A}'$. Therefore, it follows from 3 of Lemma 23 that they are models of $\mathcal{B}$. Consider the additional interpretation $\mathcal{I}''$ in Figure 11 (again there are additional points interpreting the remaining individual names in the same way as in $\mathcal{I}$ and $\mathcal{I}'$). We show the following:

Claim 1. $\mathcal{I}'' \not\models \mathcal{B}$.
Proof of Claim 1. Assume $\mathcal{I}'' \models \mathcal{B}$. Define

$$C = \neg A \sqcap \exists s.(\{a\} \sqcap \forall r.\neg A).$$

Clearly, $\mathcal{I}'' \models C(b)$. This implies that $\mathcal{I}'' \not\models \neg C(b)$, which, together with $\mathcal{I}'' \models \mathcal{B}$, yields that $\mathcal{B} \not\models \neg C(b)$. Since $\mathcal{B} \equiv_{\mathcal{L}} \mathcal{A}'$, we know that $\mathcal{A}' \not\models \neg C(b)$. This means that $\mathcal{A}' \cup \{C(b)\}$ is consistent. Analyzing $\mathcal{A}'$ and C, it is readily checked that this is not true. This finishes the proof of Claim 1.

Claim 2. $\mathcal{I}'' \models \mathcal{B}$
Proof of Claim 2. We know that $\mathcal{I} \models \mathcal{B}$ and $\mathcal{I}' \models \mathcal{B}$. Let $\varphi \in \mathcal{B}$. First assume that φ is a (possibly negated) role assertion. Since $\mathcal{I}$ and $\mathcal{I}''$ interpret role names

	$\mathcal{L}_1$	$\mathcal{L}_1\mathcal{O}$	$\mathcal{L}_1^@$	$\mathcal{L}_1\mathcal{O}^@$	$\mathcal{L}_2$	$\mathcal{L}_2\mathcal{O}$	$\mathcal{L}_2^@$	$\mathcal{L}_2\mathcal{O}^@$
$\equiv$	X	X	X	E	?	X	?	E
$\equiv_{\mathcal{L}}$	X	X	X	E	?	X	?	E
$\equiv^{\mathsf{P}}$	X	P	X	P	?	P	?	P
$\equiv^{\mathsf{P}}_{\mathcal{L}}$	?	P	?	P	?	P	?	P

Table 1: Expressivity and updates.

and individual names in the same way, we know that for all such assertions φ, $\mathcal{I} \models \varphi$ iff $\mathcal{I}'' \models \varphi$. Thus, $\mathcal{I} \models \mathcal{B}$ implies $\mathcal{I}'' \models \varphi$. Now let φ be a concept assertion $C(c)$. First assume that $c \neq b$. Then the part of $\mathcal{I}''$ that is reachable from c via any path composed of roles in which s does not occur is identical to the corresponding part of $\mathcal{I}$. Since s does not occur in $\mathcal{B}$, s does not occur in C. Since $\mathcal{I} \models \mathcal{B}$, and for C an $\mathcal{ALCQIO}$-concept in which s does not occur, the truth of assertions $C(c)$ in a model $\mathcal{J}$ only depends on the set of points reachable from $c^{\mathcal{J}}$ by the forementioned paths of roles, we get $\mathcal{I}'' \models C(c)$. Now let $c = b$. Then the part of $\mathcal{I}''$ that is reachable from b via any path of roles in which s does not occur is identical to the corresponding part of $\mathcal{I}'$. Thus, we can argue similarly to the case $c \neq b$ to show that $\mathcal{I}'' \models C(c)$. This completes the proof of Claim 2.

As Claim 1 and 2 contradict each other, no such $\mathcal{B}$ exists. ❑

As we can see in the above proof, nominals are used in the concept C and thus this proof cannot be used for showing that $\mathcal{L}$ does not have approximate updates for a DL $\mathcal{L}$ between $\mathcal{ALCQ}$ and $\mathcal{ALCQI}^@$.

It follows from Theorem 35 that, for DLs with both nominals and the @ constructor, approximate updates exist since logical updates are also approximate updates. Whether or not one can compute more succinct approximate updates is still an open problem.

5.3 Expressiveness vs. Updates — A Summary

Summing up the results about the expressivity of a DL and the existence of its updates so far, we obtain Table 1, where $\mathcal{L}_1$ is a DL in $\{\mathcal{ALC}, \mathcal{ALCN}, \mathcal{ALCI}, \mathcal{ALCNI}\}$, $\mathcal{L}_2$ is a DL in $\{\mathcal{ALCQ}, \mathcal{ALCQI}\}$, the letters P and E respectively stand for existence of updates polynomial and exponential in the size of the original ABox and the update, the letter X means the DL does not have the corresponding updates, and the symbol ? means that it is still open whether the DL has the corresponding updates.

Adding only nominals, approximate updates are not yet expressible while even logical updates are expressible with the addition of the @ constructor. On the one hand, since the exponential blowup cannot be avoided unless PTIME =NC as shown in Theorem 43, the @ constructor can be viewed as the missing constructor in standard DLs to describe logical updates. On the other hand, the @ constructor is oversized for covering approximate updates. It is unclear whether there exists a DL that offers adequate expressivity for approximate updates but not for logical updates. Whether

	$\mathcal{L}$	$\mathcal{LO}$	$\mathcal{L}^{@}$	$\mathcal{LO}^{@}$
$\equiv$	?	2E	?	E
$\equiv_{\mathcal{L}}$	?	2E	?	E
$\equiv^{\mathsf{P}}$	?	P	?	P
$\equiv^{\mathsf{P}}_{\mathcal{L}}$	?	P	?	P

Table 2: Expressivity and updates on Boolean ABoxes.

a polynomial construction of approximate updates exists for DLs with both nominals and the @ constructor is left as an open problem.

It is clear that adding nominals leads to the existence of projective updates whose size is even polynomially bounded. This also implies the existence of approximate projective updates. We believe that all the ? in Table 1 should be X, i.e, without nominals even the weakest kind of updates is not expressible.

We have introduced the notions that a DL $\mathcal{L}$ has logical updates and projective updates on Boolean ABoxes. Approximate updates and approximate projective updates on Boolean ABoxes are defined in a straightforward way. Considering updates on Boolean ABoxes, we collect the results from Theorem 38 and Theorem 49 and present them in Table 2, where $\mathcal{L}$ is a DL between $\mathcal{ALC}$ and $\mathcal{ALCQI}$ and 2E means that the size of constructed update is double exponential in the size of the whole input. Except that the lower bound of the size of logical updates for $\mathcal{LO}^{@}$ is known, none of the other constructions has been proven to be optimal.

In [Bon07], the notions of uniform ABox interpolants and uniform Boolean ABox interpolants were employed to establish the relationship between approximate updates and approximate projective updates. It is shown there that if a DL $\mathcal{L}$ has uniform ABox (Boolean ABox, respectively) interpolation, then $\mathcal{L}$ has approximate updates (on Boolean ABoxes, respectively) iff $\mathcal{L}$ has approximate projective updates (on Boolean ABoxes, respectively). This opens us another way to investigate the existence of updates. For instance, it is known that $\mathcal{ALC}$ does not have approximate updates. If $\mathcal{ALC}$ has ABox uniform interpolation, then $\mathcal{ALC}$ does not have approximate projective updates either. Unfortunately, Bong has shown that $\mathcal{ALC}$ does not have ABox uniform interpolation. The followings have also been shown by Bong:

(a) $\mathcal{ALC}$ does not have approximate updates on Boolean ABoxes, which strengthens the result in [LLMW06c] that $\mathcal{ALC}$ does not have logical updates on Boolean ABoxes.

(b) $\mathcal{ALC}$ has Boolean ABox uniform interpolation and thus the non-existence of approximate projective updates for $\mathcal{ALC}$ on Boolean ABoxes is obtained.

This leads us to an obvious question: Can we extend those proofs of (a) and (b) to more expressive DLs as we did in Lemma 55? This is left as future work. Another question is whether the proof for (a) can be directly used for showing that $\mathcal{ALC}$ does not have approximate projective updates (on non-Boolean ABoxes). The answer is

negative because in the proof of (a) a Boolean ABox assertion is constructed as a logical consequence of updated models whereas approximate projective updates on non-Boolean ABoxes preserve only non-Boolean assertions.

Chapter 6

Experimental Results

In this chapter, we present experimental results. In Section 6.1, we compare the size and the constructing time of logical updates and projective updates achieved from the implementations based on the computation introduced in the previous chapters. We explore how well those algorithms work in the experiments. As we have seen in Chapter 3, both the @ constructor and Boolean assertions are used to compute logical updates. However, DL reasoners support neither of them directly. In Section 6.2, three reasoning approaches handling inference problems of such Boolean ABoxes are described. We compare and analyze their performance. We also compare reasoning with logical updates to reasoning with projective updates. As described in Chapter 2, the projection problem can be solved via progression or regression. Updating ABoxes is an application of progression in DLs. The projection problem in DLs was also solved by a reduction to the consistency problem of ABoxes w.r.t. acyclic TBoxes, which is similar to regression [BLM$^+$05]. In Section 6.3, we present some experimental results of the implementation based on this reduction.

All the implementations in this chapter are written in Java unless it is explicitly pointed out otherwise. The size of the input and output complies with the definitions in Chapter 2. The unit of time measurement is millisecond (ms). In order to confine the execution environment and hence to induce sensible comparison, the experiments were performed on the same Linux testing server sitting in a temperature-controlled room. The server was equipped with a 3.16GHz Intel Core 2 Duo CPU and 4GB of memory. The Java runtime environment in the server is version 1.6.

6.1 The Comparison of Logical And Projective Updates

In this section, we compare the size and the constructing time of logical updates and projective updates achieved from the implementations based on the computation introduced in the previous chapters.

In order to test the performance of our implementation on a sufficiently large set of data, we implemented a random generator of $\mathcal{ALC}$-ABox assertions in which several parameters are used to control the shape of generated assertions: the number d of nesting roles in a concept assertion, the number ncs of the constructors in a concept

d	ncs	nc	nr	ni	pr	na
1 – 8	1 – 30	1 – 20	1 – 10	1 – 10	30	0 – 23

Table 3: Parameters of the random ABox generator.

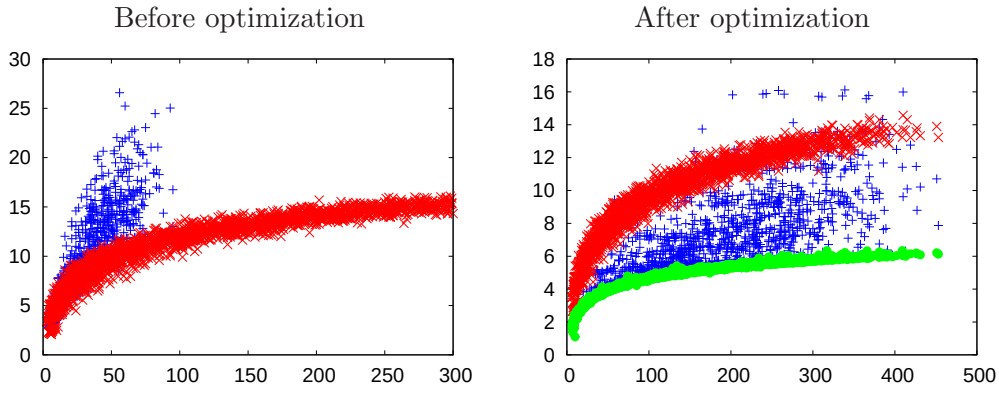


Figure 12: The size of logical and projective updates.

assertion, the numbers nc, nr, and ni of concept names, role names, and individual names in a concept assertion, the probability pr% of generating a role assertion, and the number na of assertions. The parameters used for generating testing data are displayed in Table 3. An unconditional update is associated to each generated ABox. In every update, the probability of generating a role assertion is 30%, the probability of using a concept, role, and individual name which does not appear in the ABox is 10%, and the probability of generating a negated literal is 50%. Overall, we generate 2000 ABoxes (each contains 2 – 23 assertions) and 2000 updates (each contains 1 – 11 assertions).[1]

For each ABox $\mathcal{A}$ and each update $\mathcal{U}$, we compute the logical update and the projective update of $\mathcal{A}$ with $\mathcal{U}$. The comparison of the implementations for logical updates and projective updates is depicted in the left graph of Figure 12 in which the sizes of updated ABoxes (vertical axis) are plotted against the sizes of the inputs (horizontal axis).[2] It is clear that the sizes of logical updates increase much faster than the ones of projective updates. With nesting existential and value restrictions in the inputs, the computation of logical updates easily used up memory.

An algorithm for computing logical updates based on the one in Section 3.1 is implemented in ECLiPSe-Prolog [ECL09] by Drescher with optimizations discussed

[1] All generated inputs in this chapter are in KRSS format [PSS93].

[2] Except the graphs of Figure 15, the numbers in vertical axes in the graphs in this chapter are logarithmized ($\log_e n$) in order to achieve clearer view of experimental results.

in [DLB+09a] to achieve a more compact representation of logical updates. Drescher provided the experimental results in this chapter that are related to this algorithm. Most of the optimization techniques in the implementation are syntactical such as simplifying the construction of $C^{\mathcal{D}_\mathcal{U}}(a)$ based on the UNA and identifying independent assertions, while a few of them involve DL reasoning. We do not discuss the technical details of those optimizations but rather use examples to illustrate their intuition.

Consider the concept assertion $A(a)$ and $\mathcal{D}_\mathcal{U} = \{\neg A(b)\}$. The construction defined in Section 3.1 produces $A^{\mathcal{D}_\mathcal{U}}(a) = (A \sqcup \{b\})(a)$. Based on the UNA, we can simplify it and obtain an equivalent assertion $A(a)$.

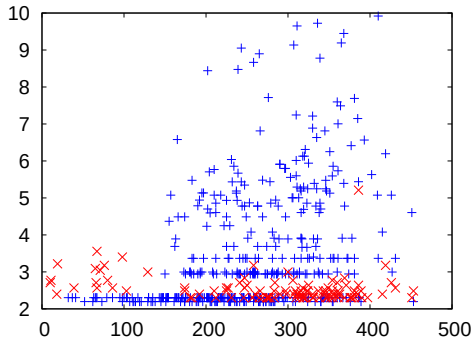
For an ABox $\mathcal{A}$ and an update $\mathcal{U}$, $\alpha \in \mathcal{A}$ is an *independent assertion* iff $\mathcal{A} * \mathcal{U} \equiv ((\mathcal{A} \setminus \{\alpha\}) * \mathcal{U}) \cup \{\alpha\}$. Intuitively, independent assertions in $\mathcal{A}$ are the assertions which may be intact when one updates $\mathcal{A}$ with $\mathcal{U}$. An easy way of syntactically detecting *some* independent assertions is to identify assertions which do not contain any concept or role names appearing in the update.

One of the optimization techniques which involve DL reasoning is to identify all diagrams $\mathcal{D}$ such that $\mathcal{D} \cup \mathcal{A}$ is inconsistent. The disjuncts for such diagrams can be dropped when the logical update is constructed [LLMW06c]. Of course, the consistency checking of $\mathcal{D} \cup \mathcal{A}$ needs the support of DL reasoning. In the implementation, we did not use the optimization techniques which require reasoning because the time spending on them was not paid off.

The comparison of the sizes of optimized logical updates and projective updates is displayed in the right graph of Figure 12. On the one hand, the graph indicates that there are a few inputs for which the logical updates are space consuming even for the optimized construction. More specifically, there are 110 inputs of which the sizes of logical updates are bigger than the sizes of projective updates including 64 inputs for which the computation of logical updates used up memory. On the other hand, the sizes of most logical updates constructed by the optimized algorithm are smaller than the sizes of projective updates. This suggests that the optimizations effectively decreased the sizes of logical updates on most testing data. The computing time of the updated ABoxes is displayed in Figure 13.[3] Building logical updates took more time than computing projective updates because of optimizations for the former. Considering the decreased size, it is clear that those optimizations are valuable.

We also plot in the right graph of Figure 12 the sizes of logical updates in $\mathcal{ALCO}^+$, which is the DL $\mathcal{ALCO}$ extended with the role constructors introduced in Section 3.3. We can see that the sizes of logical updates in $\mathcal{ALCO}^+$ are the smallest. In fact, they can be computed in a short time (within 20ms for most of them). We did not plot the time of those logical updates in Figure 13 because the points would be very close to the horizontal axis and lap over some other points. However, the main obstacle which keeps us from using those updates in practice is to find a reasoner for this DL. Likewise, reasoning on Boolean ABoxes is not directly supported by the state-of-the-art DL reasoners. In the next section, we will illustrate how to solve this problem.

[3]The function for measuring runtime in ECLiPSe-Prolog returns 0 if the runtime is less than 10 milliseconds. We also did this approximation for the runtime obtained from Java. Those points with vertical coordinates $\log_e 0$ are missing in Figure 13.



Logical updates: +, projective updates: ×
The numbers in vertical axis are logarithmized.

Figure 13: The computing time of logical and projective updates.

6.2 Reasoning with Logical Updates

Let $\mathcal{L}$ be a DL between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$. As we have seen in Chapter 3, the construction of logical updates of $\mathcal{L}$-ABoxes assembles Boolean $\mathcal{LO}^{@}$-ABoxes. Reasoning with Boolean ABoxes is not directly supported by standard DL reasoners.[4] Reasoning with $\mathcal{ALC}$-LTL formulas [BGL08] requires Boolean ABox reasoning, too. In order to do reasoning with those logical updates, we need to handle the @ constructor and Boolean assertions. In this section, we concentrate on three approaches to solving the consistency problem of such Boolean ABoxes: The first is based on a polynomial reduction which is consistency preserving, the second is based on DPLL(T), and the third resorts to the reasoner Spartacus for hybrid logic.

The reduction approach

Given a Boolean $\mathcal{LO}^{@}$-ABox $\mathcal{A}$, we construct a non-Boolean $\mathcal{LO}$-ABox $\mathcal{A}''$ which preserves consistency of $\mathcal{A}$: We first transform $\mathcal{A}$ to an equivalent non-Boolean $\mathcal{LO}^{@}$-ABox $\mathcal{A}'$ as we show in the proof of Theorem 49. Then, $\mathcal{A}'$ can be transformed to a consistency preserving non-Boolean $\mathcal{LO}$-ABox $\mathcal{A}''$ by the polynomial reduction in the proof of Lemma 50. Overall, the size of $\mathcal{A}''$ is polynomial in the size of $\mathcal{A}$. Since $\mathcal{A}''$ is a non-Boolean $\mathcal{LO}$-ABox, we can resort to DL reasoners to decide its consistency.

The DPLL(T) approach

Given a Boolean $\mathcal{LO}^{@}$-ABox $\mathcal{A}$, we first convert $\mathcal{A}$ into a consistency preserving Boolean $\mathcal{LO}$-ABox $\mathcal{A}'$ by Lemma 51 and then decide consistency of $\mathcal{A}'$ by the DPLL(T) approach.

Most modern SAT-solvers [ES03, dMB08] are variants of the DPLL procedure introduced by Davis, Putnam, Logemann, and Loveland in [DP60, DLL62]. We first briefly recall the DPLL transition rules in [NORCR07] and then illustrate how to

[4] A list of DL reasoners is available at `http://www.cs.man.ac.uk/~sattler/reasoners.html`.

employ their extension, the DPLL(T) approach, to solve the consistency problem of Boolean ABoxes.

The DPLL system can effectively decide satisfiability of a propositional formula in conjunctive normal form (CNF), which is the normal form for most modern SAT-solvers. Before we introduce the definition of the DPLL system, we first give some notions which will be used later on.

A *literal* is p or $\neg p$ for a propositional letter p. A finite disjunction of literals is called a *clause*. A *propositional formula in CNF* is a finite conjunction of clauses. A *partial model* M is a finite sequence $l_1, \ldots, l_n$ of literals (possibly labeled with d) such that there is not a literal l with both $l, \neg l$ appearing in M, where if $l = \neg p$ for some propositional letter p, then $\neg l = p$. The labeled literals in M are called *decision literals*. The *decision level of* M, denoted by $\mathsf{dl}(M)$, is the number of decision literals in M. For every literal l in M, the *decision level of* l *(*$\neg l$*, respectively)*, denoted by $\mathsf{dl}(l)$ ($\mathsf{dl}(\neg l)$, respectively)), is the number of decision literals in M from the first element of M to l.

For a partial model M and a literal l, l is *defined in* M if $l \in M$ (denoted also by $M \models l$), or $\neg l \in M$ (denoted also by $M \models \neg l$). Otherwise, l is *undefined in* M. For a partial model M and a clause C with $C = l_1 \vee \cdots \vee l_n$, $M \models \neg C$ iff $M \models \neg l_i$ for all i with $1 \leq i \leq n$. In this case, we call C a *conflict clause under* M.

Definition 60 (The DPLL System in [NORCR07])**.** The *DPLL system* consists of the transition rules in Figure 14. The data structure on which the DPLL system works is of the form $M \parallel F$, where M is a partial model and F is a propositional formula in CNF. By each application of a rule, $M \parallel F$ is transformed to $M' \parallel F'$ (denoted with $M \parallel F \Longrightarrow M' \parallel F'$). △

Given a propositional formula F (in CNF), we can run exhaustively the DPLL transition rules in Figure 14 with the input $\emptyset \parallel F$. Different strategies for the application of the rules result in different DPLL algorithms. We adopt the strategy of MiniSat [ES03] to implement the algorithm of our SAT-solver, which is the same as Algorithm 1 without Line 5, 12, and 21.

The partial model M is initialized in Line 1. The unit propagate rule is exhaustively applied in Line 4–6. The function $\mathsf{containsConflictClause}(F, M)$ in Line 7 returns true iff there exists a conflict clause C in F under the current partial model M. If no conflict clauses have been found, then depending on whether M is complete (Line 8), either the algorithm terminates with **true** (Line 9) or the decide rule is applied (Line 11). If some conflict clause is detected, then depending on the decision level of M (Line 15), either the algorithm returns **false** (Line 16) or the backjump rule is applied (Line 18–19) according to the conflict clause. A so-called *backjump clause* C' generated by $\mathsf{backjump(C)}$ is added into F in Line 20. This process will be repeated (Line 3–23) until the algorithm returns **true** or **false**.

One of the strengths of the DPLL system is to use Backjump instead of Backtrack so that the searching space can be efficiently pruned [NORCR07]. In order to apply the backjump rule, we need to construct a backjump clause $C' \vee l'$.

UnitPropagate
$M \parallel F \wedge (C \vee l) \Longrightarrow M, l \parallel F \wedge (C \vee l)$
if $M \models \neg C$ and l is undefined in M.

Decide
$M \parallel F \Longrightarrow M, l^{\mathsf{d}} \parallel F$
if l or $\neg l$ occurs in a clause of F, and l is undefined in M.

Fail
$M \parallel F \wedge C \Longrightarrow \mathsf{fail}$
if $M \models \neg C$ and $\mathsf{dl}(M) = 0$.

Backjump
$M, l^{\mathsf{d}}, N \parallel F \wedge C \Longrightarrow M, l' \parallel F \wedge C$
if $M, l^{\mathsf{d}}, N \models \neg C$, and there is some clause $C' \vee l'$ such that
$F \wedge C \models C' \vee l'$, $M \models \neg C'$, l' is undefined in M, and l' or $\neg l'$ occurs in F.

Learn
$M \parallel F \Longrightarrow M \parallel F \wedge C$
if all propositional letters of C occur in F and $F \models C$.

Figure 14: The DPLL transition rules.

There are basically two approaches to constructing a backjump clause: the implication graph based approach [ZMMM01] and the resolution based approach [NORCR07]. In a typical DPLL implementation, the clause that records the reason of the existence of each non-decision literal l in M is memorized [NORCR07]. There are two possibilities that a non-decision literal l can be added into M. The first one is the result of applying the unit propagate rule, in which case we associate l with $C \vee l$. The other one is the result of applying the backjump rule and thus we associate l with the constructed backjump clause $C' \vee l'$. For a non-decision literal l, we use the function $\mathsf{cl}(l)$ to denote the clause associated to l.

Algorithm 2 shows how to implement the backjump rule, where a backjump clause is built based on the resolution approach. The set L collects (Line 1) all literals in the conflict clause C that are in the decision level of M. The most recently defined literal l in C according to the current partial model is obtained by the function getLastDefinedElement(C) (Line 3). Then, resolution is done (Line 4) with the clause C and the clause associated to l. The set L is updated (Line 5) afterwards. Repeatedly resolving the most recently defined literal l in C until there is only one literal in the decision level of M leads to a backjump clause [NORCR07]. In Line 7, we get the value of k which is the second biggest decision level of the literals in the backjump clause. The literals in M whose decision level are bigger than k are removed from M (Line 8). The constructed backjump clause $C' \vee l'$ consists of the literals in C and the only literal in L is assigned to l' (Line 9). The partial model is updated in Line 10 to decision level k. The backjump clause is associated (Line 11) to the new non-decision literal l' in M. The backjump clause C is returned and will be added into F by the

Algorithm 1 A DPLL(T) Algorithm

Procedure isConsistent(F)
Input: a propositional formula F in CNF
Output: **true**/**false**

```
1:  M := ∅
2:  loop
3:    loop
4:      UnitPropagate()
5:      callDLReasoner()
6:    end loop
7:    if not containsConflictClause(F, M) then
8:      if all literals in F is defined in M then
9:        return true
10:     else
11:       Decide()
12:       callDLReasoner()
13:     end if
14:   else
15:     if dl(M) = 0 then
16:       return false
17:     else
18:       C := getAConflictClause()
19:       C' := Backjump(C)
20:       Learn(C')
21:       callDLReasoner()
22:     end if
23:   end if
24: end loop
```

learn rule.

The DPLL(T) approach combines a DPLL procedure with a theory solver that can handle conjunctions of literals in the theory to solve the satisfiability problem modulo theories (SMT) [NORCR07]. The consistency problem of Boolean ABoxes can be viewed as an instance of SMT where ABox assertions are the theory atoms and a DL reasoner serves as the theory solver.

Let us consider the above DPLL algorithm extended with Line 5, 12, and 21. Suppose that the input Boolean ABox $\mathcal{A}$ of Algorithm 1 is in CNF, i.e., substituting every ABox assertion in $\mathcal{A}$ with a propositional letter results in a propositional formula $F_{\mathcal{A}}$ in CNF. The idea is to run the DPLL algorithm on $F_{\mathcal{A}}$ and to call the DL resoner whenever a new literal is added into the current partial model M (by every application of UnitPropagate, Decide, or Backjump). The DL reasoner will determine consistency of the ABox containing the assertions corresponding to M. Thus, the DPLL(T) algorithm can recognize the impossibility of extending the current partial model and apply the backjump rule as soon as possible. One issue is that in general DL reasoners

Algorithm 2 A backjump algorithm

Procedure backjump(C)
Input: a conflict clause C
Output: a backjump clause

1: $L := \{l \mid l$ is a disjunct in $C \wedge \mathsf{dl}(l) = \mathsf{dl}(M)\}$
2: **while** $|L| > 1$ **do**
3: $\quad l :=$ getLastDefinedElement(C)
4: $\quad C :=$ doResolution($C, \mathsf{cl}(l)$)
5: $\quad L := \{l \mid l$ is a disjunct in $C \wedge \mathsf{dl}(l) = \mathsf{dl}(M)\}$
6: **end while**
7: $k :=$ 2ndMaxElement($\{\mathsf{dl}(l) \mid l$ occurs in $C\}$)
8: $M :=$ removeElementsBiggerThanLevel(M, k)
9: $l' :=$ getAElement(L)
10: $M := M, l'$
11: $\mathsf{cl}(l') := C$
12: **return** C

cannot directly deal with negated assertions such as $\neg(C(a))$ and $\neg(r(a,b))$. Fortunately, this issue can be easily rounded if the DL reasoners can handle nominals since $\neg(C(a)) \equiv (\neg C)(a)$ and $\neg(r(a,b)) \equiv (\forall r.\neg\{b\})(a)$. We have a preprocessing step for this before invoking the DL reasoner.

One of the challenges for improving the performance of the algorithm based on the DPLL(T) approach is to ensure that the theory solver efficiently finds a minimal explanation of an inconsistent theory. It is widely understood that small explanations tend to behave better in practice [NORCR07]. Explaining why an ABox is inconsistent is an instance of the pinpointing problem [Sch03, BP08]. An explanation is a minimal subset of the input ABox, containing only those assertions that are responsible for the inconsistency. Formally, it is defined as follows: Given an inconsistent ABox $\mathcal{A}$, an *explanation* of $\mathcal{A}$ is an inconsistent subset $\mathcal{A}'$ of $\mathcal{A}$. An explanation $\mathcal{A}'$ of $\mathcal{A}$ is *minimal* iff there is no explanation $\mathcal{B}'$ of $\mathcal{A}$ such that $\mathcal{B}'$ is a strict subset of $\mathcal{A}'$. Based on smaller explanations, one can usually build smaller backjump clauses in the DPLL(T) approach. The smaller backjump clauses are, the more search space is pruned.

As shown in Algorithm 3, if the DL reasoner reports inconsistency (Line 1), then according to the decision level of M, either the backjump rule will be applied (Line 3–7), followed by an application of the learn rule, or the algorithm isConsistent(F) terminates and reports inconsistency of the input (Line 9). In the former case, an explanation C of inconsistency will be given by the DL reasoner, which means that one of the literals in the set of literals corresponding to the assertions in the explanation has to be falsified (Line 4) since the set of those assertions in the explanation is inconsistent. In this sense, C is a conflict clause under the current partial model and hence we can do Backjump with C (Line 5). The generated backjump clause is added into F in Line 6. Since there is a new literal l' in M after the application of the backjump rule, we will recursively call the DL reasoner (Line 7).

Algorithm 3 A DL-reasoner calling Algorithm

Procedure callDLReasoner()

```
1: if not isConsistentByDLReasoner(M) then
2:   if dl(M) > 0 then
3:     C :=getAnExplanantionByDLReasoner(M)
4:     C := {¬l | l is a disjunct in C}
5:     C' :=Backjump(C)
6:     Learn(C')
7:     callDLReasoner()
8:   else
9:     reportInconsistency()
10:  end if
11: end if
```

The Spartacus approach

The reasoner Spartacus can be used for deciding the satisfiability problem of formulas in hybrid logic [Göt09]. For a Boolean $\mathcal{ALCO}^{@}$-ABox $\mathcal{A}$, we can construct, with the help of the @ constructor, a formula $\varphi_{\mathcal{A}}$ in hybrid logic such that $\mathcal{A}$ is consistent iff $\varphi_{\mathcal{A}}$ is satisfiable. Basically this construction is just to present $\mathcal{A}$ in the syntax of hybrid logic. First replace all role assertions in $\mathcal{A}$ by the following rules:

$$r(a,b) \rightsquigarrow \exists r.\{b\}(a); \quad \neg r(a,b) \rightsquigarrow \forall r.\neg\{b\}(a).$$

Then, all concept assertions $C(a)$ in $\mathcal{A}$ is replaced by $@_a C'$ where C' is obtained by applying the following rules to C:

$$\begin{array}{rclrclrcl} A & \rightsquigarrow & A; & C \sqcap D & \rightsquigarrow & C \wedge D; & C \sqcup D & \rightsquigarrow & C \vee D; \\ \{a\} & \rightsquigarrow & = a; & \exists r.C & \rightsquigarrow & \langle r\rangle C; & \forall r.C & \rightsquigarrow & [r]C. \end{array}$$

Note that there are no rules for negation and the @ constructor since they are presented in the same way in both Description Logic and hybrid logic.

Experimental results

Two algorithms which solve the consistency problem of Boolean ABoxes are implemented: one is based on the reduction approach and the other is based on the DPLL(T) approach. Pellet was chosen as the DL reasoner in the implementation because it supports nominals and pinpointing [SPG+07]. Henceforth, we call the former implementation Pellet-UR and the latter one Pellet-DPLL. When we do reasoning with Spartacus, we need to transform the testing data into the format which Spartacus accepts. The time of this transformation is not included in the runtime. The experiments were carried out on two sets of testing data: one of them is obtained from a random Boolean ABox generator and the other is from logical updates in Section 6.1.

We implemented a random generator of Boolean $\mathcal{ALC}$-ABoxes, which randomly generates a propositional formula in CNF and then assigns a randomly generated

n_1	n_2	n_3	d	ncs	nc	nr	ni	pr
3 – 53	6 – 36	8 – 83	2 – 23	6 – 106	2 – 12	1 – 12	1 – 12	20

Table 4: Parameters of the random Boolean ABox generator.

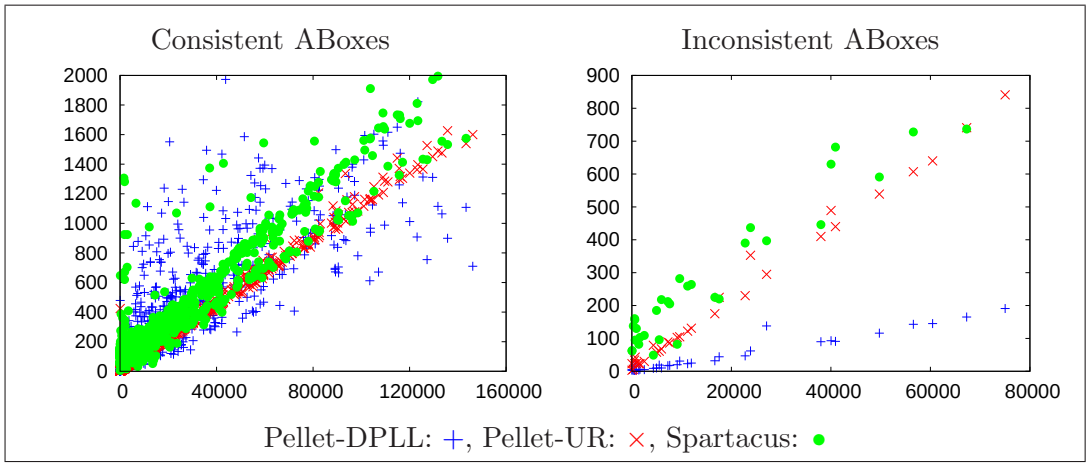


Figure 15: Reasoning with generated Boolean ABoxes.

assertion to each propositional letter. Several parameters are used to control the shape of the generated Boolean ABoxes: the number n_1 of literals in a clause, the number n_2 of propositional letters, the number n_3 of clauses, the number d of nesting roles in a concept assertion, the number ncs of the constructors in a concept assertion, the numbers nc, nr, and ni of concept names, role names, and individual names in a concept assertion, and the probability $pr\%$ of generating a role assertion. The parameters used for generating 1000 testing data are displayed in Table 4.

In Figure 15, we plot the runtime of Pellet-DPLL, Pellet-UR, and Spartacus on these testing data against the sizes of the Boolean ABoxes. We depict the performance on consistent and inconsistent Boolean ABoxes separately. For Pellet-UR, the runtime linearly increases with the size of the input. Spartacus performed similarly to Pellet-UR. On 33 inconsistent ABoxes Pellet-DPLL also exhibits a linear increase in runtime, while on 967 consistent ABoxes the runtime is less predictable. Pellet-DPLL performs better on all of the inconsistent Boolean ABoxes. On about 60% of the consistent ABoxes, the Pellet-UR approach did best. In Pellet-DPLL the frequent invocations of Pellet are more likely to pay off if inconsistency of the current partial model can be detected often: We then can build a backjump clause that helps to prune the search space. The runtime of Pellet-UR is about the same on both consistent and inconsistent input data if they have the similar size. This holds for Spartacus as well.

In Section 6.1, we have constructed 1936 Boolean ABoxes which are logical updates of the randomly generated ABoxes.[5] Those logical updates provide us another set of

[5]Notice that the logical updates computed by the optimized algorithm are Boolean ABoxes in

Time	Pellet-DPLL	Pellet-UR	Spartacus
0 – 1000	1807	1861	1859
1000 – 2000	24	30	10
2000 – 3000	17	11	8
3000 – 10000	55	18	2
10000 – 30000	16	4	9
30000 – 300000	12	12	3
300000 – 450000	3	0	0
*	2	0	0

Table 5: Reasoning with logical updates.

testing data. Since every updated ABox is consistent, we check its logical consequences instead of its consistency. For every Boolean ABox $\mathcal{A}$, we randomly generate 10 assertions, each of which contains only the names occurring in $\mathcal{A}$. The probability of generating a role assertion is 30%. Every generated concept assertion constains 30 concept constructors. Let $\mathcal{A}$ be a Boolean ABox and φ an assertion. Then φ is a logical consequence of $\mathcal{A}$ iff $\mathcal{A} \cup \{\neg\varphi\}$ is inconsistent. Although negation is not explicitly available in Boolean ABoxes, the negation sign in front of φ can be absorbed into the assertion φ as we did in the DPLL(T) approach.

We plot the testing results in the left graph of Figure 16 in which the numbers on the horizontal axis stand for the size of the ABox and the update, and the numbers on the vertical axis are the average time for reasoning against 10 assertions plus the time of computing the corresponding update. We can see that Pellet-UR worked fastest for small inputs and Spartacus did fastest for big inputs. The statistics of the runtime for those three approaches is given in Table 5 in which the numbers in the first column means the range of time, $*$ means running out of memory, and other numbers means for how many inputs the corresponding approach returned answers in the given time range.

We also compare the performance of reasoning with logical updates and projective updates. We did the testing with the latter starting from the small inputs proceeding to the big ones. On the first 1052 inputs, we got answers on 1047 inputs. However, after that using up memory happened very often (approximately once for every three inputs). We plot the testing results according to the first 1047 inputs for which reasoning with both logical and projective updates returns an answer. As shown in the right graph of Figure 16, reasoning with logical updates (Pellet-UR) is faster than reasoning with projective updates on most inputs.

For both logical and projective updates, the time for computing updates is relatively small compared to the reasoning time, and the latter is dominated by the size of the computed updated ABox. Hence, optimizations aiming to streamline updates

CNF that are usually more compact than the ones in DNF constructed according to the algorithm in Section 3.1. Cf. [DLB+09a] for a more detailed discussion of the representation of logical updates in CNF and DNF.

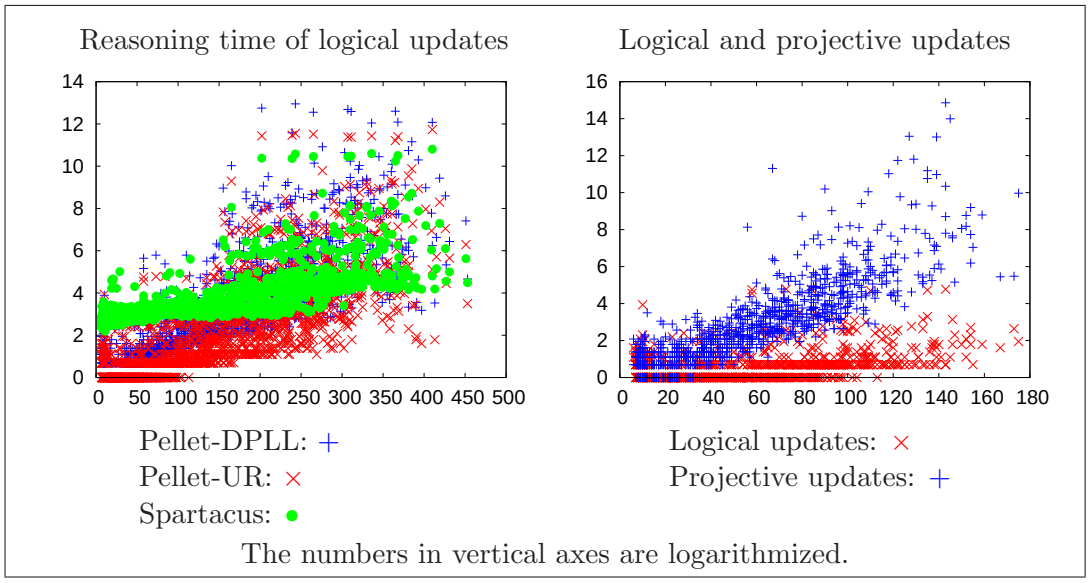


Figure 16: Reasoning with updates.

are valuable. Although projective updates exhibit better computational behavior in theory, they did not perform better than the optimized construction of logical updates. For this reason, optimizations to the former are strongly required. Note that most optimization techniques in [DLB+09a] for logical updates do not help to decrease the size of projective updates. For example, the one based on the UNA and the one detecting diagrams $\mathcal{D}$ such that $\mathcal{D} \cup \mathcal{A}$ is inconsistent are not applicable to the construction of projective updates since they aim at the specific construction of logical updates. Identifying independent assertion can be applied, but it does not impact much on the size of the constructed projective update.

Actually, the concept C_{bi} which is a conjunction of bi-implications is responsible for the size of the constructed projective update. The construction of bi-implications in Figure 9 can be simplified to $\mathsf{A}^{(0)}_{\exists r.C} \leftrightarrow \exists r.\mathsf{A}^{(0)}_{C}$ and $\mathsf{A}^{(0)}_{\forall r.C} \leftrightarrow \forall r.\mathsf{A}^{(0)}_{C}$ if r is not in $\mathsf{R}_{\mathsf{fle}}$, i.e., the update does not change the interpretation of r. This can be applied to the bi-implications for qualified number restrictions, too. We have included this optimization in our implementation. In order to improve the performance of projective updates, we need to explore other optimizations. This is left as future work.

6.3 An Implementation of the Projection Algorithm

As described in Chapter 2, the projection problem can be solved based on either progression or regression. In [BLM+05], the projection problem in DLs has also been solved by a reduction to the logical consequence problem of ABoxes w.r.t. acyclic TBoxes, which is similar to regression. In this section, we present experimental results of the implementation of this reduction.

Our testing data are the ABoxes $\mathcal{A}$ and the updates $\mathcal{U}$ generated in Section 6.1

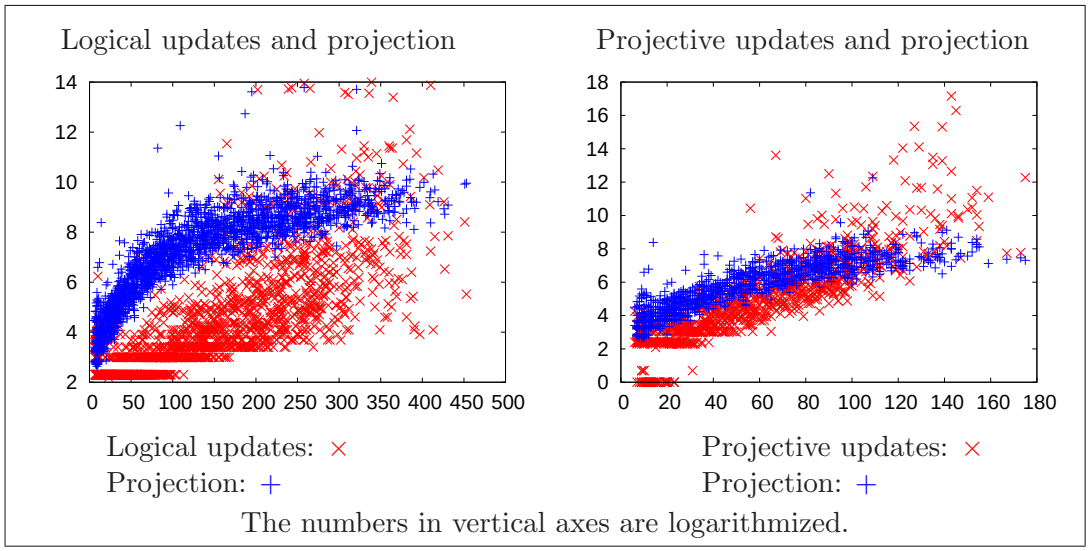


Figure 17: Experimental results on the projection problem.

and the assertions φ generated in Section 6.2. We check whether φ is a consequence of applying $\mathcal{U}$ to $\mathcal{A}$. We compare the runtime of the implementation based on the reduction and the ones based on updating ABoxes. Since we need to construct a new ABox and a new TBox for each assertion φ even without changing the original ABox $\mathcal{A}$ and the update $\mathcal{U}$, we include all the construction time in the runtime. For reasoning with updates, we count the computation time of updates only once in the runtime if $\mathcal{A}$ and $\mathcal{U}$ do not change. Notice that we include all reasoning time for 10 assertions instead of average reasoning time. The aim of the testing we did in this section is to compare the approaches to reasoning about action but not the efficiency of reasoners. For this reason, we chose Pellet to decide consistency. In principle, we can also use FaCT++ [TH06] and Spartacus.

The implementation based on the reduction can solve projection on 1991 out of 2000 inputs, which means that it returns answers for all 10 assertions on the same ABox and update. The testing finished in 10 minutes on all those inputs and in 2 minutes on most of them. The left graph of Figure 17 shows its performance compared to reasoning with logical updates (Pellet-UR). There are 101 out of 1927 inputs on which Pellet-UR spent more time. In the right graph of Figure 17, we plot the testing results for the inputs for which both reasoning with projective updates and projection return answers. Although the former run faster than the latter on 856 out of 1044 inputs, the time used by the former increases more rapidly for the relatively big inputs.

We summarize the experimental results obtained so far about projection in Table 6. The upper part of the table lists the number of experiments performed for the corresponding methods, while the lower part presents pairwise comparison among them and the numbers in parentheses say for how many inputs both the two corresponding methods did not run of the memory. It is worth observing that there are no inputs for which two of those three reasoning methods did not return answers. They

	Logical update (a)	Proj. update (b)	Projection (c)
No. of experiments	2000	1052	2000
Achieving answers	1934	1047	1991
	$<$	$>$	$=$
(a) vs. (b) (1047)	907	57	83
(a) vs. (c) (1927)	1824	101	2
(b) vs. (c) (1044)	856	188	0

Table 6: A summary of experimental results.

met the worst cases in different inputs although in principle reasoning with the inputs containing nesting existential and value restrictions is supposed to be expensive for all of them.

We also performed experiments which do reasoning with a sequence of updates. Testing data are the same as before. We check whether one assertion φ is a consequence of applying $\mathcal{U}_1, \ldots, \mathcal{U}_n$ to $\mathcal{A}$ for all $n \leq 2000$. The sequence of updates is merged into one update as discussed in Section 3.4. We did not update the ABoxes iteratively because it would blow up quickly even for one of the smallest inputs for both projective updates ($n = 15$) and projection ($n = 39$). For logical updates, whether merging or not did not have much impact on the performance of reasoning. We run the experiments on 10 ABoxes according to their sizes. The results are listed in Table 7 in which the numbers in the first column are the indices of files storing the input ABoxes and other numbers stand for the length (n) of the sequence of updates the corresponding experiment was carried out to in 30 minutes. If the length reaches 2000 in 30 minutes, then the number in parentheses indicates the time of reasoning. The order of the indices in general reflects the order of sizes of the ABoxes. As we can see from the table, reasoning with projective updates ran fastest on the small inputs. However, once the size of ABox grows to a certain degree, the reasoning with the projective updates becomes very expensive. From this point of view, logical updates are not as sensitive as projective updates to the size of the original ABox. Although projection did not perform best on the small inputs, it was the most efficient method on the large inputs.

The testing results obtained in this section are at odds with the empirical conclusion that progression outperforms regression when the history of updates is long. The reason might be that the implementations are not optimal, or that reasoning about action in DL domains essentially behaves like this.

ABox index	Pellet-UR	Proj. update	Projection
37	2000 (526429)	2000 (306661)	2000 (731408)
406	2000 (582790)	2000 (326655)	2000 (849365)
668	71	1568	693
713	168	80	2000 (834778)
1009	86	13	181
1124	78	6	2000 (903835)
1213	41	7	639
1268	68	5	668
1331	39	11	238
1712	19	1	181

Table 7: Reasoning with sequences of updates.

Chapter 7

Verification of DL-LTL Formulas

In this chapter, we verify DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates. In Section 7.1, we define the satisfiability problem and the validity problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates. We show that the latter problem can be polynomially reduced to the complement of the former problem. We investigate the satisfiability problem in DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ for two cases: unconditional updates and conditional updates. It turns out that in the former case the satisfiability problem has the same complexity as the (non-)projection problem. This is illustrated in Section 7.2. In the latter case, we present in Section 7.3 an algorithm to solve the satisfiability problem by extending the decision procedure of satisfiability of propositional LTL formulas [WVS83, VW86] with the help of the techniques to decide the projection problem [BLM$^+$05]. This algorithm even works for the more general problem that allows for sequences of updates accepted by a Büchi automaton instead of a fixed sequence of updates. Suppose that Σ is a set of updates. In the more general problem, we are offered a way to describe any infinite sequence of updates in an ω-regular language over Σ instead of only Büchi sequences of updates since the class of languages accepted by Büchi automata coincides with the class of ω-regular languages [BK08]. The lower bounds are left open in the case that conditional updates, or Büchi automata are allowed in the satisfiability problem.

7.1 The Inference Problems

In this section we formally introduce the inference problems relevant to verifying DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates.

Let Σ be a finite set of updates. An *infinite sequence* over Σ is a function $w : \mathbb{N} \mapsto \Sigma$, where $\mathbb{N}$ is the set of natural numbers. Intuitively, $w(n)$ is the n-th update in the sequence. The following definition tells us the form of infinite sequences of updates we are interested in:

Definition 61 (Büchi Sequence of Updates)**.** Let Σ be a finite set of updates. A *Büchi sequence of updates* is an infinite sequence w over Σ such that there are m, n $(m < n)$ satisfying the following property:

$$\forall i > 0 : w(m + i) = w(m + k),$$

where k is congruent to i modulo n. The *size of* w, denoted by $|w|$, is $\sum_{i=0}^{m+n} |w(i)|$. △

Intuitively, an infinite sequence w over Σ is a Büchi sequence if it runs into a cycle from $w(m+n)$ back to $w(m+1)$. Hence, we also denote a Büchi sequence of updates by $\alpha_1 \cdots \alpha_m(\beta_1 \cdots \beta_n)^\omega$. Such a sequence of updates is called a *Büchi sequence of updates* since for every Büchi automaton $\mathcal{B}$, if the language accepted by $\mathcal{B}$ is not empty, then $\mathcal{B}$ accepts a word with this form [TB73].

We follow the notion of $\mathcal{ALC}$-LTL formulas from [BGL08], where temporal operators are allowed only in front of ABox assertions, and generalize this notion to any DL. Such a formula is called a *DL-LTL formula*. Equivalently, a DL-LTL formula can be obtained by replacing propositional letters in a propositional LTL formula with an ABox assertion.

Definition 62 (DL-LTL formula)**.** The set of *DL-LTL formulas* can be defined by induction:

- if φ is an ABox assertion, then φ is a DL-LTL formula;
- if ϕ, ψ are DL-LTL formulas, then so are $\phi \wedge \psi$, $\phi \vee \psi$, $\neg\phi$, $\mathsf{X}\phi$, and $\phi \,\mathsf{U}\, \psi$.

The *size of a DL-LTL formula* φ, denoted by $|\varphi|$, is the number of LTL operators in φ plus $|\psi|$ for every occurrence of every assertion ψ in φ. △

As usual, we use true as an abbreviation for $A(a) \vee \neg A(a)$ with a concept name A and an individual name a, $\Diamond\phi$ as abbreviation for $\mathsf{true}\,\mathsf{U}\,\phi$, and $\Box\phi$ as an abbreviation for $\neg\Diamond\neg\phi$.

A next formula $\mathsf{X}\phi$ can be read as ϕ holds in the next time point. An until formula $\phi \,\mathsf{U}\, \psi$ can be read as ϕ will always hold until ψ holds.

Definition 63 (DL-LTL Structure)**.** A *DL-LTL structure* is an infinite sequence $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ of interpretations $\mathcal{I}_i = (\Delta, \cdot^{\mathcal{I}_i})$. Given a DL-LTL formula ϕ, a DL-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$, and a time point $i \in \{0, 1, 2, \ldots\}$, *validity of* ϕ *in* $\mathfrak{I}$ *at time* i (written $\mathfrak{I}, i \models \phi$) is defined inductively:

$$
\begin{array}{lll}
\mathfrak{I}, i \models \phi & \text{iff} & \mathcal{I}_i \models \phi \text{ where } \phi \text{ is an ABox assertion} \\
\mathfrak{I}, i \models \phi \wedge \psi & \text{iff} & \mathfrak{I}, i \models \phi \text{ and } \mathfrak{I}, i \models \psi \\
\mathfrak{I}, i \models \phi \vee \psi & \text{iff} & \mathfrak{I}, i \models \phi \text{ or } \mathfrak{I}, i \models \psi \\
\mathfrak{I}, i \models \neg\phi & \text{iff} & \text{not } \mathfrak{I}, i \models \phi \\
\mathfrak{I}, i \models \mathsf{X}\phi & \text{iff} & \mathfrak{I}, i+1 \models \phi \\
\mathfrak{I}, i \models \phi \,\mathsf{U}\, \psi & \text{iff} & \text{there is } k \geq i \text{ such that } \mathfrak{I}, k \models \psi \text{ and} \\
 & & \mathfrak{I}, j \models \phi \text{ for all } j, i \leq j < k
\end{array}
$$

△

In general, interpretations of concept and role names in a DL-LTL structure can be completely independent of each other, or some concept names or role names are interpreted in the same way. In the latter case, they are called *rigid names* in [BGL08].

We do not use rigid names here because the changes on the interpretations of names are restricted by the semantics of update. In DLs, an interpretation is a complete description of the world and thus a DL-LTL structure can be viewed as a sequence of consecutive snapshots of the world. In order to use a DL-LTL structure to represent the world descriptions in which changes are caused by applying updates, we introduce the following definition:

Definition 64 (DL-LTL Structure w.r.t. w)**.** Let w be a Büchi sequence of updates. A *DL-LTL structure* $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ *w.r.t.* w is a DL-LTL structure such that $\mathcal{I}_i \Longrightarrow_{w(i)} \mathcal{I}_{i+1}$ for all $i \geq 0$. △

In contrast to [BGL08], we have the restriction that we disallow general concept inclusions (GCIs) and even concept definitions to appear in the DL-LTL formula φ because we want to avoid the semantics problem. For example, if $\varphi = \Box(C \sqsubseteq D)$, then each interpretation in a DL-LTL structure satisfying φ needs to satisfy $C \sqsubseteq D$. As a result, this indirectly introduces GCIs as domain constraints into DL action formalisms. However, there does not yet exist a satisfactory semantics of such action formalisms even in the DL $\mathcal{ALC}$ [BLM$^+$05]. For concept definitions, we could allow the formulas of the form $\Box(A \equiv C)$ as a conjunction to occur at the top-level of φ. This is equivalent to including such concept definitions as domain constraints. This extension would not change the complexity results in this chapter [BLLuM05]. In order to keep the presentation uniform in this thesis, we do not consider acyclic TBoxes. If concept definitions could occur in an arbitrary form, e.g.,

$$(A \equiv C) \wedge \mathsf{X}\neg(A \equiv C),$$

then the belief revision problem is more suitable in this setting than the update problem as discussed in Section 1.2.

In the next definition, we introduce the inference problems relevant to verifying DL-LTL formulas considered in this chapter:

Definition 65 (Satisfiability and Validity)**.** Let $\mathcal{A}$ be an ABox, w a Büchi sequence of updates and φ a DL-LTL formula. Then,

- φ is *satisfiable w.r.t.* $\mathcal{A}$ *and* w iff there is a DL-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ w.r.t. w such that $\mathcal{I}_0 \models \mathcal{A}$ and $\mathfrak{I}, 0 \models \varphi$.
- φ is *valid w.r.t.* $\mathcal{A}$ *and* w iff for all DL-LTL structures $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ w.r.t. w, if $\mathcal{I}_0 \models \mathcal{A}$, then $\mathfrak{I}, 0 \models \varphi$.

△

Throughout this chapter, we abbreviate the satisfiability (validity, respectively) problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates as the satisfiability (validity, respectively) problem. We use a tuple $(\mathcal{A}, w, \varphi)$ to denote an input of the satisfiability or the validity problem, where $\mathcal{A}$, w, and φ are defined as in Definition 65. In the subsequent sections, we will show how to decide those two inference problems for DLs $\mathcal{L}$ between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$, i.e., all assertions appearing in $\mathcal{A}$, w, or φ are formulated in $\mathcal{L}$.

Note that for all DL-LTL formulas φ, all ABoxes $\mathcal{A}$, and all Büchi sequences w of updates, φ is valid w.r.t. $\mathcal{A}$ and w iff $\neg\varphi$ is unsatisfiable w.r.t. $\mathcal{A}$ and w. Thus, validity can be reduced to (un)satisfiability. From now on, we concentrate on the satisfiability problem.

7.2 Unconditional Updates

In this section, we solve the satisfiability problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates in DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ for the case that all the updates in the Büchi sequence w are unconditional updates.

Let $(\mathcal{A}, w, \varphi)$ with $w = \alpha_1 \dots \alpha_m(\beta_1 \dots \beta_n)^\omega$ be an input of the satisfiability problem. The following lemma tells us that, for all interpretations $\mathcal{I}$, if $\mathcal{I}'$ is the result of updating $\mathcal{I}$ with $\beta_1 \cdots \beta_n$, then $\mathcal{I}'$ is the result of updating $\mathcal{I}'$ with $\beta_1 \cdots \beta_n$, i.e., the sequence of interpretations runs into a circle after the application of the same sequence of updates twice.

Lemma 66. *Let $\mathcal{I}$ and $\mathcal{I}'$ be two interpretations and $\beta = \beta_1 \cdots \beta_n$ be a sequence of updates. If $\mathcal{I} \Longrightarrow_\beta \mathcal{I}'$, then $\mathcal{I}' \Longrightarrow_\beta \mathcal{I}'$.*

Proof. Suppose $\mathcal{I}' \Longrightarrow_\beta \mathcal{J}'$ for some interpretation $\mathcal{J}'$ (such a $\mathcal{J}'$ always exists, since for all updates $\mathcal{U}$, $\Longrightarrow_\mathcal{U}$ is a total function on all interpretations). Thus, it is enough to show that $\mathcal{I}' = \mathcal{J}'$. Since it follows from $\mathcal{I} \Longrightarrow_\beta \mathcal{I}'$ and $\mathcal{I}' \Longrightarrow_\beta \mathcal{J}'$ that $\Delta^\mathcal{I} = \Delta^{\mathcal{I}'} = \Delta^{\mathcal{J}'}$ and $a^\mathcal{I} = a^{\mathcal{I}'} = a^{\mathcal{J}'}$ for all $a \in \mathsf{N_I}$, it suffices to show that for all $A \in \mathsf{N_C}$ and all $r \in \mathsf{N_R}$, we have $A^{\mathcal{I}'} = A^{\mathcal{J}'}$ and $r^{\mathcal{I}'} = r^{\mathcal{J}'}$. Here we show only the former and the latter can be proved analogously.

"$\subseteq$": Assume that $A^{\mathcal{I}'} \not\subseteq A^{\mathcal{J}'}$. Then there is a $d \in \Delta^\mathcal{I}$ such that $d \in A^{\mathcal{I}'}$ and $d \notin A^{\mathcal{J}'}$. Since $\mathcal{I}' \Longrightarrow_\beta \mathcal{J}'$, there is a $\beta_j \in \{\beta_1, \dots, \beta_n\}$ such that $\neg A(a) \in \beta_j$ for some $a \in \mathsf{N_I}$ with $a^\mathcal{I} = d$ and for all i with $j < i \leq n$, we have $A(a) \notin \beta_i$. (Intuitively, it means that d is removed from A by β_j and never added afterwards.) However, together with $\mathcal{I} \Longrightarrow_\beta \mathcal{I}'$, such a β_j in β implies $d \notin A^{\mathcal{I}'}$, which contradicts the assumption.

"$\supseteq$": Assume that $A^{\mathcal{J}'} \not\subseteq A^{\mathcal{I}'}$. Then there is a $d \in \Delta^\mathcal{I}$ such that $d \in A^{\mathcal{J}'}$ and $d \notin A^{\mathcal{I}'}$. Since $\mathcal{I}' \Longrightarrow_\beta \mathcal{J}'$, there is a $\beta_j \in \{\beta_1, \dots, \beta_n\}$ such that $A(a) \in \beta_j$ for some $a \in \mathsf{N_I}$ with $a^\mathcal{I} = d$ and for all i with $j < i \leq n$, we have $\neg A(a) \notin \beta_i$. (Intuitively, it means that d is added into A by β_j and never removed afterwards.) However, together with $\mathcal{I} \Longrightarrow_\beta \mathcal{I}'$, such a β_j in β implies $d \in A^{\mathcal{I}'}$, which contradicts the assumption. ❑

It follows from Lemma 66 that each DL-LTL structure $\mathfrak{I}$ w.r.t. w is determined by the first $m + 2n$ interpretations in $\mathfrak{I}$ and thus it is enough to check satisfiability of φ only on those interpretations. Based on this observation, we can solve the satisfiability problem by the reduction to the consistency problem of ABoxes w.r.t. acyclic TBoxes:

- Construct an acyclic TBox $\mathcal{T}_\mathsf{red}$ and an ABox $\mathcal{A}_\mathsf{red}$ from $\mathcal{A}$, w, and φ; and
- Compute an ABox $\mathcal{A}_\varphi$ from φ by a tableau algorithm.

We show that φ is satisfiable w.r.t. $\mathcal{A}$ and w iff $\mathcal{A}_{\mathsf{red}} \cup \mathcal{A}_\varphi$ is consistent w.r.t. $\mathcal{T}_{\mathsf{red}}$. Intuitively, $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ are to ensure the semantics of updates and $\mathcal{A}_\varphi$ is to guarantee the semantics of LTL operators.

Without loss of generality, we can assume that there are no LTL negation signs in φ. First, we transform φ into LTL negation normal form (LTL-NNF), i.e., LTL negation signs occur only in front of ABox assertions.[1] To this end, we need to introduce the *release operator* R which is the dual operator of U. $\varphi \,\mathsf{R}\, \psi = \neg(\neg\varphi \,\mathsf{U}\, \neg\psi)$, i.e., $\mathfrak{I}, i \models \varphi \,\mathsf{R}\, \psi$ iff for all $m \geq i$, $\mathfrak{I}, m \models \psi$ or there exists a k such that $\mathfrak{I}, k \models \varphi$ and $\mathfrak{I}, j \models \psi$ for all j with $i \leq j \leq k$. First, by exhaustively applying the following rules, every DL-LTL formula φ can be transformed to an equivalent one in LTL-NNF.

$$\begin{array}{rclcrcl} \neg(\varphi \wedge \psi) & \rightsquigarrow & \neg\varphi \vee \neg\psi; & & \neg(\varphi \vee \psi) & \rightsquigarrow & \neg\varphi \wedge \neg\psi; \\ \neg(\varphi \,\mathsf{U}\, \psi) & \rightsquigarrow & \neg\varphi \,\mathsf{R}\, \neg\psi; & & \neg(\varphi \,\mathsf{R}\, \psi) & \rightsquigarrow & \neg\varphi \,\mathsf{U}\, \neg\psi; \\ \neg\mathsf{X}\varphi & \rightsquigarrow & \mathsf{X}\neg\varphi. & & & & \end{array}$$

Second, replace, respectively, $\neg(C(a))$ with $(\neg C)(a)$, $\neg(r(a,b))$ with $\neg r(a,b)$, and $\neg((\neg r)(a,b))$ with $r(a,b)$ after the LTL-NNF of φ is obtained. It is not hard to see that those replacements are satisfiability preserving and can be done in time polynomial in the size of φ. The size of the formula obtained is polynomial in the size of φ.

The construction of $\mathcal{T}_{\mathsf{red}}$ and $\mathcal{A}_{\mathsf{red}}$ is inspired from [BLM$^+$05]. Let Obj be the set of all the individual names in the input. Let Sub be the set of all the subconcepts in the input. We introduce the following concept names and role names:[2]

- For every $C \in \mathsf{Sub}$ and every $i \leq m + 2n - 1$, we introduce a concept name $T_C^{(i)}$ to represent C at the i-th time point.
- For every concept name A (role name r, respectively) and every $i \leq m + 2n - 1$, we introduce $A^{(i)}$ ($r^{(i)}$, respectively) to represent A (r, respectively) at the i-th time point, but only w.r.t. the named objects.
- We use a concept name N as abbreviation of the union of the individual names in Obj.

Let us start with constructing $\mathcal{T}_{\mathsf{red}}$ which consists of several components. The first component is $\mathcal{T}_N$.

$$\mathcal{T}_N = \{N \equiv \bigsqcup_{a \in \mathsf{Obj}} \{a\}\}.$$

For every $C \in \mathsf{Sub}$, there is a concept definition of $T_C^{(i)}$ in $\mathcal{T}_{\mathsf{Sub}}^{(i)}$. The concept definition of $T_C^{(i)}$ is defined inductively on the structure of C as described in Figure 18. We are now ready to assemble $\mathcal{T}_{\mathsf{red}}$:

$$\mathcal{T}_{\mathsf{red}} = \mathcal{T}_N \cup (\bigcup_{i=0}^{m+2n-1} \mathcal{T}_{\mathsf{Sub}}^{(i)}).$$

[1] It is required that φ is in LTL-NNF because $\mathcal{A}_\varphi$ is constructed by a tableau-based algorithm which works on DL-LTL formulas in LTL-NNF.

[2] Note that the auxiliary individual name a_{help} and ABox $\mathcal{A}_{\mathsf{aux}}$ and role names r_a for all $a \in \mathsf{Obj}$ used in [BLM$^+$05] are not necessary here since we consider only unconditional updates.

$$
\begin{array}{rcl}
T_A^{(i)} & \equiv & (N \sqcap A^{(i)}) \sqcup (\neg N \sqcap A^{(0)}) \text{ if } A \in \mathsf{N_C} \\
T_{\{a\}}^{(i)} & \equiv & \{a\} \\
T_{\neg C}^{(i)} & \equiv & \neg T_C^{(i)} \\
T_{C \sqcap D}^{(i)} & \equiv & T_C^{(i)} \sqcap T_D^{(i)} \\
T_{C \sqcup D}^{(i)} & \equiv & T_C^{(i)} \sqcup T_D^{(i)} \\
T_{\exists r.C}^{(i)} & \equiv & \Big(N \sqcap \big((\exists r^{(0)}.(\neg N \sqcap T_C^{(i)})) \sqcup (\exists r^{(i)}.(N \sqcap T_C^{(i)}))\big)\Big) \\
 & & \sqcup (\neg N \sqcap \exists r^{(0)}.T_C^{(i)}) \\
T_{\forall r.C}^{(i)} & \equiv & \Big(N \rightarrow \big((\forall r^{(0)}.(\neg N \rightarrow T_C^{(i)})) \sqcap (\forall r^{(i)}.(N \rightarrow T_C^{(i)}))\big)\Big) \\
 & & \sqcap (\neg N \rightarrow \forall r^{(0)}.T_C^{(i)}) \\
T_{(\geqslant n\ r\ C)}^{(i)} & \equiv & \Big(N \sqcap \bigsqcup_{0 \leq j \leq \min\{n, \#\mathsf{Obj}\}} \big((\geqslant j\ r^{(i)}\ (N \sqcap T_C^{(i)})) \sqcap \\
 & & \qquad (\geqslant (n-j)\ r^{(0)}\ (\neg N \sqcap T_C^{(i)}))\big)\Big) \\
 & & \sqcup (\neg N \sqcap (\geqslant n\ r^{(0)}\ T_C^{(i)})) \\
T_{(\leqslant n\ r\ C)}^{(i)} & \equiv & \Big(N \rightarrow \bigsqcap_{0 \leq j \leq \min\{n+1, \#\mathsf{Obj}\}} \big(\neg(\geqslant j\ r^{(i)}\ (N \sqcap T_C^{(i)})) \sqcup \\
 & & \qquad \neg(\geqslant (n-j)\ r^{(0)}\ (\neg N \sqcap T_C^{(i)}))\big)\Big) \\
 & & \sqcap (\neg N \rightarrow (\leqslant n\ r^{(0)}\ T_C^{(i)}))
\end{array}
$$

Figure 18: Concept definitions in $\mathcal{T}_{\mathsf{Sub}}^{(i)}$.

The TBoxes $\mathcal{T}_N$ and $\mathcal{T}_{\mathsf{sub}}^{(i)}$ ensure that for all concept $C \in \mathsf{Sub}$, at the i-th time point C is represented by the concept name $T_C^{(i)}$ and that in particular the interpretations of concept and role names remain unchanged by updates on the anonymous objects. The changes by updates on the named objects will be guaranteed by $\mathcal{A}_{\mathsf{red}}$. For every ABox assertion φ we define $\varphi^{(i)}$ as

$$
\varphi^{(i)} = \begin{cases} T_C^{(i)}(a) & \text{if } \varphi = C(a) \\ r^{(i)}(a,b) & \text{if } \varphi = r(a,b) \\ \neg r^{(i)}(a,b) & \text{if } \varphi = \neg r(a,b) \end{cases} \tag{8}
$$

For $1 \leq i \leq m + 2n - 1$, we define the ABox $\mathcal{A}_{\mathsf{rhs}}^{(i)}$ as follows:

$$
\mathcal{A}_{\mathsf{rhs}}^{(i)} = \{\psi^{(i)} \mid \psi \in w(i-1)\}.
$$

Intuitively, the ABox $\mathcal{A}_{\mathsf{rhs}}^{(i)}$ ensures that the effects of $w(i-1)$ hold at the i-th time point. For $1 \leq i \leq m + 2n - 1$, the ABox $\mathcal{A}_{\mathsf{min}}^{(i)}$ only contains

1. the following assertions for every $a \in \mathsf{Obj}$ and every concept name A which occurs in the input:

$$
\begin{array}{rcl}
a & : & (A^{(i-1)} \rightarrow A^{(i)}) \text{ if } \neg A(a) \notin w(i-1); \\
a & : & (\neg A^{(i-1)} \rightarrow \neg A^{(i)}) \text{ if } A(a) \notin w(i-1).
\end{array}
$$

2. the following assertions for all $a, b \in \mathsf{Obj}$ and every role name r which occurs in the input:

$$\begin{aligned} a &: (\exists r^{(i-1)}.\{b\} \rightarrow \exists r^{(i)}.\{b\}) \text{ if } \neg r(a,b) \notin w(i-1); \\ a &: (\forall r^{(i-1)}.\neg\{b\} \rightarrow \forall r^{(i)}.\neg\{b\}) \text{ if } r(a,b) \notin w(i-1). \end{aligned}$$

The ABox $\mathcal{A}^{(i)}_{\mathsf{min}}$ guarantees that the interpretations of concept names and role names remain intact on named objects if they are not affected by the update $w(i-1)$. The ABox $\mathcal{A}_{\mathsf{ini}}$ which assures that the ABox $\mathcal{A}$ is satisfied at time point 0 is defined as follows:

$$\mathcal{A}_{\mathsf{ini}} = \{\varphi^{(0)} \mid \varphi \in \mathcal{A}\}.$$

Then, we construct $\mathcal{A}_{\mathsf{red}}$ as follows:

$$\mathcal{A}_{\mathsf{red}} = \mathcal{A}_{\mathsf{ini}} \cup \bigcup_{i=1}^{m+2n-1} \mathcal{A}^{(i)}_{\mathsf{rhs}} \cup \bigcup_{i=1}^{m+2n-1} \mathcal{A}^{(i)}_{\mathsf{min}}.$$

As revealed in [BLM+05], from every interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{T}_{\mathsf{red}}$ and $\mathcal{I} \models \mathcal{A}_{\mathsf{red}}$ we can construct the crucial part of a DL-LTL structure w.r.t. w and vice versa.

Lemma 67. *Let $(\mathcal{A}, w, \varphi)$ be an input of the satisfiability problem. Let $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ be respectively the ABox, and the TBox obtained according to the above construction. Then, we have*

(a) for every $\mathcal{I}_0, \ldots, \mathcal{I}_{m+2n-1}$ such that $\mathcal{I}_0 \models \mathcal{A}$ and $\mathcal{I}_i \Longrightarrow_{w(i)} \mathcal{I}_{i+1}$ for every i with $0 \leq i < m+2n-1$, there exists an interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$, $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$, and for all $i \in \{0, \ldots, m+2n-1\}$ and for all assertions ψ in the input, $\mathcal{I}_i \models \psi$ iff $\mathcal{J} \models \psi^{(i)}$.

(b) for every interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$, $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$, there exist interpretations $\mathcal{I}_0, \ldots, \mathcal{I}_{m+2n-1}$ such that $\mathcal{I}_0 \models \mathcal{A}$, for every i with $0 \leq i < m+2n-1$, $\mathcal{I}_i \Longrightarrow_{w(i)} \mathcal{I}_{i+1}$, and for all $i \in \{0, \ldots, m+2n-1\}$ and for all assertions ψ in the input, $\mathcal{I}_i \models \psi$ iff $\mathcal{J} \models \psi^{(i)}$.

The tableau rules displayed in Figure 19 are designed to satisfy the semantics of LTL operators in the DL-LTL formula φ, where

- in the $\vee$-rule we have:

$$\begin{aligned} \mathcal{B}' &= (\mathcal{A} \setminus \{(\varphi_1 \vee \varphi_2)^{(i)}\}) \cup \{\varphi_1^{(i)}\}, \text{ and} \\ \mathcal{B}'' &= (\mathcal{A} \setminus \{(\varphi_1 \vee \varphi_2)^{(i)}\}) \cup \{\varphi_2^{(i)}\}; \end{aligned}$$

- in the U-rule$_1$ and the U-rule$_2$, we have:

$$\begin{aligned} \mathcal{B}_k &= (\mathcal{A} \setminus \{(\varphi_1 \,\mathsf{U}\, \varphi_2)^{(i)}\}) \cup \{\varphi_1^{(i)}, \ldots, \varphi_1^{(k-1)}, \varphi_2^{(k)}\} \\ &\quad \text{for all } k \text{ with } i \leq k < m+2n, \text{ and} \\ \mathcal{B}_k &= (\mathcal{A} \setminus \{(\varphi_1 \,\mathsf{U}\, \varphi_2)^{(i)}\}) \cup \{\varphi_1^{(i)}, \ldots, \varphi_1^{(m+2n-1)}, \varphi_1^{(m+n)}, \ldots, \varphi_1^{(k-1)}, \varphi_2^{(k)}\}, \\ &\quad \text{for all } k \text{ with } m+n \leq k < i; \end{aligned}$$

$$\frac{\mathcal{A} \in \mathfrak{S} \wedge (\varphi_1 \wedge \varphi_2)^{(i)} \in \mathcal{A}}{\mathcal{A} := (\mathcal{A} \setminus \{(\varphi_1 \wedge \varphi_2)^{(i)}\}) \cup \{\varphi_1^{(i)}, \varphi_2^{(i)}\}} \wedge\text{-rule}$$

$$\frac{\mathcal{A} \in \mathfrak{S} \wedge (\varphi_1 \vee \varphi_2)^{(i)} \in \mathcal{A}}{\mathfrak{S} := (\mathfrak{S} \setminus \{\mathcal{A}\}) \cup \{\mathcal{B}', \mathcal{B}''\}} \vee\text{-rule}$$

$$\frac{\mathcal{A} \in \mathfrak{S} \wedge (\mathsf{X}\varphi)^{(i)} \in \mathcal{A} \wedge i < m + 2n - 1}{\mathcal{A} := \mathcal{A} \setminus \{(\mathsf{X}\varphi)^{(i)}\} \cup \{\varphi^{(i+1)}\}} \mathsf{X}\text{-rule}_1$$

$$\frac{\mathcal{A} \in \mathfrak{S} \wedge (\mathsf{X}\varphi)^{(i)} \in \mathcal{A} \wedge i = m + 2n - 1}{\mathcal{A} := \mathcal{A} \setminus \{(\mathsf{X}\varphi)^{(i)}\} \cup \{\varphi^{(n+m)}\}} \mathsf{X}\text{-rule}_2$$

$$\frac{\mathcal{A} \in \mathfrak{S} \wedge (\varphi_1 \,\mathsf{U}\, \varphi_2)^{(i)} \in \mathcal{A} \wedge i \leq m + n}{\mathfrak{S} := \mathfrak{S} \setminus \{\mathcal{A}\} \cup \{\mathcal{B}_i, \ldots, \mathcal{B}_{m+2n-1}\}} \mathsf{U}\text{-rule}_1$$

$$\frac{\mathcal{A} \in \mathfrak{S} \wedge (\varphi_1 \,\mathsf{U}\, \varphi_2)^{(i)} \in \mathcal{A} \wedge i > m + n}{\mathfrak{S} := \mathfrak{S} \setminus \{\mathcal{A}\} \cup \{\mathcal{B}_{m+n}, \ldots, \mathcal{B}_{m+2n-1}\}} \mathsf{U}\text{-rule}_2$$

$$\frac{\mathcal{A} \in \mathfrak{S} \wedge (\varphi_1 \,\mathsf{R}\, \varphi_2)^{(i)} \in \mathcal{A} \wedge i \leq m + n}{\mathfrak{S} := \mathfrak{S} \setminus \{\mathcal{A}\} \cup \{\mathcal{B}_i, \ldots, \mathcal{B}_{m+2n-1}, \mathcal{B}_1^\infty\}} \mathsf{R}\text{-rule}_1$$

$$\frac{\mathcal{A} \in \mathfrak{S} \wedge (\varphi_1 \,\mathsf{R}\, \varphi_2)^{(i)} \in \mathcal{A} \wedge i > m + n}{\mathfrak{S} := \mathfrak{S} \setminus \{\mathcal{A}\} \cup \{\mathcal{B}_{m+n}, \ldots, \mathcal{B}_{m+2n-1}, \mathcal{B}_2^\infty\}} \mathsf{R}\text{-rule}_2$$

Figure 19: Tableau rules.

- and in the R-rule$_1$ and the R-rule$_2$ we have:

$$\begin{aligned}
\mathcal{B}_k &= (\mathcal{A} \setminus \{(\varphi_1 \,\mathsf{R}\, \varphi_2)^{(i)}\}) \cup \{\varphi_2^{(i)}, \ldots, \varphi_2^{(k)}, \varphi_1^{(k)}\}, \\
&\quad \text{for all } k \text{ with } i \leq k < m + 2n, \\
\mathcal{B}_1^\infty &= \{\varphi_2^{(i)}, \ldots, \varphi_2^{(m+2n-1)}\}, \\
\mathcal{B}_k &= (\mathcal{A} \setminus \{(\varphi_1 \,\mathsf{U}\, \varphi_2)^{(i)}\}) \cup \{\varphi_2^{(i)}, \ldots, \varphi_2^{(m+2n-1)}, \varphi_2^{(m+n)}, \ldots, \varphi_2^{(k)}, \varphi_1^{(k)}\}, \\
&\quad \text{for all } k \text{ with } n + m \leq k < i, \text{ and} \\
\mathcal{B}_2^\infty &= \{\varphi_2^{(m+n)}, \ldots, \varphi_2^{(m+2n-1)}\}.
\end{aligned}$$

As we can see from Figure 19, the tableau rules operate on a set of sets of DL-LTL formulas. Each formula is labeled with (i). Intuitively, the label stands for the time point, e.g., $\psi^{(i)}$ can be read as the formula ψ holds at time point i. We say that a tableau rule is *applied to a formula* ψ if ψ is the formula appearing explicitly in the upper part of the rule. We say that a set $\mathcal{B}$ of labeled formulas is *generated by an*

application of a tableau rule if $\mathcal{B}$ occurs explicitly in the appended set or gets assigned in the lower part of the rule. We exhaustively apply the tableau rules to $\mathfrak{S} = \{\{\varphi^{(0)}\}\}$ for a DL-LTL formula φ. The following lemma tells us that every application of a tableau rule to a formula preserves satisfiability of the formula:

Lemma 68. *Let $(\mathcal{A}, w, \varphi)$ be an input of the satisfiability problem. Let $\mathfrak{S}$ be the set in some status of the tableau algorithm starting with $\{\{\varphi^{(0)}\}\}$. Assume that $\mathfrak{S}'$ is obtained from $\mathfrak{S}$ by an application of one of the tableau rules in Figure 19 to some formula in $\mathcal{A}_l$ with $\mathcal{A}_l \in \mathfrak{S}$. Then for every DL-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\dots}$ w.r.t. w, the following statements are equivalent:*

- *$\mathfrak{I}, i \models \varphi$ for all $\varphi^{(i)} \in \mathcal{A}_l$.*
- *there exists a generated $\mathcal{B}_k \in \mathfrak{S}'$ by the application of the tableau rule such that $\mathfrak{I}, i \models \varphi$ for all $\varphi^{(i)} \in \mathcal{B}_k$.*

Proof. It is obvious for the $\wedge$-rule and the $\vee$-rule. By Lemma 66, $\mathfrak{I}$ is of the following form:

$$(\mathcal{I}_0, \dots, \mathcal{I}_{m+n}, \dots, \mathcal{I}_{m+2n-1}, \mathcal{I}_{m+n}, \dots, \mathcal{I}_{m+2n-1}, \mathcal{I}_{m+n}, \dots).$$

Thus, it follows immediately that the statements in this lemma are equivalent for the X-rule$_1$ and the X-rule$_2$.

U-rule$_1$: Suppose that this rule is applied to $(\varphi_1 \,\mathsf{U}\, \varphi_2)^{(i)} \in \mathcal{A}_l$. Then we know that for all $i \leq m+n$, $\mathfrak{I}, i \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff (by the semantics of U) there exists a $k \geq i$ such that $\mathfrak{I}, k \models \varphi_2$ and $\mathfrak{I}, j \models \varphi_1$ for all $i \leq j < k$ iff (from the form of $\mathfrak{I}$) there exists a k with $i \leq k < m+2n$ such that $\mathfrak{I}, k \models \varphi_2$ and $\mathfrak{I}, j \models \varphi_1$ for all j with $i \leq j < k$, i.e., there is one generated $\mathcal{B}_k$ such that $\mathfrak{I}, i \models \varphi$ for all $\varphi^{(i)} \in \mathcal{B}_k \setminus \mathcal{A}_l$.

U-rule$_2$: Suppose that this rule is applied to $(\varphi_1 \,\mathsf{U}\, \varphi_2)^{(i)} \in \mathcal{A}_l$. Then we know that for all $i > m+n$, $\mathfrak{I}, i \models \varphi_1 \,\mathsf{U}\, \varphi_2$ iff (by the semantics of U) there exists a $k \geq i$ such that $\mathfrak{I}, k \models \varphi_2$ and $\mathfrak{I}, j \models \varphi_1$ for all $i \leq j < k$ iff (from the form of $\mathfrak{I}$) there exists a k with $i \leq k \leq m+2n-1$ such that $\mathfrak{I}, k \models \varphi_2$ and $\mathfrak{I}, j \models \varphi_1$ for all j with $i \leq j < k$ or there exists a k with $m+n \leq k < i$ such that $\mathfrak{I}, k \models \varphi_2$ and $\mathfrak{I}, j \models \varphi_1$ for all j with $i \leq j \leq m+2n$ and for all j with $m+n \leq j < k$, i.e., there is one generated $\mathcal{B}_k$ such that $\mathfrak{I}, i \models \varphi$ for all $\varphi^{(i)} \in \mathcal{B}_k \setminus \mathcal{A}_l$.

Similarly, the form of $\mathfrak{I}$, together with the semantics of R, implies that the two statements in the lemma are equivalent if either of the R-rule$_1$ and the R-rule$_2$ is applied. ❑

After the tableau algorithm terminates with $\mathfrak{S}$, for every $\mathcal{A}$ in $\mathfrak{S}$, every labeled formula in $\mathcal{A}$ is an ABox assertion and the function defined in (8) can be applied to those assertions.[3] Thus, every $\mathcal{A}$ in $\mathfrak{S}$ can be viewed as an ABox. Then, we use the set $\mathfrak{S}$, together with the constructed $\mathcal{T}_{\mathsf{red}}$ and $\mathcal{A}_{\mathsf{red}}$ to decide whether φ is satisfiable w.r.t. $\mathcal{A}$ and w.

Lemma 69. *Let $\mathfrak{S}$ be the set when the tableau algorithm terminates. Then φ is satisfiable w.r.t. $\mathcal{A}$ and w iff there is an $\mathcal{A}_\varphi \in \mathfrak{S}$ such that $\mathcal{A}_{\mathsf{red}} \cup \mathcal{A}_\varphi$ is consistent w.r.t. $\mathcal{T}_{\mathsf{red}}$.*

[3]The termination of the tableau algorithm will be addressed later on when we analyze the complexity.

Proof. "⇒": If φ is satisfiable w.r.t. $\mathcal{A}$ and w then there is a DL-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\dots}$ w.r.t. w such that $\mathcal{I}_0 \models \mathcal{A}$ and $\mathfrak{I}, 0 \models \varphi$. By (a) of Lemma 67, there exists an interpretation $\mathcal{J}$ such that $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$, $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$ and for all $i \in \{0, \dots, m + 2n - 1\}$ and for all assertions φ in the input, $\mathcal{I}_i \models \varphi$ iff $\mathcal{J} \models \varphi^{(i)}$. By Lemma 68, $\mathfrak{I}, 0 \models \varphi$ implies that there exists $\mathcal{A}_\varphi \in \mathfrak{S}$ such that $\mathfrak{I}, i \models \varphi$ for all $\varphi^{(i)} \in \mathcal{A}_\varphi$. Since for every $\varphi^{(i)} \in \mathcal{A}_\varphi$, φ is an assertion, $\mathfrak{I}, i \models \varphi$ yields $\mathcal{I}_i \models \varphi$. Hence, $\mathcal{J} \models \mathcal{A}_\varphi$.

"⇐": Let $\mathcal{J}$ be a common model of $\mathcal{A}_{\mathsf{red}} \cup \mathcal{A}_\varphi$ and $\mathcal{T}_{\mathsf{red}}$ with some $\mathcal{A}_\varphi \in \mathfrak{S}$. Then it follows from (b) of Lemma 67 that there exist interpretations $\mathcal{I}_0, \dots, \mathcal{I}_{n+2m-1}$ such that $\mathcal{I}_0 \models \mathcal{A}$ and $\mathcal{I}_i \Longrightarrow_{w(i)} \mathcal{I}_{i+1}$ for all i with $0 \leq i < m+2n-1$ and for all assertions φ in the input, $\mathcal{I}_i \models \varphi$ iff $\mathcal{J} \models \varphi^{(i)}$. Define $\mathfrak{I}$ as follows:

$$(\mathcal{I}_0, \dots, \mathcal{I}_{m+n}, \dots, \mathcal{I}_{m+2n-1}, \mathcal{I}_{m+n}, \dots, \mathcal{I}_{m+2n-1}, \mathcal{I}_{m+n}, \dots).$$

By Lemma 66, $\mathfrak{I}$ is a DL-LTL structure w.r.t. w. $\mathcal{J} \models \mathcal{A}_\varphi$ implies that for all $\psi^{(i)} \in \mathcal{A}_\varphi$, $\mathcal{I}_i \models \psi$, i.e., $\mathfrak{I}, i \models \psi$. By Lemma 68, we have $\mathfrak{I}, 0 \models \varphi$. ❑

The *size of an input* $(\mathcal{A}, w, \varphi)$ is defined as $|\mathcal{A}|+|w|+|\varphi|$. Given an input $(\mathcal{A}, w, \varphi)$, the size of $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ is polynomial in the size of input and they can be constructed in time polynomial in the size of the input. This is independent of the codings of numbers in the number restrictions in the input [Mil08].

Let $\mathfrak{S}$ be the set obtained by exhaustively applying the tableau rules to $\{\{\varphi^{(0)}\}\}$. In general, the size of $\mathfrak{S}$ can be exponential in the size of the input. However, we need only one element $\mathcal{A}_\varphi$ in $\mathfrak{S}$ such that $\mathcal{A}_{\mathsf{red}} \cup \mathcal{A}_\varphi$ is consistent w.r.t. to $\mathcal{T}_{\mathsf{red}}$. For a DL-LTL formula φ, $\mathcal{A}_\varphi$ can be constructed in NPSpace since

- each application of a tableau rule generates at most $m + 2n$ (i.e., polynomially many) sets of labeled formulas;
- every labeled formula in generated sets is a strict subformula of the formula that the rule applies to and i in all labels (i) is never over $m + 2n - 1$;
- there is a tableau rule applicable iff there is an LTL operator in $\mathfrak{S}$.

By Savitch's theorem [Pap94], the construction of $\mathcal{A}_\varphi$ can be done in PSpace. Overall, $\mathcal{A}_{\mathsf{red}}$, $\mathcal{A}_\varphi$ and $\mathcal{T}_{\mathsf{red}}$ can be constructed in PSpace. Consistency checking of an $\mathcal{ALCQO}$-ABox w.r.t. an acyclic $\mathcal{ALCQO}$-TBox is in PSpace [BML+05] if the numbers in qualified number restriction are coded in unary. For $\mathcal{ALCIO}$, it is in ExpTime [ABM99]. For $\mathcal{ALCQIO}$, a fragment of C^2, it is in NExpTime [Tob00, PH05], even if the numbers are in binary coding. Thus, we obtain an upper bound of the satisfiability problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates in DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$.

Lemma 70. *The satisfiability problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates is*

- *in* PSpace *for* $\mathcal{ALCQO}$ *if the numbers in qualified number restrictions are coded in unary;*
- *in* ExpTime *for* $\mathcal{ALCIO}$*;*
- *in* NExpTime *for* $\mathcal{ALCQIO}$*.*

In what follows, we show that those upper bounds are tight by reducing the projection problem to the (un)satisfiability problem. It has been shown in [BLM$^+$05] that for DLs $\mathcal{L}$ between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$, the projection problem in $\mathcal{L}$ is as hard as the (in)consistency problem in $\mathcal{LO}$ even if every update is unconditional.

We can reduce the projection problem in $\mathcal{L}$ to the validity problem in $\mathcal{L}$. Let $\mathcal{A}$ be an ABox and $\mathcal{U}_i$ an unconditional update for all i with $1 \leq i \leq n$. It is easy to see that an assertion φ is a consequence of applying $\mathcal{U}_1 \ldots \mathcal{U}_n$ to $\mathcal{A}$ iff $\mathsf{X}^n\varphi$ is valid w.r.t. $\mathcal{A}$, and $\mathcal{U}_1 \ldots \mathcal{U}_n(\emptyset)^\omega$ (in which X^n is the abbreviation of n Xs). Thus, the complexity results about the projection problem in [BLM$^+$05] imply the following lemma:

Lemma 71. *The validity problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates is*

- PSpace*-hard for* $\mathcal{ALC}$*;*
- ExpTime*-hard for* $\mathcal{ALCI}$*;*
- *co*-NExpTime*-hard for* $\mathcal{ALCQI}$*.*

The above lemma does not rely on the coding of numbers. Recall that the validity problem can be further reduced to the (un)satisfiability problem. Thus,

Theorem 72. *The satisfiability problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates for a DL* $\mathcal{L}$ *is*

- PSpace*-complete if* $\mathcal{L}$ *is in* $\{\mathcal{ALC}, \mathcal{ALCO}, \mathcal{ALCQ}, \mathcal{ALCQO}\}$ *and the numbers in qualified number restriction are coded in unary;*
- ExpTime*-complete if* $\mathcal{L}$ *is in* $\{\mathcal{ALCI}, \mathcal{ALCIO}\}$*;*
- NExpTime*-complete if* $\mathcal{L}$ *is in* $\{\mathcal{ALCQI}, \mathcal{ALCQIO}\}$*.*

7.3 Conditional Updates

Lemma 66 is crucial for deciding the satisfiability problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates because it tells us that an infinite sequence of interpretations in a DL-LTL structure w.r.t. a sequence of unconditional updates consists of only finitely many interpretations. As a result, the semantics of temporal operators involving "infinite" meaning such as U and R can be easily checked. However, when we allow conditional updates, the corresponding claim of Lemma 66 does not hold any more. For example, if φ of φ/ψ in an update β_1 holds only after applying the update β_2, then ψ will not indeed take effect until the second execution of β_1, and thus, it cannot be guaranteed that the sequence of interpretations runs into a circle immediately after applying updates twice.

The key observation here is that there are only finitely many updates occurring in a Büchi sequence of updates. As a result, for every DL-LTL structure $\mathfrak{I}$ w.r.t. w, there are only *finitely* many interpretations appearing in $\mathfrak{I}$, although we are not

sure by which time point they have all shown up in $\mathfrak{I}$. Based on this observation, we combine the algorithm of the satisfiability problem of propositional LTL formulas and the one of the projection problem in DLs to solve the satisfiability problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates. It turns out that this satisfiability problem in DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ is still decidable. The solution even works for a more general satisfiability problem in which Büchi sequences of updates are given by a nondeterministic Büchi automaton [Büc60]:

Definition 73 (Nondeterministic Büchi Automaton)**.** A *(nondeterministic) Büchi automaton* $\mathcal{A}$ is a tuple $\mathcal{A} = (Q, \Sigma, \Delta, I, F_1, \ldots, F_n)$ with $n \geq 0$ where

- Q is a finite set of states;
- Σ is a finite alphabet;
- $\Delta \subseteq Q \times \Sigma \times Q$ is a transition relation;
- $I \subseteq Q$ is a set of initial states.
- for all i with $1 \leq i \leq n$, $F_i \subseteq Q$ is a set of final states.

Let $w = a_0 a_1 \cdots \in \Sigma^\omega$, a *run* of $\mathcal{A}$ on w is a sequence of $q_0 q_1 \cdots \in Q^\omega$ such that $q_0 \in I$ and $(q_i, a_i, q_{i+1}) \in \Delta$ for all $i \geq 0$. A run $q_0 q_1 \ldots$ is *accepting* if the set $\{i \mid q_i \in F_j\}$ is infinite for all $j \in \{1, \ldots, n\}$. The ω-language *accepted* by $\mathcal{A}$, denoted by $L_\omega(\mathcal{A})$, is defined by

$$L_\omega(\mathcal{A}) = \{w \in \Sigma^\omega \mid \text{ there is an accepting run of } \mathcal{A} \text{ on } w\}.$$

The *size of a Büchi-automaton* $\mathcal{A}$, denoted by $|\mathcal{A}|$, is defined as $\#Q$ plus the sum of the size of every element of Σ. △

We are ready now to introduce the generalized inference problems:

Definition 74 (Satisfiability and Validity with Generalized Inputs)**.** Let $\mathcal{A}$ be an ABox, φ a DL-LTL formula, and Σ a finite set of (possibly conditional) updates. Let $\mathcal{B} = (Q, \Sigma, I, \Delta, F)$ be a Büchi automaton.[4] Then we say that

- φ is *satisfiable w.r.t. $\mathcal{A}$ and $\mathcal{B}$* iff φ is satisfiable w.r.t. $\mathcal{A}$ and w for some $w \in L_\omega(\mathcal{B})$.
- φ is *valid w.r.t. $\mathcal{A}$ and $\mathcal{B}$* iff φ is valid w.r.t. $\mathcal{A}$ and w for all $w \in L_\omega(\mathcal{B})$.

△

Still, the validity problem of DL-LTL formulas w.r.t. ABoxes and Büchi automata can be reduced to the (un)satisfiability problem w.r.t. ABoxes and Büchi automata since $\neg\varphi$ is unsatisfiable w.r.t. $\mathcal{A}$ and $\mathcal{B}$ iff φ is valid w.r.t. $\mathcal{A}$ and $\mathcal{B}$. Thus, we can focus on the satisfiability problem. Moreover, for all $w = \alpha_1 \cdots \alpha_m (\beta_1 \cdots \beta_n)^\omega$ there exists a Büchi automaton $\mathcal{B}$ such that $L_\omega(\mathcal{B}) = \{w\}$. Such a $\mathcal{B}$ can be constructed in time

[4] Without loss of generality, we assume that $\mathcal{B}$ has only one set of final states [GPVW95, BK08].

polynomial in the size of w. It is obvious that φ is satisfiable w.r.t. $\mathcal{A}$ and w iff φ is satisfiable w.r.t. $\mathcal{A}$ and $\mathcal{B}$.

The problems in Definition 74 generalize the ones in the previous section: they allow for conditional updates, and instead of a fixed sequence of updates the sequences of updates in the input are the language accepted by a Büchi automaton.

We use a tuple $(\mathcal{A}, \mathcal{B}, \varphi)$ to denote an input of the satisfiability problem as in Definition 74. We first introduce some notions. Let Obj be the set of all individual names occurring in the input. We define

$$\begin{aligned}\mathcal{D} = \quad & \{\psi \mid \psi \text{ is an ABox assertion in the input }\} \cup \\ & \{A(a) \mid A \text{ is a concept name appearing in the input} \wedge a \in \mathsf{Obj}\} \cup \\ & \{r(a,b) \mid r \text{ is a role name appearing in the input} \wedge a, b \in \mathsf{Obj}\}.\end{aligned}$$

For every $\psi \in \mathcal{D}$, we introduce a propositional letter p_ψ. We use PL to denote the set of propositional letters corresponding to the ABox assertions in $\mathcal{D}$. For all DL-LTL formulas φ such that all assertions occurring in φ are in $\mathcal{D}$, we use the function $\mathsf{pl}(\varphi)$ to denote the propositional LTL formula obtained by replacing all assertions ψ in φ with p_ψ. We construct the following (propositional) LTL formula from φ and $\mathcal{A}$:

$$\hat{\varphi} = \mathsf{pl}(\varphi) \wedge \bigwedge_{\psi \in \mathcal{A}} \mathsf{pl}(\psi).$$

W.l.o.g., we assume that for all $p \in \mathsf{PL}$, p occurs in $\hat{\varphi}$. Otherwise we add $p \vee \neg p$ into $\hat{\varphi}$ as a conjunct. Moreover, we assume w.l.o.g. that there are no "$\vee$" signs in $\hat{\varphi}$ (otherwise replace $\hat{\varphi}_1 \vee \hat{\varphi}_2$ with $\neg(\neg\hat{\varphi}_1 \wedge \neg\hat{\varphi}_2)$).[5] It is clear that those replacements are satisfiability preserving and the size of the obtained formula is polynomial in the size of φ and can be constructed in time polynomial in the size of φ [BK08]. Let us recall how to decide the satisfiability problem of propositional LTL formulas with the help of Büchi automata according to [WVS83, VW86] where the notion of the closure of a propositional LTL formula is used.

Definition 75. The *closure* $\mathsf{cl}(\hat{\varphi})$ of a propositional LTL formula $\hat{\varphi}$ is defined inductively as follows:

- $\mathsf{cl}(p) := \{p, \neg p\}$;
- $\mathsf{cl}(\neg\hat{\varphi}) := \mathsf{cl}(\hat{\varphi})$;
- if $\hat{\varphi} \in \{\hat{\varphi}_1 \wedge \hat{\varphi}_2, \hat{\varphi}_1 \,\mathsf{U}\, \hat{\varphi}_2\}$, then $\mathsf{cl}(\hat{\varphi}) := \{\hat{\varphi}, \neg\hat{\varphi}\} \cup \mathsf{cl}(\hat{\varphi}_1) \cup \mathsf{cl}(\hat{\varphi}_2)$;
- $\mathsf{cl}(\mathsf{X}\hat{\varphi}) := \{\mathsf{X}\hat{\varphi}, \neg\mathsf{X}\hat{\varphi}\} \cup \mathsf{cl}(\hat{\varphi})$.

△

The closure $\mathsf{cl}(\hat{\varphi})$ of $\hat{\varphi}$ is the set of subformulas of $\hat{\varphi}$ and their negations. A type for $\hat{\varphi}$ is a maximal and "local" consistent subset of $\mathsf{cl}(\hat{\varphi})$. Formally, it is defined as follows:

Definition 76. A *type for* $\hat{\varphi}$ is a subset $T \subseteq \mathsf{cl}(\hat{\varphi})$ such that:

[5] We require this here because the decision procedure of satisfiability of propositional LTL formulas is defined on formulas with this form.

1. $\hat{\psi} \in T$ iff $\neg\hat{\psi} \notin T$, for all $\neg\hat{\psi} \in \mathsf{cl}(\hat{\varphi})$;
2. $\{\hat{\varphi}_1, \hat{\varphi}_2\} \subseteq T$ iff $\hat{\varphi}_1 \wedge \hat{\varphi}_2 \in T$, for all $\hat{\varphi}_1 \wedge \hat{\varphi}_2 \in \mathsf{cl}(\hat{\varphi})$;

We use $\mathsf{TP}(\hat{\varphi})$ to denote the set of types for $\hat{\varphi}$. For $T, T' \in \mathsf{TP}(\hat{\varphi})$, we write $T \rightarrow_{\mathsf{X}} T'$ if

- for all $\mathsf{X}\hat{\psi} \in \mathsf{cl}(\hat{\varphi})$, $\mathsf{X}\hat{\psi} \in T$ iff $\hat{\psi} \in T'$;
- for all $\hat{\varphi}_1 \,\mathsf{U}\, \hat{\varphi}_2 \in \mathsf{cl}(\hat{\varphi})$, we have $\hat{\varphi}_1 \,\mathsf{U}\, \hat{\varphi}_2 \in T$ iff
  - $\hat{\varphi}_2 \in T$ or
  - $\hat{\varphi}_1 \in T$ and $\hat{\varphi}_1 \,\mathsf{U}\, \hat{\varphi}_2 \in T'$.

$\triangle$

Let there be l until formulas (which are the formulas of the form $\psi_1 \,\mathsf{U}\, \psi_2$) in $\mathsf{cl}(\hat{\varphi})$ and assume that these formulas are linearly ordered. Let $\mathcal{P}(\mathsf{PL})$ be the set of subsets of PL. We define the Büchi automaton $\mathcal{A}_{\hat{\varphi}}$ as

$$\mathcal{A}_{\hat{\varphi}} = (\mathsf{TP}(\hat{\varphi}), \mathcal{P}(\mathsf{PL}), I_{\hat{\varphi}}, \Delta_{\hat{\varphi}}, F^1_{\hat{\varphi}}, \ldots, F^l_{\hat{\varphi}}),$$

where

- $I_{\hat{\varphi}} = \{T \in \mathsf{TP}(\hat{\varphi}) \mid \hat{\varphi} \in T\}$;
- $\Delta_{\hat{\varphi}} = \{(T, a, T') \mid a \cap \mathsf{cl}(\hat{\varphi}) = T \cap \mathsf{PL} \text{ and } T \rightarrow_{\mathsf{X}} T'\}$;
- $F^i_{\hat{\varphi}} = \{T \in \mathsf{TP}(\hat{\varphi}) \mid \hat{\varphi}_1 \,\mathsf{U}\, \hat{\varphi}_2 \notin T \text{ or } \hat{\varphi}_2 \in T\}$ if the i-th until formula in $\mathsf{cl}(\hat{\varphi})$ is $\hat{\varphi}_1 \,\mathsf{U}\, \hat{\varphi}_2$.

It has been shown that $\mathcal{A}_{\hat{\varphi}}$ has the following property [WVS83, VW86]:

Lemma 77. *For all $M = (X_i)_{i=0,1,\ldots} \in (\mathcal{P}(\mathsf{PL}))^\omega$, $M \in L_\omega(\mathcal{A}_{\hat{\varphi}})$ iff $M, 0 \models \hat{\varphi}$. If $T_0 T_1 \cdots$ is an accepting run of $\mathcal{A}_{\hat{\varphi}}$ on M, then for all $i \geq 0$ and all $\hat{\psi} \in \mathsf{cl}(\hat{\varphi})$, we have $\hat{\psi} \in T_i$ iff $M, i \models \hat{\psi}$.*

Let $\mathcal{S}$ be a subset of $\mathcal{P}(\mathsf{PL})$, i.e., a set of subsets of PL. We define a Büchi automaton $\mathcal{A}_{\mathcal{S}} = (Q_{\mathcal{S}}, \Sigma_{\mathcal{S}}, I_{\mathcal{S}}, \Delta_{\mathcal{S}}, F^1_{\mathcal{S}}, \ldots, F^l_{\mathcal{S}}, F^{l+1}_{\mathcal{S}})$ based on $\mathcal{A}_{\hat{\varphi}}$, $\mathcal{S}$, and $\mathcal{B}$ as following:

- $Q_{\mathcal{S}} = \mathsf{TP}(\hat{\varphi}) \times Q$;
- $\Sigma_{\mathcal{S}} = \mathcal{P}(\mathsf{PL}) \times \Sigma$;
- $I_{\mathcal{S}} = \{(T, q) \mid T \in I_{\hat{\varphi}} \wedge q \in I\}$;
- $((T_1, q_1), (X, \mathcal{U}), (T_2, q_2)) \in \Delta_{\mathcal{S}}$ iff the following conditions hold:
  - $X \in \mathcal{S}$;
  - $(T_1, X, T_2) \in \Delta_{\hat{\varphi}}$ and $(q_1, \mathcal{U}, q_2) \in \Delta$;
  - for all $\psi/\phi \in \mathcal{U}$, $\mathsf{pl}(\psi) \in T_1$ implies $\mathsf{pl}(\phi) \in T_2$.

  - for all $p \in \mathsf{PL}$ with $p = \mathsf{pl}(A(a))$ for some $A(a) \in \mathcal{D}$ with $A \in \mathsf{N_C}$,
    - if $p \in T_1$ and there is no $\varphi_1/\neg A(a) \in \mathcal{U}$ such that $\mathsf{pl}(\varphi_1) \in T_1$, then $p \in T_2$, and
    - if $\neg p \in T_1$ and there is no $\varphi_1/A(a) \in \mathcal{U}$ such that $\mathsf{pl}(\varphi_1) \in T_1$, then $\neg p \in T_2$
  - for all $p \in \mathsf{PL}$ with $p = \mathsf{pl}(r(a,b))$ for some $r(a,b) \in \mathcal{D}$ with $r \in \mathsf{N_R}$,
    - if $p \in T_1$ and there is no $\varphi_1/\neg r(a,b) \in \mathcal{U}$ such that $\mathsf{pl}(\varphi_1) \in T_1$, then $p \in T_2$, and
    - if $\neg p \in T_1$ and there is no $\varphi_1/r(a,b) \in \mathcal{U}$ such that $\mathsf{pl}(\varphi_1) \in T_1$, then $\neg p \in T_2$

- $F^j_{\mathcal{S}} = \{(T,q) \mid T \in F^j_{\hat{\varphi}}\}$ for all $1 \leq j \leq l$ and $F^{l+1}_{\mathcal{S}} = \{(T,q) \mid q \in F\}$.

The above automaton has two functionalities: the changes on the named objects respect the semantics of updates and the semantics of LTL operators in φ are taken care of. Let Sub be the set of all the subconcepts in the input. The interpretations of the anonymous objects are guaranteed to remain unchanged by an acyclic TBox $\mathcal{T}_{\mathsf{red}}$:

$$\mathcal{T}_{\mathsf{red}} = \mathcal{T}_N \cup (\bigcup_{i=1}^{k} \mathcal{T}^{(i)}_{\mathsf{Sub}}),$$

where $\mathcal{T}_N$ and $\mathcal{T}^{(i)}_{\mathsf{Sub}}$ defined as in the last section.

For all $1 \leq i \leq k$, we define

$$\mathcal{A}_i = \{\psi^{(i)} \mid \exists p \in X_i : \psi = \mathsf{pl}(p)\} \cup \{\neg\psi^{(i)} \mid \exists p \in \mathsf{PL} \setminus X_i : \psi = \mathsf{pl}(p)\},$$

where $\psi^{(i)}$ are defined as in the last section. After absorbing LTL negation signs into assertions, $\mathcal{A}_i$ can be viewed as an ABox. The realizability of the propositional types in DLs is ensured by the following ABox $\mathcal{A}_{\mathsf{red}}$:

$$\mathcal{A}_{\mathsf{red}} = \bigcup_{i=1}^{k} \mathcal{A}_i.$$

Lemma 78. *φ is satisfiable w.r.t. $\mathcal{A}$ and $\mathcal{B}$ iff there is a set $\mathcal{S} \subseteq \mathcal{P}(\mathsf{PL})$ such that $L_\omega(\mathcal{A}_{\mathcal{S}}) \neq \emptyset$ and $\mathcal{A}_{\mathsf{red}}$ is consistent w.r.t. $\mathcal{T}_{\mathsf{red}}$.*

Proof. "⇒": Suppose that φ is satisfiable w.r.t. $\mathcal{A}$ and $\mathcal{B}$. Then there exists a $w = \mathcal{U}_0\mathcal{U}_1 \cdots \in L_\omega(\mathcal{B})$ such that φ is satisfiable w.r.t. $\mathcal{A}$ and w, and thus there is a DL-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ w.r.t. w such that $\mathcal{I}_0 \models \mathcal{A}$ and $\mathfrak{I}, 0 \models \varphi$. We define X_i for all $i \geq 0$ and $\mathcal{S}$ as follows:

$$X_i = \{p \in \mathsf{PL} \mid \exists \psi \in \mathcal{D} : \mathsf{pl}(\psi) = p \wedge \mathcal{I}_i \models \psi\}.$$

$$\mathcal{S} = \{X_i \mid i = 0, 1, \ldots\}.$$

For all $i \geq 0$, we define

$$T_i = \{\hat{\psi} \in \mathsf{cl}(\hat{\varphi}) \mid \hat{\psi} = \mathsf{pl}(\psi) \wedge \mathfrak{I}, i \models \psi\}.$$

Since $\mathfrak{I}, 0 \models \varphi$ and $\mathcal{I}_0 \models \mathcal{A}$, it follows from the above definition that $\hat{\varphi} \in T_0$. Moreover, it is easy to see that for all $i \geq 0$, $T_i \in \mathsf{TP}(\hat{\varphi})$.

Since $w = \mathcal{U}_0\mathcal{U}_1 \cdots \in L_\omega(\mathcal{B})$, then there exists an accepting run $q_0q_1 \dots$ of $\mathcal{B}$ on w. Now we show that $(T_0, q_0)(T_1, q_1) \dots$ is a run of $\mathcal{A}_\mathcal{S}$ on $(X_0, \mathcal{U}_0)(X_1, \mathcal{U}_1) \dots$:

- For all $i \geq 0$, $T_i \in \mathsf{TP}(\hat{\varphi})$ and $q_i \in Q$, and thus $(T_i, q_i) \in Q_\mathcal{S}$.
- Since $\hat{\varphi} \in T_0$, we have $T_0 \in I_{\hat{\varphi}}$, which, together with $q_0 \in I$, implies $(T_0, q_0) \in I_\mathcal{S}$.
- For all $i \geq 0$, $((T_i, q_i), (X_i, \mathcal{U}_i), (T_{i+1}, q_{i+1})) \in \Delta_\mathcal{S}$. This holds since
  - $X_i \in \mathcal{S}$ is an direct consequence of the definition of $\mathcal{S}$.
  - $(q_i, \mathcal{U}_i, q_{i+1}) \in \Delta$ since $q_0q_1 \dots$ is an accepting run of $\mathcal{B}$ on w. Moreover, $(T_i, X_i, T_{i+1}) \in \Delta_{\varphi\mathcal{S}}$ since
    - $X_i \cap \mathsf{cl}(\hat{\varphi}) = T_i \cap \mathsf{PL}$: It suffices to show that for all $p \in \mathsf{PL}$ with $p = \mathsf{pl}(\psi)$, $\mathcal{I}_i \models \psi$ iff $p \in T_i$. This is implied by the definitions of T_i and X_i.
    - $T_i \rightarrow_\mathsf{X} T_{i+1}$ for all $i \geq 0$:
      - for all $\mathsf{X}\hat{\psi} \in \mathsf{cl}(\hat{\varphi})$ with $\mathsf{X}\hat{\psi} = \mathsf{pl}(\mathsf{X}\psi)$, $\mathsf{X}\hat{\psi} \in T_i$ iff (by the definition of T_i) $\mathfrak{I}, i \models \mathsf{X}\psi$ iff (by the semantics of DL-LTL formulas) $\mathfrak{I}, i+1 \models \psi$ iff (by the definition of T_{i+1}) $\hat{\psi} \in T_{i+1}$.
      - for all $\hat{\psi}\,\mathsf{U}\,\hat{\phi} \in \mathsf{cl}(\hat{\varphi})$ with $\hat{\psi}\,\mathsf{U}\,\hat{\phi} = \mathsf{pl}(\psi\,\mathsf{U}\,\phi)$, $\hat{\psi}\,\mathsf{U}\,\hat{\phi} \in T_i$ iff (by the definition of T_i) $\mathfrak{I}, i \models \psi\,\mathsf{U}\,\phi$ iff (by the semantics of DL-LTL formulas) there exists a $k \geq i$ such that $\mathfrak{I}, k \models \phi$ and for all j with $i \leq j < k$, $\mathfrak{I}, j \models \psi$ iff there exists a k such that $k = i$ and $\mathfrak{I}, k \models \phi$ or there exists a $k > i$ such that $\mathfrak{I}, k \models \phi$ and for all j with $i \leq j < k$, $\mathfrak{I}, j \models \psi$ iff (by the definition of T_{i+1}) $\hat{\phi} \in T_i$, or $\hat{\psi} \in T_i$ and $\mathfrak{I}, i+1 \models \psi\,\mathsf{U}\,\phi$ (i.e., $\hat{\psi}\,\mathsf{U}\,\hat{\phi} \in T_{i+1}$).
  - the other conditions follow immediately from the semantics of updates and the definition of T_i.

Moreover, the above sequence is accepting: Suppose that for some $j \in \{1, \dots, l+1\}$, the set $\{i \mid (T_i, q_i) \in F^j_\mathcal{S}\}$ is finite. Since $q_0q_1 \dots$ is an accepting run of $\mathcal{B}$, $\{i \mid (T_i, q_i) \in F^{l+1}_\mathcal{S}\}$ is infinite. Thus, j is in $\{1, \dots, l\}$. This yields that $\{i \mid T_i \in F^j_{\hat{\varphi}}\}$ is finite. Then there exists a natural number i_0 such that $T_i \notin F^j_{\hat{\varphi}}$ for all $i \geq i_0$. This implies for the j-th until formula $\hat{\phi}\,\mathsf{U}\,\hat{\psi}$, $\hat{\phi}\,\mathsf{U}\,\hat{\psi} \in T_i$ and $\hat{\psi} \notin T_i$ for all $i \geq i_0$. Suppose $\hat{\psi}\,\mathsf{U}\,\hat{\phi} = \mathsf{pl}(\psi\,\mathsf{U}\,\phi)$. Thus, $\mathfrak{I}, i \models \phi\,\mathsf{U}\,\psi$ and $\mathfrak{I}, i \not\models \psi$ for all $i \geq i_0$, which is a contradiction to the semantics.

It remains to show that $\mathcal{A}_\mathsf{red}$ is consistent w.r.t. $\mathcal{T}_\mathsf{red}$. It follows from the construction of $\mathcal{S}$ that there are only finitely many elements in $\mathcal{S}$. Suppose that $\mathcal{S} = \{X_1, \dots, X_k\}$. For each $\iota \geq 0$, we know that there is an $i_\iota \in \{1, \dots, k\}$ such that $X_{i_\iota} = \{p \in \mathsf{PL} \mid \exists \psi \in \mathcal{D} : \mathsf{pl}(\psi) = p \wedge \mathcal{I}_\iota \models \psi\}$. Conversely, for each $i \in \{1, \dots, k\}$, there is an $\iota \geq 0$ such that $i = i_\iota$. Let $\iota_1, \dots, \iota_k \in \{0, 1, \dots\}$ be

such that $i_{\iota_1} = 1, \ldots, i_{\iota_k} = k$. The interpretation $\mathcal{J}$ is obtained from $\mathcal{I}_{\iota_i}$ by the following construction:[6]

- $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}_{\iota_1}}$ $(= \Delta^{\mathcal{I}_{\iota_2}} = \cdots = \Delta^{\mathcal{I}_{\iota_k}})$,
- $a^{\mathcal{J}} = a^{\mathcal{I}_{\iota_1}}$ $(= a^{\mathcal{I}_{\iota_2}} = \cdots = a^{\mathcal{I}_{\iota_k}})$ for all $a \in \mathsf{N_I}$,
- $N^{\mathcal{J}} = \{a^{\mathcal{J}} \mid a \in \mathsf{Obj}\}$,
- $(A^{(i)})^{\mathcal{J}} = A^{\mathcal{I}_{\iota_i}}$ for all concept names A in the input and $1 \leq i \leq k$,
- $(A^{(0)})^{\mathcal{J}} = A^{\mathcal{I}_0}$ for all concept names A in the input,
- $(r^{(i)})^{\mathcal{J}} = r^{\mathcal{I}_{\iota_i}}$ for all role names r in the input and $1 \leq i \leq k$,
- $(r^{(0)})^{\mathcal{J}} = r^{\mathcal{I}_0}$ for all role names r in the input, and
- $(T_C^{(i)})^{\mathcal{J}} = C^{\mathcal{I}_{\iota_i}}$ for all $C \in \mathsf{Sub}$ and $1 \leq i \leq k$.

It follows from the definition of $\mathcal{J}$ that $\mathcal{J} \models \mathcal{T}_N$. By induction on the structure of C, it can be shown that for all $C \in \mathsf{Sub}$ and for all i with $1 \leq i \leq k$, $\mathcal{J}$ satisfies the concept definition of $T_C^{(i)}$ (cf. the proof of Lemma 3.2.3 in [Mil08] for details). Thus, we get $\mathcal{J} \models \mathcal{T}_{\mathsf{red}}$.

The definition of $\mathcal{J}$ implies that for all i with $1 \leq i \leq k$ and for all $p \in \mathsf{PL}$ with $p = \mathsf{pl}(\psi)$, $\mathcal{I}_{\iota_i} \models \psi$ iff $\mathcal{J} \models \psi^{(i)}$. The definition of X_i implies that $p \in X_i$ iff $\mathcal{I}_{\iota_i} \models \psi$. Thus, $\mathcal{J} \models \mathcal{A}_i$ for all i with $1 \leq i \leq k$. This yields $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$.

"$\Leftarrow$": Suppose that there is a set $\mathcal{S} \subseteq \mathcal{P}(\mathsf{PL})$ such that $L_\omega(\mathcal{A}_\mathcal{S}) \neq \emptyset$ and $\mathcal{A}_{\mathsf{red}}$ is consistent w.r.t. $\mathcal{T}_{\mathsf{red}}$. Thus, there exists a common model $\mathcal{J}$ of $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$. For $i \in \{1, \ldots, k\}$, we define $\mathcal{J}_i$ as follows:

- $\Delta^{\mathcal{J}_i} = \Delta^{\mathcal{J}}$,
- $a^{\mathcal{J}_i} = a^{\mathcal{J}}$ for every individual name $a \in \mathsf{N_I}$,
- $A^{\mathcal{J}_i} = (T_A^{(i)})^{\mathcal{J}}$ for every concept name A in the input, and
- $r^{\mathcal{J}_i} = (r^{(i)})^{\mathcal{J}} \cap (N^{\mathcal{J}} \times N^{\mathcal{J}}) \cup (r^{(0)})^{\mathcal{J}} \cap (\Delta^{\mathcal{J}} \times (\neg N)^{\mathcal{J}} \cup (\neg N)^{\mathcal{J}} \times \Delta^{\mathcal{J}})$ for every role name r in the input.

By induction on the structure of C, we can show that for each $C \in \mathsf{Sub}$, $C^{\mathcal{J}_i} = (T_C^{(i)})^{\mathcal{J}}$ (cf. the proof of Lemma 3.2.3 in [Mil08] for details). In addition, $\mathcal{J} \models \mathcal{A}_{\mathsf{red}}$ implies that for all i with $1 \leq i \leq k$, $\mathcal{J} \models \mathcal{A}_i$. Thus, for all $p \in \mathsf{PL}$ with $p = \mathsf{pl}(\psi)$, $p \in X_i$ iff $\mathcal{J}_i \models \psi$ (this is going to be used later on in the proof of (9)).

Since $L_\omega(\mathcal{A}_\mathcal{S}) \neq \emptyset$, then there is an accepting run $(T_0, q_0)(T_1, q_1)\ldots$ of $\mathcal{A}_\mathcal{S}$ on $(X_0, \mathcal{U}_0)(X_1, \mathcal{U}_1)\ldots$. We define $M = (X_\iota)_{\iota=0,1,\ldots}$ and $w = \mathcal{U}_0\mathcal{U}_1\ldots$. Then, it follows from the definition of $\mathcal{A}_\mathcal{S}$ that $T_0T_1\ldots$ is an accepting run of $A_{\hat{\varphi}}$ on M and $q_0q_1\ldots$

[6] It is possible that $\mathcal{I}_{\iota_1}, \ldots, \mathcal{I}_{\iota_k}$ do not respect the order in $\mathfrak{I}$, but this does not matter for our proof.

is an accepting run of $\mathcal{B}$ on w. The latter yields $w \in L_\omega(\mathcal{B})$. It remains to show that there exists a DL-LTL structure $\mathfrak{I} = (\mathcal{I}_i)_{i=0,1,\ldots}$ w.r.t. w such that $\mathcal{I}_0 \models \mathcal{A}$ and $\mathfrak{I}, 0 \models \varphi$.

By the definition of $\Delta_\mathcal{S}$, for all $\iota \geq 0$, there exists exactly one i_ι such that $1 \leq i_\iota \leq k$, $X_{i_\iota} \in \mathcal{S}$ and $X_\iota = X_{i_\iota}$. Consider the DL-LTL structure $\mathfrak{I} = (\mathcal{I}_\iota)_{\iota=0,1,\ldots}$ with $\mathcal{I}_\iota = \mathcal{J}_{i_\iota}$ for all $\iota \geq 0$.

It follows from Lemma 77 that for all $\iota \geq 0$, for all $\hat{\psi} \in \mathsf{cl}(\hat{\varphi})$, $\hat{\psi} \in T_\iota$ iff $M, \iota \models \hat{\psi}$. By induction on the structure of $\hat{\psi}$, we can show that

$$\forall \iota \geq 0. \forall \hat{\psi} \in \mathsf{cl}(\hat{\varphi}) \text{ with } \mathsf{pl}(\psi) = \hat{\psi} : M, \iota \models \hat{\psi} \text{ iff } \mathfrak{I}, \iota \models \psi. \tag{9}$$

Thus, we know that

$$\forall \iota \geq 0. \forall \hat{\psi} \in \mathsf{cl}(\hat{\varphi}) \text{ with } \mathsf{pl}(\psi) = \hat{\psi} : \hat{\psi} \in T_\iota \text{ iff } \mathfrak{I}, \iota \models \psi. \tag{10}$$

Now we show that for all $\iota \geq 0$, $\mathcal{I}_\iota \Longrightarrow_{\mathcal{U}_\iota} \mathcal{I}_{\iota+1}$. By the definition of $\mathcal{I}_\iota$, we know that all of $\mathcal{I}_\iota$ share the domain and interpretation of individuals. It follows from the definitions of $\mathcal{J}_1, \ldots, \mathcal{J}_k$ and the fact $\mathcal{J} \models \mathcal{T}_\mathsf{red}$ that for all $x, y \in \Delta^\mathcal{J}$, we have

- for each concept name A in the input, if $x \notin N^\mathcal{J}$, then $x \in A^{\mathcal{J}_i}$ iff $x \in (A^{(0)})^\mathcal{J}$; and
- for each role name r in the input, if $x \notin N^\mathcal{J}$ or $y \notin N^\mathcal{J}$, then $(x, y) \in r^{\mathcal{J}_i}$ iff $(x, y) \in (r^{(0)})^\mathcal{J}$.

This implies the anonymous objects respect the semantics of updates, which, together with (10) and the definition of $\Delta_\mathcal{S}$ yields that the conditions in Definition 17 are satisfied. Since $\hat{\varphi} \in T_0$ and T_0 is a type for $\hat{\varphi}$, we know that $\mathsf{pl}(\varphi) \in T_0$. By (10), $\mathfrak{I}, 0 \models \varphi$. Similarly, $\mathfrak{I}, 0 \models \psi$ for all $\psi \in \mathcal{A}$, i.e., $\mathcal{I}_0 \models \mathcal{A}$. ❑

The *size of an input* $(\mathcal{A}, \mathcal{B}, \varphi)$ is defined as $|\mathcal{A}| + |\mathcal{B}| + |\varphi|$. Given an input $(\mathcal{A}, w, \varphi)$ with the size n, $\mathcal{P}(\mathsf{PL})$ can be constructed in time exponential in n. A subset $\mathcal{S}$ of $\mathcal{P}(\mathsf{PL})$ can be constructed in nondeterministic time exponential in n and the size of $\mathcal{S}$ is bounded exponentially in n. The size of $\mathcal{A}_\mathcal{S}$ is bounded exponentially in n and it can be constructed in time exponential in n after $\mathcal{S}$ is given. The emptiness problem of $\mathcal{A}_\mathcal{S}$ can then be decided in time exponential in n [EL85b, EL85a]. $\mathcal{A}_\mathsf{red}$ and $\mathcal{T}_\mathsf{red}$ can be constructed in time exponential in n and the size of $\mathcal{A}_\mathsf{red}$ and $\mathcal{T}_\mathsf{red}$ is bounded exponentially in n.

For the DLs in which consistency checking of an ABox together with an acyclic TBox can be decided in PSpace (ExpTime and NExpTime, respectively), we do the following to decide whether φ is satisfiable w.r.t. $\mathcal{A}$ and $\mathcal{B}$:

1. Guess $\mathcal{S}$ in NExpTime.
2. Construct $\mathcal{A}_\mathcal{S}$ and decide $L_\omega(\mathcal{A}_\mathcal{S}) = \emptyset$ in ExpTime.
3. Construct $\mathcal{A}_\mathsf{red}$ and $\mathcal{T}_\mathsf{red}$ and decide its consistency in ExpSpace (2-ExpTime and 2-NExpTime).

Overall, the satisfiability problem can be decided in ExpSpace (2-ExpTime and 2-NExpTime, respectively).

Lemma 79. *The satisfiability problem of DL-LTL formulas w.r.t. ABoxes and Büchi automata is*

- *in* ExpSpace *for* $\mathcal{ALCQO}$ *if the numbers in qualified number restrictions are coded in unary;*
- *in 2-*ExpTime *for* $\mathcal{ALCIO}$*;*
- *in 2-*NExpTime *for* $\mathcal{ALCQIO}$*.*

The above upper bounds hold for the complemented problem of the validity problem of DL-LTL formulas w.r.t. ABoxes and Büchi automata.

Acyclic TBoxes can be introduced as domain constraints and updates can be extended to the actions with pre-conditions and occlusions as defined in [BLM$^+$05]. With those, the upper bound of the computational complexity of the inference problems presented in this section will not change [BLLuM05]. In order to keep uniformity, we focus on *updates* of ABoxes *without TBoxes*.

Comparing to the complexity of (non-)projection, there exists an exponential blowup since the size of the constructed $\mathcal{A}_{\mathsf{red}}$ and $\mathcal{T}_{\mathsf{red}}$ is exponential in the size of the input. The lower bounds are still open in the case that one of conditional updates, Büchi automata, and occlusions is allowed in the satisfiability problem. Notice that occlusions do not make the planning problem harder than the projection problem [Mil07] and that, to the best of our knowledge, there is no known solution to the planning problem when conditional actions are allowed which has the same upper bound as the projection problem. In this sense, allowing for unconditional actions with occlusions could be the first step to go to look for a better algorithm.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In this thesis, we have investigated updates of ABoxes in DLs and analyzed their computational behavior. The main motivation for this endeavor is to establish the theoretical foundations of progression in action theory based on DLs and to provide support for reasoning about action in DLs. We introduced four forms of updates of ABoxes under the Winslett's possible model approach semantics: logical updates, approximate updates, projective updates, and approximate projective updates. Logical updates are the most exact updates in the sense that they capture precisely the set of updated models and projective updates are exact with a restricted signature. Both of them are independent of the underlying DL. Approximate updates preserve the logical consequences of logical updates in a fixed DL and approximate projective updates preserve the logical consequences in a fixed DL w.r.t. a restricted signature.

We presented two algorithms for $\mathcal{ALCQIO}^{@}$ and its fragments: one is to compute logical updates and the other is to compute projective updates. The size of the computed logical update of an ABox $\mathcal{A}$ with an update $\mathcal{U}$ is exponential both in the size of $\mathcal{A}$ and in the size of $\mathcal{U}$. We show that the exponential blowup in the size of the whole input cannot be completely avoided unless the complexity classes PTime and NC coincide, which is believed to be as unlikely as PTime = NP. If we compute logical updates iteratively, the exponential blowups will not add up. The size of the projective update built with our algorithm is both polynomial in the size of $\mathcal{A}$ and in the size of $\mathcal{U}$. Differing from logical updates, the construction of iterative projective updates will change the computational behavior because it gets an exponential blowup in the size of $\mathcal{A}$. In order to avoid this, we propose to keep the original ABox $\mathcal{A}$ and the history of updates $\mathcal{U}_1 \cdots \mathcal{U}_n$ in memory when we compute projective updates. When a new update $\mathcal{U}_{n+1}$ arrives, instead of computing the update iteratively we compute the projective update from $\mathcal{A}$ and $\mathcal{U}_1 \cdots \mathcal{U}_{n+1}$ directly. The size of the projective update computed this way is polynomial both in the size of $\mathcal{A}$ and the size of $\mathcal{U}_1 \cdots \mathcal{U}_{n+1}$.

We also explore the relationship between the expressiveness of DLs and the existence of updates in them. The positive results shown by our algorithms are that logical updates exist for DLs between $\mathcal{ALCO}^{@}$ and $\mathcal{ALCQIO}^{@}$ and that projective

updates exist for DLs between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}^{@}$. The negative results are that without either the @ constructor or nominals approximate updates do not exist and that without nominals projective updates do not exist. More precisely, we proved that approximate updates do not exists for DLs between $\mathcal{ALCO}$ and $\mathcal{ALCQIO}$ and DLs between $\mathcal{ALC}$ and $\mathcal{ALCNI}^{@}$, and projective updates do not exist for DLs between $\mathcal{ALC}$ and $\mathcal{ALCNI}^{@}$.

The algorithms have been implemented and their performance compared. The implementation of the projection problem based on the approach similar to regression has been compared to the one employing updated ABoxes. Logical updates and projective updates can be computed efficiently but the time of reasoning with them is mainly dominated by their sizes. With the optimizations, the implementation based on logical updates performed better even than projective updates, especially for the inputs with big ABoxes. The implementation based on the approach similar to regression is more efficient when the sequence of updates is long.

Another problem investigated in this thesis is verifying DL-LTL formulas. We introduced the satisfiability problem and the validity problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates. The latter can be reduced to the complement problem of the former. It turned out that the satisfiability problem has the same computational complexity as the projection problem if only unconditional updates are allowed. For conditional updates, we showed that the satisfiability problem is still decidable by presenting an algorithm to solve the satisfiability problem which works even for a more general problem in which the infinite sequence of updates is given via a Büchi automaton.

8.2 Future Work

There are at least the following directions for future work. The first one is to find answers to the open problems left in this thesis. Secondly, one can restrict the ABox update problem for better computational behavior. The third one is to extend the problem in order to fit a more general setting.

We have shown that for logical updates, the exponential blowup in the size of the whole input most likely cannot be completely avoided. It is still open whether we can avoid a blowup in the original ABox. Concerning the existence of updates, approximate updates present no more positive results than logical updates. One can look for a better construction of approximate updates other than computing them via logical updates or enhance the lower bound of logical updates to approximate updates. Although projective updates exhibit theoretically computational behavior, they did not perform better than the optimized construction of logical updates. For this reason, optimizations of the former are strongly required. One can also work on the questions listed in Table 1 and Table 2 in Section 5.3. We can see that we still have little knowledge about approximate projective updates. Since the other stronger form of updates had some difficulty with testing particularly for iterative updates, exploring a direct construction of approximate projective updates might give rise to a more satisfactory solution.

As we have seen in Section 3.3, allowing only concept literals to occur in updates or building updates in DLs with role constructors will avoid the exponential blowup of logical updates in the size of ABoxes. One can try to find similar restrictions on other kinds of updates to achieve smaller updates. The state-of-the-art reasoners supporting role constructors are also strongly required since logical updates are very succinct with them. For DLs in which updates do not exist, one can look for the exact corresponding syntactical conditions. Alternatively, one can investigate updates in $\mathcal{EL}^{++}$ which is a family of lightweight DLs [BBL05]. Note that it is unlikely to get an algorithm which computes updates of $\mathcal{EL}$-ABoxes in a polynomial time since the projection problem in $\mathcal{EL}$ is already co-NP-hard. However, spending more time on achieving the updates of polynomial size may be still worth doing in practice, because reasoning in $\mathcal{EL}$ is tractable.

Another possible extension of the ABox update problem is to search for a satisfactory semantics for updates containing complex concepts or roles, maybe even for GCIs as domain constaints. This is a challenging work. In action theories, there are proposals for this employing circumscription [McC86, Lif94] to minimize the changes on names. Circumscription in DLs is investigated in [BLW09]. The question is how to apply their results to obtain a decent semantics for updates or actions.

Comparing the satisfiability problem of DL-LTL formulas w.r.t. ABoxes and Büchi sequences of updates to the (non-)projection problem, there exists an exponential blowup in the complexity of the algorithm presented in Section 7.3 for DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$. The algorithm can be easily extended to actions defined in [BLM$^+$05]. It is still an open problem whether or not those upper bounds are optimal.

Bibliography

[ABHM03] C. Areces, P. Blackburn, B. M. Hernandez, and M. Marx. Handling Boolean ABoxes. In *Proceedings of the 2003 International Workshop on Description Logics (DL-2003)*. 2003. ↩ p. 8, 20

[ABM99] C. Areces, P. Blackburn, and M. Marx. A Road-Map on Complexity for Hybrid Logics. In *CSL: 13th Workshop on Computer Science Logic*. LNCS, Springer-Verlag, 1999. ↩ p. 108

[AdR01] C. Areces and M. de Rijke. From Description to Hybrid Logics, and Back. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, eds., *Advances in Modal Logic. Volume 3*. CSLI Publications, 2001. ↩ p. 8, 20

[AGM85] C. Alchourrón, P. Gärdenfors, and D. Makinson. On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. *Journal of Symbolic Logic*, 50:510–530, 1985. ↩ p. 6

[BBL05] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ Envelope. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence IJCAI-05*. Morgan-Kaufmann Publishers, Edinburgh, UK, 2005. ↩ p. 10, 121

[BCM+03] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003. ↩ p. 1, 6, 18, 22

[BdRV01] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, Cambridge, England, 2001. ISBN 0 521 52714 7 (pbk). ↩ p. 3, 27

[BGL08] F. Baader, S. Ghilardi, and C. Lutz. LTL over Description Logic Axioms. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR-2008)*. 2008. ↩ p. 10, 11, 86, 100, 101

[BH93] G. Brewka and J. Hertzberg. How to Do Things with Worlds: on Formalizing Actions and Plans. *Journal of Logic and Computation*, 3(5):517–532, 1993. ↩ p. 4, 23

[BHS05] F. Baader, I. Horrocks, and U. Sattler. Description Logics as Ontology Languages for the Semantic Web. In D. Hutter and W. Stephan, eds., *Mechanizing Mathematical Reasoning: Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, volume 2605 of *Lecture Notes in Artificial Intelligence*, pp. 228–248. Springer-Verlag, 2005. ↩ p. 3

[BK08] C. Baier and J. P. Katoen. *Principles of Model Checking*. MIT Press, New York, 2008. ISBN 978-0-262-02649-9. ↩ p. 11, 99, 110, 111

[BLHL01] Berners-Lee, T. Hendler, and J. Lassila. The Semantic Web. *Scientific American*, 2001. ↩ p. 3

[BLLuM05] F. Baader, H. Liu, C. Lutz, and A. ul Mehdi. Integrate Action Formalisms into Linear Temporal Description Logics. LTCS-Report LTCS-09-03, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2005. See http://lat.inf.tu-dresden.de/research/reports.html. ↩ p. 101, 117

[BLM+05] F. Baader, C. Lutz, M. Miličić, U. Sattler, and F. Wolter. Integrating Description Logics and Action Formalisms: First Results. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-2005)*. Pittsburgh, PA, USA, 2005. ↩ p. 5, 6, 10, 12, 19, 23, 31, 83, 94, 99, 101, 103, 105, 109, 117, 121

[BLW09] P. Bonatti, C. Lutz, and F. Wolter. The Complexity of Circumscription in Description Logic. *Journal of Artificial Intelligence Research*, 2009. ↩ p. 121

[BML+05] F. Baader, M. Miličić, C. Lutz, U. Sattler, and F. Wolter. Integrating Description Logics and Action Formalisms for Reasoning about Web Services. LTCS-Report LTCS-05-02, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2005. See http://lat.inf.tu-dresden.de/research/reports.html. ↩ p. 9, 108

[Bon07] Y. Bong. *Description Logic ABox Updates Revisited*. Master thesis, TU Dresden, Germany, 2007. ↩ p. 8, 9, 67, 80

[Bor85] A. Borgida. Language Features for Flexible Handling of Exceptions in Information Systems. *ACM Transactions on Database Systems*, 10:565–603, 1985. ↩ p. 4

[Bor96] A. Borgida. On the Relative Expressiveness of Description Logics and Predicate Logics. *Artificial Intelligence*, 82(1 - 2):353–367, 1996. ↩ p. 50

[BP08] F. Baader and R. Peñaloza. Automata-Based Axiom Pinpointing. In A. Armando, P. Baumgartner, and G. Dowek, eds., *Proceedings of the*

4th International Joint Conference on Automated Reasoning (IJCAR-2008), volume 5195 of *Lecture Notes in Artificial Intelligence*, pp. 226–241. Springer, 2008. ↩ p. 90

[BS85] R. J. Brachman and J. G. Schmolze. An Overview of the KL-ONE Knowledge Representation System. *Cognitive Science*, 9(2):171–216, 1985. ↩ p. 1

[Büc60] J. R. Büchi. On a Decision Method in Restricted Second Order Arithmetic. In *Proceedings of the 1960 International Congress of Logic, Methodology and Philosophy of Science*, pp. 1–12. Stanford University Press, 1960. ↩ p. 110

[Byl94] T. Bylander. The Computational Complexity of Propositional STRIPS Planning. *Artificial Intelligence*, 69(1–2):165–204, 1994. ↩ p. 5

[CDLS99] M. Cadoli, F. M. Donini, P. Liberatore, and M. Schaerf. The Size of a Revised Knowledge Base. *Artificial Intelligence*, 115(1):25–64, 1999. ↩ p. 8, 30, 45, 49

[Cra57] W. Craig. Three Uses of the Herbrand-Gentzen Theorem in Relating Model Theory and Proof Theory. *Journal of Symbolic Logic*, 22(3):269–285, 1957. ↩ p. 30

[D'A98] G. D'Agostino. *Modal Logic and Non-Well-Founded Set Theory: Translation, Bisimulation, Interpolation*. Ph.D. thesis, Universiteit van Amsterdam, 1998. ↩ p. 46

[Dal88] M. Dalal. Investigations into a Theory of Knowedge Base Revision. Preliminary Report. In *Proceedings of AAAI-88, 7th Conference of the American Association for Artificial Intelligence*, pp. 475–479. 1988. ↩ p. 4

[Dav93] M. Davis. First Order Logic. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, eds., *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume I, pp. 31–65. Oxford University Press, 1993. ↩ p. 3

[DLB+09a] C. Drescher, H. Liu, F. Baader, S. Guhlemann, U. Petersohn, P. Steinke, and M. Thielscher. Putting ABox Updates into Action. In *The Seventh International Symposium on Frontiers of Combining Systems (FroCoS-2009)*. LNCS, Springer-Verlag, 2009. ↩ p. 13, 45, 51, 85, 93, 94

[DLB+09b] C. Drescher, H. Liu, F. Baader, P. Steinke, and M. Thielscher. Putting ABox Updates into Action. In *Proceedings of the 8th IJCAI International Workshop on Nonmontonic Reasoning, Action and Change (NRAC-2009)*. 2009. ↩ p. 13

[DLL62] M. Davis, G. Logemann, and D. Loveland. A Machine Program for Theorem-Proving. *Communications of the ACM*, 5(7):394–397, 1962. ISSN 0001-0782. ↩ p. 86

[DLMB98] P. Doherty, W. Lukaszewicz, and E. Madalinska-Bugaj. The PMA and Relativizing Change for Action Update. In *In Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning (KR-1998*, pp. 258–269. Morgan Kaufmann, 1998. ↩ p. 4, 23

[dMB08] L. de Moura and N. Bjørner. Z3: An Efficient SMT Solver. In C. R. Ramakrishnan and J. Rehof, eds., *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, (TACAS-2008)*, volume 4963 of *Lecture Notes in Computer Science*, pp. 337–340. Springer, 2008. ↩ p. 86

[DP60] M. Davis and H. Putnam. A Computing Procedure for Quantification Theory. *Journal of ACM*, 7(3):201–215, 1960. ISSN 0004-5411. ↩ p. 86

[DT07] C. Drescher and M. Thielscher. Integrating Action Calculi and Description Logics. In *KI '07: Proceedings of the 30th annual German conference on Advances in Artificial Intelligence*, pp. 68–83. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 978-3-540-74564-8. ↩ p. 9

[ECL09] ECLiPSe Implementors Group. *ECLiPSe User Manual*, 2009. `http://www.eclipse-clp.org`. ↩ p. 84

[EG92] T. Eiter and G. Gottlob. On the Complexity of Propositional Knowledge Base Revision, Updates and Counterfactuals. In *ACM Symposium on Principles of Database Systems*. 1992. ↩ p. 4

[EL85a] E. A. Emersona and C.-L. Lei. Modalities for Model Checking: Branching Time Strikes Back. In *Conference Record of the Twelfth Annual ACM Symposium on Principles of Programming Languages*, pp. 84–96. ACM, 1985. ↩ p. 116

[EL85b] E. A. Emersona and C.-L. Lei. Temporal Model Checking under Generalized Fairness Constraints. In *Proceedings of the Eighteenth Hawaii International Conference on System Sciences*, volume 1, pp. 277–288. Western Periodicals Company, North-Holland, CA, 1985. ↩ p. 116

[ES03] N. Een and N. Sörensson. An Extensible SAT-solver. In *International Conference on Theory and Applications of Satisfiability Testing (SAT), LNCS*, volume 6. 2003. ↩ p. 86, 87

[Flo06] G. Flouris. *On Belief Change and Ontology Evolution*. Ph.D. thesis, University of Crete, 2006. ↩ p. 6

[FPA05] G. Flouris, D. Plexousakis, and G. Antoniou. On Applying the AGM Theory to DLs and OWL. In *International Semantic Web Conference*, pp. 216–231. 2005. ↩ p. 6

[FPA06] G. Flouris, D. Plexousakis, and G. Antoniou. Evolving Ontology Evolution. In *SOFSEM*, pp. 14–29. 2006. ↩ p. 6

[FUV83] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the Semantics of Updates in Databases. In *Proceedings of the 2nd ACM SIGACT-SIGMOD symposium on Principles of database systems (PODS-1983)*, pp. 352–365. ACM, New York, NY, USA, 1983. ISBN 0-89791-097-4. ↩ p. 4

[Gär88] P. Gärdenfors. *Knowledge in Flux*. The MIT Press, Cambridge, Massachusetts, 1988. ↩ p. 4

[Gin86] M. L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30:35–79, 1986. ↩ p. 4

[GLPR06] G. D. Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the Update of Description Logic Ontologies at the Instance Level. In *Proceedings of the 21st National Conference on Artificial Intelligence*. 2006. ↩ p. 9

[GLPR07] G. D. Giacomo, M. Lenzerini, A. Poggi, and R. Rosati. On the Approximation of Instance Level Update and Erasure in Description Logics. In *Proceedings of the 22nd National Conference on Artificial Intelligence*. 2007. ↩ p. 9

[GOR97] E. Grädel, M. Otto, and E. Rosen. Two-Variable Logic with Counting is Decidable. In *Proceedings of Twelfth IEEE Symposium on Logic in Computer Science (LICS-1997)*. 1997. ↩ p. 50, 51

[Göt09] D. Götzmann. *Spartacus: A Tableau Prover for Hybrid Logic*. Master thesis, Saarland University, Germany, 2009. ↩ p. 8, 91

[GPVW95] R. Gerth, D. Peled, M. Vardi, and P. Wolper. Simple On-the-fly Automatic Verification of Linear Temporal Logic. In *Protocol Specification Testing and Verification*, pp. 3–18. Chapman & Hall, Warsaw, Poland, 1995. ↩ p. 110

[Her96] A. Herzig. The PMA Revisited. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR-1996)*, pp. 40–50. 1996. ↩ p. 4, 23

[HM01] V. Haarslev and R. Möller. RACER System Description. In R. Goré, A. Leitsch, and T. Nipkow, eds., *International Joint Conference on Automated Reasoning, (IJCAR-2001)*, pp. 701–705. Springer-Verlag, Siena, Italy, 2001. ↩ p. 3

[HPS03] I. Horrocks and P. F. Patel-Schneider. Reducing OWL Entailment to Description Logic Satisfiability. In D. Fensel, K. Sycara, and J. Mylopoulos, eds., *Proceedings of the 2nd International Semantic Web Conference (ISWC-2003)*, volume 2870 of *Lecture Notes in Computer Science*, pp. 17–29. Springer, 2003. ISBN 3-540-20362-1. ↩ p. 3

[HPSMW07] I. Horrocks, P. F. Patel-Schneider, D. L. McGuinness, and C. A. Welty. OWL: a Description Logic Based Ontology Language for the Semantic Web. In F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, eds., *The Description Logic Handbook: Theory, Implementation, and Applications (2nd Edition)*, chapter 14. Cambridge University Press, 2007. ↩ p. 3

[HWKP06] C. Halaschek-Wiener, Y. Katz, and B. Parsia. Belief Base Revision for Expressive Description Logics. In *OWL: Experiences and Direction (OWLED) Workshop*. 2006. ↩ p. 6

[KM89] H. Katsuno and A. O. Mendelzon. A Unified View of Propositional Knowledge Base Updates. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI-1989)*, pp. 1413–1419. 1989. ↩ p. 4

[KM92] H. Katsuno and A. Mendelzon. On the Difference Between Updating a Knowledge Base and Revising It. In P. Gärdenfors, ed., *Belief Revision*, pp. 183–203. Cambridge University Press, Cambridge, 1992. ↩ p. 4, 6

[Lif94] V. Lifschitz. Circumscription. In D. M. Gabbay, C. J. Hogger, and J. A. Robinson, eds., *Handbook of Logic in Artificial Intelligence and Logic Programming*, pp. 297–352. Clarendon Press, Oxford, UK, 1994. ↩ p. 121

[Lin95] F. Lin. Embracing Causality in Specifying the Indirect Effects of Actions. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence IJCAI-95*, pp. 1985–1993. 1995. ↩ p. 4

[Lin96] F. Lin. Embracing Causality in Specifying the Indeterminate Effects of Actions. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-1996)*, pp. 670–676. 1996. ↩ p. 4

[LL09] Y. Liu and G. Lakemeyer. On First-Order Definability and Computability of Progression for Local-Effect Actions and Beyond. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI-2009)*. Morgan Kaufmann, 2009. ↩ p. 5

[LLM08] H. Liu, C. Lutz, and M. Miličić. The Projection Problem for $\mathcal{EL}$ Actions. In *Proceedings of the 2008 International Workshop on Description Logics (DL2008)*, volume 353 of *CEUR-WS*. 2008. ↩ p. 10, 13

[LLMW06a] H. Liu, C. Lutz, M. Miličić, and F. Wolter. Description Logic Actions with General TBoxes: a Pragmatic Approach. In *Proceedings of the 2006 International Workshop on Description Logics (DL-2006)*. 2006. ↩ p. 10, 13

[LLMW06b] H. Liu, C. Lutz, M. Miličić, and F. Wolter. Reasoning about Actions using Description Logics with General TBoxes. In M. Fisher, W. van der Hoek, B. Konev, and A. Lisitsa, eds., *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA-2006)*, volume 4160 of *Lecture Notes in Artificial Intelligence*, pp. 266–279. Springer-Verlag, 2006. ↩ p. 10, 13

[LLMW06c] H. Liu, C. Lutz, M. Miličić, and F. Wolter. Updating Description Logic ABoxes. In P. Doherty, J. Mylopoulos, and C. Welty, eds., *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR-2006)*, pp. 46–56. AAAI Press, 2006. ↩ p. 6, 13, 80, 85

[LR97] F. Lin and R. Reiter. How to Progress a Database. *Artificial Intelligence*, 92(1-2):131–167, 1997. ↩ p. 5

[LRL+97] H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. B. Scherl. GOLOG: A Logic Programming Language for Dynamic Domains. *Journal of Logic Programming*, 31:59–84, 1997. ↩ p. 5, 31

[LS01] C. Lutz and U. Sattler. The Complexity of Reasoning with Boolean Modal Logics. In F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, eds., *Advances in Modal Logics Volume 3*. CSLI Publications, Stanford, CA, USA, 2001. ↩ p. 50

[LSW01] C. Lutz, U. Sattler, and F. Wolter. Modal Logics and the Two-Variable Fragment. In *Annual Conference of the European Association for Computer Science Logic (CSL-2001)*, LNCS. Springer Verlag, Paris, France, 2001. ↩ p. 50

[McC86] J. McCarthy. Applications of Circumscription to Formalizing Common Sense Knowledge. *Artificial Intelligence*, 28:89–116, 1986. ↩ p. 121

[Mil07] M. Miličić. Complexity of Planning in Action Formalisms Based on Description Logics. In *Proceedings of the 14th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR-2007)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2007. ↩ p. 10, 117

[Mil08] M. Miličić. *Action, Time and Space in Description Logics*. Ph.D. thesis, Technische Universität Dresden, 2008. ↩ p. 13, 20, 108, 115

[Min74] M. Minsky. A Framework for Representing Knowledge. (MIT) Artificial Intelligence Memo 306, Department of Computer Science, Massachusetts Institute of Technology, 1974. ↩ p. 1

[NORCR07] R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell, and A. Rubio. Challenges in Satisfiability Modulo Theories. In F. Baader, ed., *18th International Conference on Term Rewriting and Applications (TRA-2007)*, volume 4533 of *Lecture Notes in Computer Science*, pp. 2–18. Springer, 2007. ↩ p. 8, 86, 87, 88, 89, 90

[Pap94] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994. ↩ p. 8, 33, 45, 47, 48, 108

[PH05] I. Pratt-Hartmann. Complexity of the Two-Variable Fragment with Counting Quantifiers. *Journal of Logic, Language, and Information*, 14(3):369–395, 2005. ↩ p. 51, 108

[Pnu81] A. Pnueli. The Temporal Semantics of Concurrent Programs. *Theoretical Computer Science*, 13:45–60, 1981. ↩ p. 10

[PSS93] P. F. Patel-Schneider and B. Swartout. Description-Logic Knowledge Representation System Specification from the KRSS Group of the ARPA Knowledge Sharing Effort, 1993. ↩ p. 84

[PST00] L. Pacholski, W. Szwast, and L. Tendera. Complexity Results for First-Order Two-Variable Logic with Counting. *SIAM Journal on Computing*, 29(4):1083–1117, 2000. ↩ p. 51

[Qui67] M. R. Quillian. Word Concepts: A Theory and Simulation of Some Basic Semantic Capabilities. *Behavioral Science*, 12:410–430, 1967. ↩ p. 1

[Rei82] R. Reiter. Towards a Logical Reconstruction of Relational Database Theory. In *On Conceptual Modelling (Intervale)*, pp. 191–233. 1982. ↩ p. 4

[Rei01] R. Reiter. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press, Bradford Books, Cambridge, MA, 2001. ↩ p. 4, 31

[Sch91] K. Schild. A Correspondence Theory for Terminological Logics: Preliminary Report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-1991)*, pp. 466–471. Sidney, Australia, 1991. ↩ p. 3

[Sch03] S. Schlobach. Non-Standard Reasoning Services for the Debugging of Description Logic Terminologies. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, pp. 355–362. Morgan Kaufmann, 2003. ↩ p. 90

[Sow92] J. F. Sowa. Semantic Networks. In S. C. Shapiro, ed., *Encyclopedia of Artificial Intelligence*. Wiley, 1992. ↩ p. 1

[SPG+07] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A Practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53, 2007. ISSN 1570-8268. ↩ p. 3, 91

[SS89] M. Schmidt-Schaubß. Subsumption in KL-ONE is Undecidable. In *Proceedings of the first international conference on Principles of knowledge representation and reasoning (KR-1989)*, pp. 421–431. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989. ↩ p. 2

[SS91] M. Schmidt-Schauß and G. Smolka. Attributive Concept Descriptions with Complements. *Artificial Intelligence*, 48(1):1–26, 1991. ↩ p. 3

[ST07] R. A. Schmidt and D. Tishkovsky. Using Tableau to Decide Expressive Description Logics with Role Negation. In *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*, pp. 438–451. Springer, 2007. ↩ p. 51

[Sun09] B. Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies*. Ph.D. thesis, Technische Universität Dresden, 2009. ↩ p. 10

[Tar56] A. Tarski. *Logic, Semantics, Metamathematics: Papers from 1923 to 1938*. Oxford University Press, 1956. ↩ p. 16

[TB73] B. Trakhenbrot and Y. Barzdin. *Finite Automata: Behavior and Synthesis*. North–Holland, Amsterdam, 1973. ↩ p. 11, 100

[TH06] D. Tsarkov and I. Horrocks. FaCT++ Description Logic Reasoner: System Description. In *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR-2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pp. 292–297. Springer, 2006. ↩ p. 3, 95

[Thi97] M. Thielscher. Ramification and Causality. *Artificial Intelligence Journal*, 89(1–2):317–364, 1997. ↩ p. 4

[Thi05a] M. Thielscher. FLUX: A Logic Programming Method for Reasoning about Agents. *Theory and Practice of Logic Programming*, 5:533–565, 2005. ↩ p. 5, 32

[Thi05b] M. Thielscher. *Reasoning Robots: The Art and Science of Programming Robotic Agents*. Springer, 2005. ↩ p. 4, 32

[Tob00] S. Tobies. The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics. *Journal of Artificial Intelligence Research*, 12:199–217, 2000. ↩ p. 51, 108

[VW86] M. Y. Vardi and P. Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *Proceedings of the First Symposium on Logic in Computer Science*, pp. 322–331. Cambridge, 1986. ↩ p. 99, 111, 112

[VW94] M. Y. Vardi and P. Wolper. Reasoning about Infinite Computations. *Information and Computation*, 115:1–37, 1994. ↩ p. 10

[Web86] A. Weber. Updating Propositional Formulas. In *Proceedings of First. Conference on Expert Database Systems*, pp. 487–500. 1986. ↩ p. 4

[Win88] M. Winslett. Reasoning about Action Using a Possible Models Approach. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-1988)*, pp. 89–93. 1988. ↩ p. 4, 23

[Win90] M. Winslett. *Updating Logical Databases*. Cambridge University Press, 1990. ↩ p. 4, 23, 33, 49

[WVS83] P. Wolper, M. Y. Vardi, and A. P. Sistla. Reasoning about Infinite Computation Paths. In *Proceedings. 24th IEEE Symposium on Foundations of Computer Science*, pp. 185–194. Tucson, 1983. ↩ p. 99, 111, 112

[WZ00] F. Wolter and M. Zakharyaschev. Temporalizing Description Logics. In D. Gabbay and M. de Rijke, eds., *Frontiers of Combining Systems II (FroCos-2000)*, pp. 379–401. Kluwer Academic Publishers, 2000. ↩ p. 10

[ZMMM01] L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. Malik. Efficient Conflict Driven Learning in a Boolean Satisfiability Solver. In *International Conference on Computer-Aided Design (ICCAD-2001)*, pp. 279–285. 2001. ↩ p. 88