Updating Description Logic ABoxes

Hongkai Liu[1], **Carsten Lutz**[1], **Maja Miličić**[1], **Frank Wolter**[2]

[1]Institut für Theoretische Informatik
TU Dresden, Germany
lastname@tcs.inf.tu-dresden.de

[2]Department of Computer Science
University of Liverpool, UK
frank@csc.liv.ac.uk

Abstract

Description logic (DL) ABoxes are a tool for describing the state of affairs in an application domain. In this paper, we consider the problem of updating ABoxes when the state changes. We assume that changes are described at an atomic level, i.e., in terms of possibly negated ABox assertions that involve only atomic concepts and roles. We analyze such basic ABox updates in several standard DLs by investigating whether the updated ABox can be expressed in these DLs and, if so, whether it is computable and what is its size. It turns out that DLs have to include nominals and the "@" constructor of hybrid logic (or, equivalently, admit Boolean ABoxes) for updated ABoxes to be expressible. We devise algorithms to compute updated ABoxes in several expressive DLs and show that an exponential blowup in the size of the whole input (original ABox + update information) cannot be avoided unless every PTIME problem is LOGTIME-parallelizable. We also exhibit ways to avoid an exponential blowup in the size of the original ABox, which is usually large compared to the update information.

Introduction

Description logics (DLs) are a prominent family of logic-based formalisms for the representation of and reasoning about conceptual knowledge (Baader *et al.* 2003). In DLs, concepts are used to describe classes of individuals sharing common properties. For example, the following concept describes the class of all parents with only happy children:

$$\mathsf{Person} \sqcap \exists \mathsf{has\text{-}child}.\mathsf{Person} \sqcap \forall \mathsf{has\text{-}child}.(\mathsf{Person} \sqcap \mathsf{Happy})$$

This concept is formulated in $\mathcal{ALC}$, the basic DL that contains all Boolean operators. Concepts are the most important ingredient of description logic *ABoxes*, whose purpose is to describe a snapshot of the world. For example, the following ABox, which is also formulated in $\mathcal{ALC}$, says that John is a parent with only happy children, that Peter is his child, and that Mary is a person:

$$\begin{aligned} &\mathsf{john}{:}\mathsf{Person} \sqcap \exists \mathsf{has\text{-}child}.\mathsf{Person} \sqcap \\ &\qquad \forall \mathsf{has\text{-}child}.(\mathsf{Person} \sqcap \mathsf{Happy}) \\ &\mathsf{has\text{-}child}(\mathsf{john}, \mathsf{peter}) \\ &\mathsf{mary}{:}\mathsf{Person} \end{aligned}$$

Copyright © 2006, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

In many applications of DLs, an ABox is used to represent the current state of affairs in the application domain (Baader *et al.* 2003). In such applications, it is necessary to update the ABox in the case that the state has changed. Such an update should incorporate the information about the new state while retaining all knowledge that is not affected by the change (as demanded by the principle of inertia, see e.g. (M.P.Shanahan 1997)). For example, if Mary is now unhappy, we should update the above ABox to the following one. This updated ABox is formulated in $\mathcal{ALCO}$, the extension of $\mathcal{ALC}$ with nominals (i.e., individual names inside concept descriptions):

$$\begin{aligned} &\mathsf{john}{:}\mathsf{Person} \sqcap \exists \mathsf{has\text{-}child}.\mathsf{Person} \sqcap \\ &\qquad \forall \mathsf{has\text{-}child}.(\mathsf{Person} \sqcap (\mathsf{Happy} \sqcup \{\mathsf{mary}\})) \\ &\mathsf{has\text{-}child}(\mathsf{john}, \mathsf{peter}) \\ &\mathsf{mary}{:}\mathsf{Person} \sqcap \neg \mathsf{Happy} \end{aligned}$$

To see that this ABox is obtained by the update operation, note that ABoxes adopt the open world assumption and thus represent the domain in an incomplete way (Baader *et al.* 2003). In the example above, we have no information about whether or not Mary is a child of John. However, because we cannot exclude that this is the case, John may now have an unhappy child (which is Mary). Thus, the new information concerning Mary also resulted in an update of the information concerning John.

Surprisingly, formal theories of ABox updates have never been developed. In applications, ABoxes are usually updated in an ad-hoc way, and effects such as the information change for John above are simply ignored. The current paper aims at curing this deficiency. Its purpose is *to provide a first formal analysis of ABox updates in many common description logics, concentrating on the most basic kind of updates*. The basic kind of update we consider is as follows: the new information to be incorporated into the ABox is a set of possibly negated assertions $a{:}A$ and $r(a, b)$, where A is an *atomic* concept. As discussed in more detail later, there are both semantic and computational problems with unrestricted updates that are avoided by adopting these restrictions.

We consider ABox updates in the expressive DL $\mathcal{ALCQIO}$ and its fragments. It turns out that, in many natural DLs such as $\mathcal{ALC}$, the updated ABox cannot be expressed. As an example, consider the $\mathcal{ALC}$ ABox given

above. To express the ABox obtained by the update with mary:¬Happy, we had to resort to the more expressive DL $\mathcal{ALCO}$. But even the introduction of nominals does not suffice to guarantee that updated ABoxes are expressible. Only if we further add the "@" concept constructor from hybrid logic (Areces & de Rijke 2001) or Boolean ABoxes (we show that these two are equivalent in the presence of nominals), updated ABoxes can be expressed. Here, the @ constructor allows the formation of *concepts* of the form $@_a C$ expressing that the individual a satisfies C, and Boolean ABoxes are a generalization of standard ABoxes: while the latter can be thought of as a conjunction of ABox assertions of the form $a{:}C$ and $r(a, b)$, Boolean ABoxes are a Boolean combination of such ABox assertions. Our expressiveness results do not only concern $\mathcal{ALC}$: similar proofs as those given in this paper can be used to show that, in any standard DL in which nominals and the "@" constructor are not expressible, updated ABoxes cannot be expressed.

We show that updated ABoxes exist and are computable in $\mathcal{ALCQIO}^{@}$, the extension of $\mathcal{ALCQIO}$ (which includes nominals) with the @ constructor. The proposed algorithm can easily be adapted to the fragments $\mathcal{ALCO}^{@}$, $\mathcal{ALCIO}^{@}$, and $\mathcal{ALCQO}^{@}$. An important issue is the size of updated ABoxes: the updated ABoxes computed by our algorithm may be of size exponential both in the size of the original ABox and in the size of the new information (henceforth called the *update*). We show that an exponential blowup cannot be completely avoided by proving that, even in the case of propositional logic, the size of updated theories is not polynomial in the size of the (combined) input unless every PTIME-algorithm is LOGTIME-parallelizable (the "P vs. NP" question of parallel computation).[1] In the update literature, an exponential blowup in the size of the update is viewed as much more tolerable than an exponential blowup in the size of the original ABox since the former tend to be small compared to the latter. We believe that, in the case of $\mathcal{ALCQIO}^{@}$ and its two fragments mentioned above, the exponential blowup in the size of the original ABox cannot be avoided. While we leave a proof as an open problem, we exhibit two ways around the blowup: the first is to allow the introduction of new concept definitions in an acyclic TBox when computing the update. The second is to move to extensions of $\mathcal{ALCQIO}^{@}$ that allow Boolean operators on roles, thus eliminating the asymmetry between concepts and roles found in standard DLs. In both cases, we show how to compute updated ABoxes that are polynomial in the size of the original ABox (and exponential in the size of the update). Thus, the blowup induced by updates in these expressive DLs is not worse than in propositional logic. We also show that the blowup produced by iterated updates is not worse than the blowup produced by a single update.

[1] In contrast to the results by Cadoli et al. (Cadoli *et al.* 1999), 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 (Cadoli *et al.* 1999), 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.

Name	Syntax	Semantics
inverse role	r^-	$(r^{\mathcal{I}})^{-1}$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
at-least number restriction	$(\geqslant n\ r\ C)$	$\{x \mid \#\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$
at-most number restriction	$(\leqslant n\ r\ C)$	$\{x \mid \#\{y \mid (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$
@ constructor	$@_a C$	$\Delta^{\mathcal{I}}$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and $\emptyset$ otherwise

Figure 1: Syntax and semantics of $\mathcal{ALCQIO}$.

Full proofs of the results presented in this paper can be found in the accompanying technical report (Liu *et al.* 2005).

Description Logics

In DLs, *concepts* are inductively defined with the help of a set of *constructors*, starting with a set $\mathsf{N_C}$ of *concept names* and a set $\mathsf{N_R}$ of *role names*, and (possibly) a set $\mathsf{N_I}$ of *individual names*. In this section, we introduce the DL $\mathcal{ALCQIO}^{@}$, whose concepts are formed using the constructors shown in Figure 1. There, the inverse constructor is the only role constructor, whereas the remaining seven constructors are concept constructors. In Figure 1 and in what follows, we use $\#S$ to denote the cardinality of a set S, a and b to denote individual names, r and s to denote roles (i.e., role names and inverses thereof), A, B to denote concept names, and C, D to denote (possibly complex) concepts. As usual, we use $\top$ as abbreviation for an arbitrary (but fixed) propositional tautology, $\bot$ for $\neg\top$, $\rightarrow$ and $\leftrightarrow$ for the usual Boolean abbreviations, $\exists r.C$ (*existential restriction*) for $(\geqslant 1\ r\ C)$, and $\forall r.C$ (*universal restriction*) for $(\leqslant 0\ r\ \neg C)$.

The DL that allows only for negation, conjunction, disjunction, and universal and existential restrictions is called $\mathcal{ALC}$. The availability of additional constructors is indicated by concatenation of a corresponding letter: $\mathcal{Q}$ stands for number restrictions; $\mathcal{I}$ stands for inverse roles, $\mathcal{O}$ for nominals and superscript "@" for the @ constructor. This explains the name $\mathcal{ALCQIO}^{@}$ for our DL, and also allows us to refer to its sublanguages in a simple way.

The semantics of $\mathcal{ALCQIO}^{@}$-concepts is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The *domain* $\Delta^{\mathcal{I}}$ is a non-empty set of individuals and the *interpretation function* $\cdot^{\mathcal{I}}$ maps each concept name $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name $r \in \mathsf{N_R}$ to a binary relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$, and each individual name $a \in \mathsf{N_I}$ to an individual $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to inverse roles and arbitrary concepts is inductively defined as shown in the third column of Figure 1.

An $\mathcal{ALCQIO}^{@}$ *assertional box (ABox)* is a finite set of *concept assertions* $C(a)$ and *role assertions* $r(a, b)$ and $\neg r(a, b)$ (where r may be an inverse role). For readability, we sometimes write concept assertions as $a{:}C$. Observe that

there is no need for explicitly introducing negated concept assertions as negation is available as a concept constructor in $\mathcal{ALCQIO}^{@}$. An ABox $\mathcal{A}$ is *simple* if $C(a) \in \mathcal{A}$ implies that C is a *concept literal*, i.e., a concept name or a negated concept name.

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 ABox assertion α by an intepretation $\mathcal{I}$ with $\mathcal{I} \models \alpha$. An interpretation $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$ (written $\mathcal{I} \models \mathcal{A}$) if it satisfies all assertions in $\mathcal{A}$. An ABox is *consistent* iff it has a model. Two ABoxes $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent* (written $\mathcal{A} \equiv \mathcal{A}'$) iff they have the same models. We use $M(\mathcal{A})$ to denote the set of all models of the ABox $\mathcal{A}$.

ABox Updates

We introduce a simple form of ABox update where complex concepts are not allowed in the update information.

Definition 1 (Interpretation Update). An *update* $\mathcal{U}$ is a simple ABox that is consistent. 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 hold for all concept names A and role names r:

$$\begin{aligned} A^{\mathcal{I}'} &= (A^{\mathcal{I}} \cup \{a^{\mathcal{I}} \mid A(a) \in \mathcal{U}\}) \setminus \{a^{\mathcal{I}} \mid \neg A(a) \in \mathcal{U}\} \\ r^{\mathcal{I}'} &= (r^{\mathcal{I}} \cup \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid r(a,b) \in \mathcal{U}\}) \\ &\qquad \setminus \{(a^{\mathcal{I}}, b^{\mathcal{I}}) \mid \neg r(a,b) \in \mathcal{U}\} \end{aligned}$$

It is easily seen that, for each fixed 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 $\mathcal{I} \Longrightarrow_{\mathcal{U}} \mathcal{I}'$.

Based on the relation "$\Longrightarrow_{\mathcal{U}}$", we can now define ABox updates.

Definition 2 (ABox Update). Let $\mathcal{A}$ be an ABox and $\mathcal{U}$ an update. An ABox $\mathcal{A}'$ is the *result of updating $\mathcal{A}$ with $\mathcal{U}$* , in symbols $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$, if

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

We then call $\mathcal{A}$ the *original ABox* and $\mathcal{A}'$ the *updated ABox*.

Note that updated ABoxes are unique up to logical equivalence. This justifies talking of *the* result of updating $\mathcal{A}$ with $\mathcal{U}$.

There are at least two advantages of disallowing complex concepts in updates: first, the semantics given above is uncontroversial and coincides with all standard semantics for updates considered in the literature, see e.g., (Thielscher 2000b; Scherl & Levesque 2003; Winslett 1990; Reiter 2001). In contrast, several different and equally natural semantics become available as soon as we allow disjunction (or even non-Boolean constructors) in updates, see e.g. (Winslett 1990; Forbus 1989; Lin 1996; Thielscher 2000a; Zhang & Foo 2000; Herzig 1996; Reiter 2001). Second, it follows from the results in (Baader *et al.* 2005) that, at least under Winslett-style PMA semantics (Winslett 1990), unrestricted ABox updates in relatively simple DLs are not computable. It seems very likely that the other available semantics suffer from similar computational problems. Practically, our restriction means that the user has to describe updates at an atomic level.

We now give a more involved example of updating ABoxes than that given in the introduction. The following $\mathcal{ALCO}$ ABox expresses that John and Mary are married. We also know that (at least) one of them is unhappy, but we do not know which of the two this is. Moreover, Peter and Sarah both have a happy parent:

$$\begin{aligned} &\mathsf{spouse}(\mathsf{john}, \mathsf{mary}) \\ &\mathsf{peter}{:}\exists \mathsf{parent}.\mathsf{Happy} \\ &\mathsf{sarah}{:}\exists \mathsf{parent}.\mathsf{Happy} \\ &\mathsf{john}{:}\neg \mathsf{Happy} \sqcup \forall \mathsf{spouse}.(\{\mathsf{mary}\} \rightarrow \neg \mathsf{Happy}) \end{aligned}$$

Suppose that, because one of them is unhappy, John and Mary have an argument. This results in both John and Mary being unhappy now. Hence, we should apply the following update to the above ABox:

$$\neg \mathsf{Happy}(\mathsf{john}), \quad \neg \mathsf{Happy}(\mathsf{mary}).$$

Then, the updated ABox can be expressed in $\mathcal{ALCO}^{@}$ as follows. Here and in what follows, we assume that the @ constructor has higher precedence than conjunction:

$$\begin{aligned} &\mathsf{spouse}(\mathsf{john}, \mathsf{mary}) \\ &\mathsf{dummy}{:}\big(@_{\mathsf{peter}}\exists \mathsf{parent}.(\mathsf{Happy} \sqcup \{\mathsf{john}\}) \sqcap \\ &\qquad @_{\mathsf{sarah}}\exists \mathsf{parent}.(\mathsf{Happy} \sqcup \{\mathsf{john}\})\big) \sqcup \\ &\qquad \big(@_{\mathsf{peter}}\exists \mathsf{parent}.(\mathsf{Happy} \sqcup \{\mathsf{mary}\}) \sqcap \\ &\qquad @_{\mathsf{sarah}}\exists \mathsf{parent}.(\mathsf{Happy} \sqcup \{\mathsf{mary}\})\big) \\ &\neg \mathsf{Happy}(\mathsf{john}) \\ &\neg \mathsf{Happy}(\mathsf{mary}) \end{aligned}$$

The only surprising assertion in the updated ABox is the second one. Intuitively, it represents the update of the second and third assertion of the original ABox:[2] the first disjunct captures the case where John was unhappy and Mary was happy, and the second disjunct captures the case when it was the other way around. In the remaining that both Mary *and* John were unhappy, the update of the second and third assertions is

$$\mathsf{dummy}{:}@_{\mathsf{peter}}\exists \mathsf{parent}.\mathsf{Happy} \sqcap @_{\mathsf{sarah}}\exists \mathsf{parent}.\mathsf{Happy}$$

(because nothing has changed through the update). A corresponding disjunct in the second assertion of the updated ABox is not needed since it would imply each of the first two disjuncts.

Also note that the last line of the original ABox is subsumed by the last two lines of the updated ABox.

We shall later refer back to this example as the "spouse example" and prove that the updated ABox cannot be expressed in $\mathcal{ALCO}$.

[2] Note that the individual name dummy in front of the colon does not carry any information: we could exchange it with any other individual name without changing the meaning of this assertion.

Description Logics without Updates

We say that a description logic $\mathcal{L}$ *has ABox updates* iff, for every ABox $\mathcal{A}$ formulated in $\mathcal{L}$ and every update $\mathcal{U}$, there exists an ABox $\mathcal{A}'$ formulated in $\mathcal{L}$ such that $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$. In this section, we show that a lot of basic DLs are lacking ABox updates.

Updates in $\mathcal{ALC}$

We analyze the basic description logic $\mathcal{ALC}$ and show that it does not have ABox updates. Consider the following $\mathcal{ALC}$ ABox $\mathcal{A}$, update $\mathcal{U}$, and $\mathcal{ALCO}$ ABox $\mathcal{A}'$:

$$\begin{array}{rcl} \mathcal{A} & := & \{\forall r.A(a)\} \\ \mathcal{U} & := & \{\neg A(b)\} \\ \mathcal{A}' & := & \{\neg A(b), \forall r.(A \sqcup \{b\})(a)\}. \end{array}$$

The following is not difficult to verify using Definition 2.

Lemma 3. $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$.

To show that $\mathcal{ALC}$ does not have ABox updates, it thus suffices to prove that there is no $\mathcal{ALC}$ ABox equivalent to the $\mathcal{ALCO}$ ABox $\mathcal{A}'$. This is an easy exercise: find two models $\mathcal{I}$ and $\mathcal{I}'$ that are bisimilar[3] such that $\mathcal{I} \models \mathcal{A}'$ and $\mathcal{I}' \not\models \mathcal{A}'$. Then use the fact that $\mathcal{ALC}$ concepts cannot distinguish between bisimilar models to show the desired result. Details of this and following proofs can be found in (Liu *et al.* 2005).

Lemma 4. *There is no $\mathcal{ALC}$ ABox equivalent to $\mathcal{A}'$.*

Note that our proof applies to the case where the update contains only concept assertions, but no role assertions.

Theorem 5. *$\mathcal{ALC}$ does not have ABox updates, even if updates contain only concept assertions.*

The fact that the updated ABox $\mathcal{A}'$ used in this section is actually an $\mathcal{ALCO}$ ABox may give rise to the conjecture that adding nominals to $\mathcal{ALC}$ recovers the existence of updates. Unfortunately, as shown in the following section, this is not the case.

Updates in $\mathcal{ALCO}$

To show that $\mathcal{ALCO}$ does not have ABox updates, we proceed in two steps: we first give a straightforward proof of the non-existence of updated ABoxes in $\mathcal{ALCO}$ that relies on the use of role assertions in updates. In the second step, we use a slightly more complex construction to show that $\mathcal{ALCO}$ does not have ABox updates even if only concept assertions are allowed in updates.

Consider the following $\mathcal{ALC}$ ABox $\mathcal{A}$ (which thus also is an $\mathcal{ALCO}$ ABox), update $\mathcal{U}$, and $\mathcal{ALCO}^{@}$ ABox $\mathcal{A}'$:

$$\begin{array}{rcl} \mathcal{A} & := & \{\exists r.A(a)\} \\ \mathcal{U} & := & \{\neg r(a,b)\} \\ \mathcal{A}' & = & \{(\exists r.(A \sqcap \neg\{b\}) \sqcup @_b A)(a), \neg r(a,b)\}. \end{array}$$

Again, the following is not difficult to verify:

Lemma 6. $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$.

[3]W.r.t. the standard notion of bisimilarity for $\mathcal{ALC}$ that is independent of individual names (Kurtonina & de Rijke 1999).

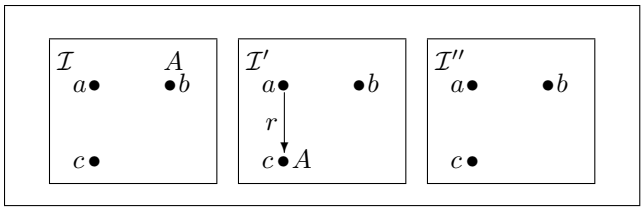

Figure 2: Interpretations for Lemma 7

We now show that there exists no $\mathcal{ALCO}$ ABox that is equivalent to the $\mathcal{ALCO}^{@}$ ABox $\mathcal{A}'$. It follows that $\mathcal{ALCO}$ does not have ABox updates.

Consider the interpretations $\mathcal{I}$, $\mathcal{I}'$ and $\mathcal{I}''$ depicted in Figure 2. We assume that the individual names a, b, and c are mapped to the individuals of the same name, and that all other individual names are mapped to the individual c. Moreover, the concept name A is interpreted as shown in the figure and all other concept names are interpreted as the empty set in all three interpretations. Then we have $\mathcal{I} \models \mathcal{A}'$, $\mathcal{I}' \models \mathcal{A}'$ and $\mathcal{I}'' \not\models \mathcal{A}'$. We will show that, if an $\mathcal{ALCO}$ ABox $\mathcal{B}$ is equivalent to $\mathcal{A}'$, then $\mathcal{I}'' \models \mathcal{B}$, which is a contradiction.

Lemma 7. *There is no $\mathcal{ALCO}$ ABox that is equivalent to the $\mathcal{ALCO}^{@}$ ABox $\mathcal{A}' = \{(\exists r.(A \sqcap \neg\{b\}) \sqcup @_b A)(a), \neg r(a,b)\}$.*

Proof. Assume there is an $\mathcal{ALCO}$ ABox $\mathcal{B}$ that is equivalent to $\mathcal{A}'$. Then $\mathcal{I} \models \mathcal{B}$, $\mathcal{I}' \models \mathcal{B}$, and $\mathcal{I}'' \not\models \mathcal{B}$. We show that, for all assertions $\varphi \in \mathcal{B}$, we have $\mathcal{I}'' \models \varphi$, thus obtaining a contradiction to $\mathcal{I}'' \not\models \mathcal{B}$. First, $\mathcal{B}$ does not contain any positive role assertion since $\mathcal{I} \models \mathcal{B}$ and $\mathcal{I}$ does not satisfy any positive role assertions. Second, if φ is a negative role assertion, then $\mathcal{I}'' \models \varphi$ since $\mathcal{I}''$ satisfies all negative role assertions. Finally, let φ be a concept assertion. Then, $\mathcal{I}'' \models \varphi$ is a consequence of $\mathcal{I} \models \varphi$, $\mathcal{I}' \models \varphi$, and the fact that the truth of an assertion $C(\hat{a})$ in a model $\mathcal{J}$, C an $\mathcal{ALCO}$-concept, only depends on the set of points reachable from $\hat{a}^{\mathcal{J}}$ by role paths. ❏

Note that our proof also shows that $\mathcal{ALC}$ does not have ABox updates even if we restrict ourselves to updates containing only role assertions.

Theorem 8. *$\mathcal{ALC}$ and $\mathcal{ALCO}$ do not have ABox updates, even if updates contain only role assertions.*

This result raises the question whether or not restricting updates to concept assertions regains the existence of updated ABoxes in $\mathcal{ALCO}$. We answer this question to the negative by returning to the spouse example. Let $\mathcal{A}$, $\mathcal{U}$, and $\mathcal{A}'$ denote the original ABox (formulated in $\mathcal{ALCO}$), update, and updated ABox (formulated in $\mathcal{ALCO}^{@}$) from this example. We have already argued that $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$. It suffices to prove that there is no $\mathcal{ALCO}$ ABox equivalent to $\mathcal{A}'$.

Consider the interpretations $\mathcal{I}$, $\mathcal{I}'$ and $\mathcal{I}''$ depicted in Figure 3 where we abbreviate the individual names from the spouse example using their initial letter, x denotes an individual, all horizontal edges are for the role spouse, and all vertical edges are for the role parent. We assume that the four individual names j, m, p, s are mapped to individuals

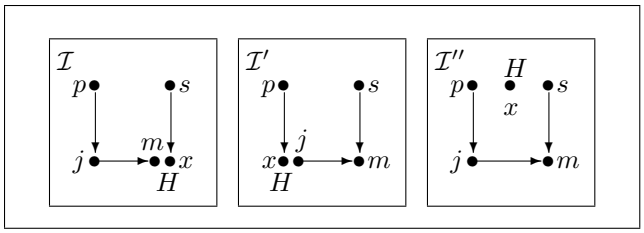

Figure 3: Interpretations for Lemma 9

of the same name, and that all other individual names are mapped to the individual x. Moreover, H is interpreted as indicated and all other concept names are interpreted as the empty set.

The proof of the following lemma uses the facts that $\mathcal{I} \models \mathcal{A}'$, $\mathcal{I}' \models \mathcal{A}'$, but $\mathcal{I}'' \not\models \mathcal{A}'$. It is quite similar to the proof of Lemma 7.

Lemma 9. *There is no $\mathcal{ALCO}$-ABox that is equivalent to the updated ABox from the spouse example.*

Thus, we obtain the following result:

Theorem 10. *$\mathcal{ALCO}$ does not have ABox updates, even if updates contain only concept assertions.*

Updates in $\mathcal{ALC}^@$ and Boolean ABoxes in $\mathcal{ALC}$

The proofs of Theorems 8 and 10 suggest that there is a connection between ABox updates and the "@" constructor. Indeed, we will later see that the DL $\mathcal{ALCO}^@$ has ABox updates. Here, we show that adding *only* the @ constructor to $\mathcal{ALC}$ does not suffice to guarantee the existence of updated ABoxes. Indeed, we even consider Boolean ABoxes (Areces *et al.* 2003), which are closely related to the @ constructor but strictly more expressive.

Boolean ABox assertions are Boolean combinations of ABox assertions expressed in terms of the connectives $\wedge$ and $\vee$. Then, a *Boolean ABox* is simply a finite set of Boolean ABox assertions. 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 paper. For example, the following is a Boolean ABox:

$$\{B(a), (A(a) \wedge r(a,b)) \vee \neg\exists s.A(b)\}.$$

An interpretation $\mathcal{I}$ is a model of a Boolean ABox $\mathcal{A}$ if every Boolean ABox assertion in $\mathcal{A}$ evaluates to true. The following lemma relates Boolean ABoxes and the @ constructor. It shows that non-Boolean $\mathcal{ALCO}^@$ ABoxes have exactly the same expressive power as Boolean $\mathcal{ALCO}$-ABoxes, and that the same does not hold for $\mathcal{ALC}$: while every $\mathcal{ALC}^@$ ABox can be translated into an equivalent Boolean $\mathcal{ALC}$ ABox, there are Boolean $\mathcal{ALC}$ ABoxes for which no equivalent non-Boolean $\mathcal{ALC}^@$ ABox exists.

Lemma 11.
(i) For every Boolean $\mathcal{ALC}^@$ ABox ($\mathcal{ALCO}^@$ ABox), there exists an equivalent Boolean $\mathcal{ALC}$ ABox ($\mathcal{ALCO}$ ABox);
(ii) For every Boolean $\mathcal{ALCO}$ ABox, there exists an equivalent non-Boolean $\mathcal{ALCO}^@$ ABox;
(iii) There exists no non-Boolean $\mathcal{ALC}^@$ ABox that is equivalent to the Boolean $\mathcal{ALC}$ ABox $\{A(a) \vee r(b,c)\}$.

Since, when added to $\mathcal{ALC}$, Boolean ABoxes are more expressive than the @ constructor, it is more general to consider the former when proving the lack of ABox updates.

Theorem 12. *There exists an $\mathcal{ALC}$ ABox $\mathcal{A}$ and an update $\mathcal{U}$ such that there exists no Boolean $\mathcal{ALC}$ ABox $\mathcal{A}'$ with $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$.*

The proof of Theorem 12 uses the ABoxes $\mathcal{A}$, $\mathcal{A}'$ and the update $\mathcal{U}$ that have been used in the proof that $\mathcal{ALC}$ does not have ABox updates. To show that no Boolean ABox $\mathcal{B}$ is equivalent to $\mathcal{A}$, $\mathcal{B}$ is first converted into disjunctive normal form and then proceeds similar to the non-Boolean case. By Lemma 11, we obtain the following corollary.

Corollary 13. *$\mathcal{ALC}^@$ does not have ABox updates.*

Observe that both Theorem 12 and Corollary 13 remain true if we restrict updates to only concept assertions.

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

Straightforward extensions of the results obtained in the previous section show that none of the standard DLs between $\mathcal{ALC}$ and $\mathcal{ALCQIO}$ has ABox updates. In this section, we show that adding nominals and the @ constructor to such DLs suffices to have ABox updates. More presicely, we prove that the expressive DL $\mathcal{ALCQIO}^@$ has ABox updates. The proof is easily adapted to the fragments of $\mathcal{ALCQIO}^@$ obtained by dropping number restrictions, inverse roles, or both.

Our construction of updated ABoxes is an extension of the corresponding construction for propositional logic described in (Winslett 1990), and 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 update $\mathcal{U}$ into a concept $C^\mathcal{U}$ such that the following holds:

($*$) for all interpretations $\mathcal{I}$ and $\mathcal{I}'$ such that $\mathcal{I}$ satisfies *no* assertion in $\mathcal{U}$ and $\mathcal{I} \Longrightarrow_\mathcal{U} \mathcal{I}'$, we have $C^\mathcal{I} = (C^\mathcal{U})^{\mathcal{I}'}$.

So 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. Our aim is to use the translation $C^\mathcal{U}$ to update concept assertions in ABoxes. We will later see how to overcome the restriction that $\mathcal{I}$ has to satisfy no assertion in $\mathcal{U}$.

For defining the concepts $C^\mathcal{U}$, we first introduce a bit of notation. For an ABox $\mathcal{A}$, we use $\mathsf{Obj}(\mathcal{A})$ to denote the set of individual names in $\mathcal{A}$, and $\mathsf{sub}(\mathcal{A})$ to denote the set of subconcepts of the concepts occurring in $\mathcal{A}$. For an ABox $\mathcal{A}$, we use $\neg\mathcal{A}$ to denote $\{\neg\varphi \mid \varphi \in \mathcal{A}\}$. The inductive translation that takes a concept C and an update $\mathcal{U}$ to a concept $C^\mathcal{U}$ as explained above is given in Figure 4.

Lemma 14. *The translation of concepts C into concepts $C^\mathcal{U}$ given in Figure 4 satisfies ($*$).*

$$
\begin{array}{rcl}
A^{\mathcal{U}} & = & A \sqcup \bigsqcup_{\neg A(a) \in \mathcal{U}} \{a\} \sqcap \neg(\bigsqcup_{A(a) \in \mathcal{U}} \{a\}) \\
(@_a C)^{\mathcal{U}} & = & @_a C^{\mathcal{U}} \\
(C \sqcap D)^{\mathcal{U}} & = & C^{\mathcal{U}} \sqcap D^{\mathcal{U}} \\
(\geqslant m\ r\ C)^{\mathcal{U}} & = & (\bigsqcap_{a \in \mathsf{Obj}(\mathcal{U})} \neg\{a\} \sqcap (\geq m\ r\ C^{\mathcal{U}})) \\
& & \sqcup \bigsqcup_{a \in \mathsf{Obj}(\mathcal{U})} \Big(\{a\} \sqcap \bigsqcup_{m_1+m_2+m_3=m} \big((\geq m_1\ r \bigsqcap_{b \in \mathsf{Obj}(\mathcal{U})} \neg\{b\} \sqcap C^{\mathcal{U}}) \sqcap (\geq m_2\ r \bigsqcup_{b \in \mathsf{Obj}(\mathcal{U}), r(a,b) \notin \mathcal{U}} \{b\} \sqcap C^{\mathcal{U}}) \\
& & \qquad \sqcap \bigsqcup_{S \subseteq \{b \mid \neg r(a,b) \in \mathcal{U}\}, |S| = m_3} \bigsqcap_{b \in S} @_b C^{\mathcal{U}} \big) \Big) \\
(\leqslant m\ r\ C)^{\mathcal{U}} & = & (\bigsqcap_{a \in \mathsf{Obj}(\mathcal{U})} \neg\{a\} \sqcap (\leq m\ r\ C^{\mathcal{U}})) \\
& & \sqcup \bigsqcup_{a \in \mathsf{Obj}(\mathcal{U})} \Big(\{a\} \sqcap \bigsqcup_{m_1+m_2+m_3=m} \big((\leq m_1\ r \bigsqcap_{b \in \mathsf{Obj}(\mathcal{U})} \neg\{b\} \sqcap C^{\mathcal{U}}) \sqcap (\leq m_2\ r \bigsqcup_{b \in \mathsf{Obj}(\mathcal{U}), r(a,b) \notin \mathcal{U}} \{b\} \sqcap C^{\mathcal{U}}) \\
& & \qquad \sqcap \bigsqcap_{S \subseteq \{b \mid \neg r(a,b) \in \mathcal{U}\}, |S| = m_3+1} \bigsqcup_{b \in S} \neg @_b C^{\mathcal{U}} \big) \Big)
\end{array}
\qquad
\begin{array}{rcl}
\{a\}^{\mathcal{U}} & = & \{a\} \\
(\neg C)^{\mathcal{U}} & = & \neg C^{\mathcal{U}} \\
(C \sqcup D)^{\mathcal{U}} & = & C^{\mathcal{U}} \sqcup D^{\mathcal{U}}
\end{array}
$$

Figure 4: Constructing $C^{\mathcal{U}}$

We now extend the update of concepts to the update of ABoxes. Let $\mathcal{A}$ be an ABox and $\mathcal{U}$ an update. Then define the ABox $\mathcal{A}^{\mathcal{U}}$ by setting

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

We can now establish a property that corresponds to $(*)$, but concerns ABoxes instead of concepts.

Lemma 15. *Let $\mathcal{A}$ be an ABox and $\mathcal{U}$ an update. For every interpretation $\mathcal{I}$ with $\mathcal{I} \models \neg\mathcal{U}$, we have $\mathcal{I} \models \mathcal{A}$ iff $\mathcal{I}^{\mathcal{U}} \models \mathcal{A}^{\mathcal{U}}$.*

Similar to the concepts $C^{\mathcal{U}}$, the ABox update $\mathcal{A}^{\mathcal{U}}$ works only if the interpretations $\mathcal{I}$ of $\mathcal{A}$ satisfy no assertion in $\mathcal{U}$. For a fixed interpretation $\mathcal{I}$, we can overcome this problem 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, in general we are confronted with the problem that an ABox can have many different models, and these models can violate different assertions 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 $\mathcal{U}' \subseteq \mathcal{U}$ of violated assertions separately, and then simply take the disjunction.

Let $\mathcal{A}$ be an ABox and $\mathcal{U}$ an update. A simple ABox $\mathcal{D}$ is called a *diagram for $\mathcal{U}$* if it is a maximal consistent subset of $L_{\mathcal{U}}$, where $L_{\mathcal{U}} = \{\psi, \neg\psi \mid \psi \in \mathcal{U}\}$ is the set of *literals* over $\mathcal{U}$. Intuitively, a diagram gives a complete description of the part of a model of $\mathcal{A}$ that is "relevant" for the update $\mathcal{U}$. Let $\mathfrak{D}$ be the set of all diagrams for $\mathcal{U}$ and consider for $\mathcal{D} \in \mathfrak{D}$ the set $\mathcal{D}_{\mathcal{U}} := \{\psi \mid \neg\psi \in \mathcal{D} \text{ and } \psi \in \mathcal{U}\}$ which corresponds to taking a subset of $\mathcal{U}$ as described above: we retain only those parts of $\mathcal{U}$ that are violated by interpretations whose relevant part is described by $\mathcal{D}$. We now define the updated ABox $\mathcal{A}'$ as

$$
\mathcal{A}' \ = \ \bigvee_{\mathcal{D} \in \mathfrak{D}} \bigwedge \mathcal{A}^{\mathcal{D}_{\mathcal{U}}} \cup \mathcal{D}_{\mathcal{U}} \cup (\mathcal{D} \setminus \neg\mathcal{D}_{\mathcal{U}}).
$$

Here, we use Boolean ABox operators only as an abbreviation for the "@" constructor. The expansion of this abbreviation does not substantially increase the size of the updated ABox: the translation from Boolean ABoxes to non-Boolean ones described in (Liu *et al.* 2005) is linear. To achieve a less redundant ABox, it is possible to drop from $\mathcal{A}'$ those disjuncts for which the diagram $\mathcal{D}$ is not consistent w.r.t. $\mathcal{A}$. This is, however, not strictly necessary since the ABox $\mathcal{D} \setminus \neg\mathcal{D}_{\mathcal{U}}$ ensures that these disjuncts are inconsistent.

Lemma 16. $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$.

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 universal restrictions in the $C^{\mathcal{U}}$ translation rather than number restrictions. The corresponding clauses are shown in Figure 5. The lemmas proved above for $\mathcal{ALCQIO}^{@}$ are then easily adapted.

Theorem 17. *All of the following DLs have ABox updates: $\mathcal{ALCO}^{@}$, $\mathcal{ALCIO}^{@}$, $\mathcal{ALCQO}^{@}$, and $\mathcal{ALCQIO}^{@}$.*

Now that we know that updated ABoxes always exist in the above DLs, we should have a look at their size. Let us first make precise what we mean with this. The *length* of a concept C, denoted by $|C|$, is the number of symbols needed to write C. Note that it makes a considerable difference whether we assume the numbers inside number restrictions to be written in unary or in binary: if written in unary, we have $|(\leqslant n\ r\ C)| \in \mathcal{O}(n)$, and if written in binary, we have

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

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

$|(\leqslant\ n\ r\ C)| \in \mathcal{O}(\log n)$. In the following, we will always point out to which coding our results apply.[4] The *size* of an ABox assertion $C(a)$ is $|C|$, the size of $r(a,b)$ and $\neg r(a,b)$ is 1. Finally, the *size* of an ABox $\mathcal{A}$, denoted by $|\mathcal{A}|$, is the sum of the sizes of all assertions in $\mathcal{A}$.

A close inspection of our construction reveals the following: first, the size the concepts $C^{\mathcal{D}_\mathcal{U}}$ is exponential in the size of $\mathcal{A}$ and polynomial in the size of $\mathcal{U}$; and second, the number of disjuncts in $\mathcal{A}'$ is exponential in the size of $\mathcal{U}$. These bounds hold regardless of the coding of numbers.

Theorem 18.
Let $\mathcal{L} \in \{\mathcal{ALCO}^@, \mathcal{ALCIO}^@, \mathcal{ALCQO}^@, \mathcal{ALCQIO}^@\}$. Then there are polynomials p_1, p_2, and q such that, for every $\mathcal{L}$ ABox $\mathcal{A}$ and every update $\mathcal{U}$, there is an $\mathcal{L}$ ABox $\mathcal{A}'$ such that

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

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. Then, it is clearly important that the exponential blowups of the individual updates do not add up. The following theorem, which can be proved by carefully investigating our update construction, shows that this is indeed not the case. It holds independently of the coding of numbers.

Theorem 19. *There are polynomials p_1, p_2 such that the following holds: for all ABoxes $\mathcal{A}_0, \ldots, \mathcal{A}_n$ and 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*

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

Conditional Updates

For the sake of simplicity, we have defined ABox updates to be unconditional: the assertions in the update $\mathcal{U}$ are unconditionally true after the update and we cannot express statements such as "$A(a)$ is true after the update if $C(b)$ was true before". In some applications such as reasoning about actions with DLs (Baader *et al.* 2005), it is more useful to have *conditional updates*, where the initial interpretation determines the changes that are triggered.

A *conditional update* $\mathcal{U}$ is a finite set of expressions φ/ψ, where the *precondition* φ is an ABox assertion (possibly involving non-atomic concepts) and the *postcondition* ψ is an assertion of the form

$$A(a), \neg A(a), r(a,b), \neg r(a,b)$$

with A a concept name. Intuitively, an expression φ/ψ means that if φ holds in the initial interpretation, then ψ holds after the update. As in the case of unconditional updates, we require a consistency condition: if φ/ψ and $\varphi'/\neg\psi$ are both in $\mathcal{U}$, then the ABox $\{\varphi, \varphi'\}$ has to be inconsistent.

The definition of "$\Longrightarrow_\mathcal{U}$" is easily adapted to the case of conditional updates: for $\mathcal{U}$ a conditional update, we write $\mathcal{I} \Longrightarrow_\mathcal{U} \mathcal{I}'$ if the following hold:

- for all concept names A,
$$A^{\mathcal{I}'} = (A^\mathcal{I} \cup \{a^\mathcal{I} \mid \varphi/A(a) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}) \setminus \{a^\mathcal{I} \mid \varphi/\neg A(a) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}$$
- for all role names r,
$$r^{\mathcal{I}'} = (r^\mathcal{I} \cup \{(a^\mathcal{I}, b^\mathcal{I}) \mid \varphi/r(a,b) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}) \setminus \{(a^\mathcal{I}, b^\mathcal{I}) \mid \varphi/\neg r(a,b) \in \mathcal{U} \wedge \mathcal{I} \models \varphi\}.$$

Then the result of updating an ABox is defined exactly as in the case of unconditional updates. Clearly, conditional updates generalize unconditional once since assertions ψ of unconditional updates can be expressed as $\top(a)/\psi$, with a an arbitrary individual name.

We now show how to adapt our construction of updated ABoxes to conditional updates. For $\mathcal{U}$ a conditional update, we use $\mathsf{rhs}(\mathcal{U})$ to denote $\{\psi \mid \varphi/\psi \in \mathcal{U}\}$, and $\mathsf{lhs}(\mathcal{U})$ for $\{\varphi \mid \varphi/\psi \in \mathcal{U}\}$. In the original algorithm, the updated ABox $\mathcal{A}'$ is assembled by taking one disjunct for every diagram for $\mathcal{U}$. The intuition is that, when a diagram $\mathcal{D}$ is satisfied by an interpretation $\mathcal{I}$, then we know which assertions in $\mathcal{U}$ have already been satisfied in $\mathcal{I}$ before $\mathcal{U}$ is applied. We generalize this idea to conditional updates by taking one disjunct for each pair $(\mathcal{D}, \mathcal{U}')$, where $\mathcal{D}$ is a diagram for $\mathsf{rhs}(\mathcal{U})$, and $\mathcal{U}'$ is a subset of $\mathcal{U}$. Intuitively, $\mathcal{U}'$ determines the set of assertions from $\mathcal{U}$ whose preconditions are satisfied in the initial model, and $\mathcal{D}$ determines the post-conditions that actually cause a change.

[4] In fact, all results except Theorems 26 and 27 apply to both unary and binary coding.

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

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

Then we can assemble the updated ABox $\mathcal{A}'$ as follows:

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

The notion of a description logic $\mathcal{L}$ *having conditional ABox updates* is defined in the obvious way.

Theorem 20. *All of the following DLs have conditional ABox updates: $\mathcal{ALCO}^{@}$, $\mathcal{ALCIO}^{@}$, $\mathcal{ALCQO}^{@}$, and $\mathcal{ALCQIO}^{@}$.*

Concerning the size and computability of updated ABoxes, we obtain the same bounds as in Theorem 18, independently of the coding of numbers.

A Lower Bound for the Size of Updated ABoxes

In the following sections, we are interested in the question whether or not the exponential blowup observed in Theorems 18 and 19 can be avoided. In this section, we consider updates of *propositional logic theories* where the updates are of the restricted form considered in this paper, i.e., conjunctions of literals. We prove that, even in this case, an exponential blowup in the size of the whole input (original ABox + update) cannot be avoided unless the complexity classes PTIME and NC coincide. As discussed in (Papadimitriou 1994), this is believed to be similarly unlikely as PTIME $=$ NP. It is not difficult to prove that this lower bound on the size of updated ABoxes transfers to all DLs considered in this paper.

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 $\sqcup$. A *propositional update* $\mathcal{U}$ contains only assertions of the form $A(a)$ and $\neg A(a)$. 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 individual x. Since we are dealing with propositional ABoxes and updates, we assume that interpretations do not intepret role names, and that interpretation domains have only a single element x with $a^{\mathcal{I}} = x$. We introduce a couple of notions. For a concept C, let $\mathsf{C}(C)$ denote the set of concept names used in C. For an interpretation $\mathcal{I}$ and a set of concept names Γ, let $\mathcal{I}|_\Gamma$ denote the restriction of $\mathcal{I}$ that interpretes only the concept names in Γ. Let C be a concept and $\Gamma \subseteq \mathsf{C}(C)$. Then a concept D is called a *uniform Γ-interpolant* of C iff $\mathsf{C}(D) \subseteq \Gamma$ and $\{\mathcal{I}|_\Gamma \mid x \in C^{\mathcal{I}}\} = \{\mathcal{I}|_\Gamma \mid x \in D^{\mathcal{I}}\}$. It is easily seen that, for any propositional concept C and subset $\Gamma \subseteq \mathsf{C}(C)$, the uniform Γ-interpolant of C exists and is unique up to equivalence. The following lemma establishes a tight connection between uniform interpolants and propositional updates.

Lemma 21. *Let $\mathcal{A} = \{C(a)\}$ be a propositional ABox, $\mathcal{U}$ a propositional update, Γ the set of concept names in C not occurring in $\mathcal{U}$, $\widehat{C}$ the shortest uniform Γ-interpolant of C, and*

$$\mathcal{A}' = \{a : (\widehat{C} \sqcap \prod_{A(a)\in\mathcal{U}} A)\}.$$

Then we have the following:

(i) $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}'$;

(ii) if $\mathcal{A} * \mathcal{U} \equiv \mathcal{A}''$, then $|\mathcal{A}'| \leq |\mathcal{U}| + |\mathcal{A}''|$.

It thus remains to show that the size of (smallest) uniform interpolants of propositional concepts is not bounded polynomially in the size of the interpolated concept unless PTIME $=$ NC.

The size of uniform interpolants of propositional concepts is closely related to the relative succinctness of propositional logic (PL) formulas and Boolean circuits. We remind that both PL formulas and Boolean circuits compute Boolean functions and refer, e.g., to (Papadimitriou 1994) for exact definitions. We use $|c|$ to denote the number of gates in the Boolean circuit c, and $|\varphi|$ to denote the length of the PL formula φ. It is known that, unless PTIME $=$ NC, there exists no polynomial p such that every Boolean circuit c can be converted into a PL formula φ that computes the same function as c_i and satisfies $|\varphi| \leq p(|c_i|)$, see e.g. Exercise 15.5.4 of (Papadimitriou 1994). In the following, we show that non-existence of such a polynomial p implies that uniform interpolants are not bounded polynomially in the size of the interpolated concept. 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. Let $\mathcal{G}$ be the set of concept names introduced for gate outputs, and let $O \in \mathcal{G}$ be the concept name for the output of the gate computing the final output of c. It is not difficult to see that this translation can be done such that there exists a polynomial q such that, for all Boolean circuits c,

(i) $|D_c| \leq q(|c|)$ and

(ii) for all interpretations $\mathcal{I}$ and all $x \in D_c^{\mathcal{I}}$, $x \in O^{\mathcal{I}}$ iff c outputs "true" on input $b_1, \ldots, b_k$, where $b_j = 1$ if $x \in I_j^{\mathcal{I}}$ and $b_j = 0$ otherwise.

Now, set $\Gamma := \mathcal{G} \setminus \{O\}$. Then the uniform Γ-interpolant $\widehat{D}_c$ of D_c also satisfies (ii). Thus, $\widehat{D}_c$ is a (notational variant of a) propositional logic formula computing the same Boolean function as c. If the size of $\widehat{D}_c$ would be bounded polynomially in the size of D_c, we thus had obtained a contradiction to our assumption on the non-existence of the polynomial p. Together with Lemma 21, we obtain the following theorem.

Theorem 22. *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*
- *$|\mathcal{A}'| \leq p(|\mathcal{A}| + |\mathcal{U}|)$.*

Our result is closely related to a result of (Cadoli *et al.* 1999) who prove an analogue of Theorem 22 under the complexity-theoretic assumption that the polynomial hierarchy does not collapse. However, Cadoli et al.'s technique does not appear to work with the restricted form of updates (conjunctions of literals) considered in this paper.

Small(er) Updated ABoxes

Theorem 18 does not differentiate between exponential blowups in the size of the original ABox and exponential blowups in the size of the update. In contrast to the former, the latter is usually considered acceptable since updates will usually be small compared to the original ABox $\mathcal{A}$. We believe that, in the DLs mentioned in Theorem 17, the exponential blowup in the size of $\mathcal{A}$ is unavoidable. However, we have to leave a proof as an open problem. In the following, we exhibit three different ways to extend $\mathcal{ALCQIO}^{@}$ and its fragments such that it becomes possible to compute updated ABoxes that are only polynomial in the size of the original ABox.

A first, rather restrictive solution is to admit only concept assertions in updates. Then, in all DLs captured by Theorem 17, computing the concepts $C^{\mathcal{U}}$ becomes a lot simpler: just replace every concept name A in C with

$$A \sqcup \bigsqcup_{\neg A(a) \in \mathcal{B}} \{a\} \sqcap \neg (\bigsqcup_{A(a) \in \mathcal{B}} \{a\}).$$

If modified in this way, our original construction 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 and also applies to iterated updates.

Small Updates Through TBoxes

We show how to produce smaller updated ABoxes by allowing the introduction of auxiliary concept names via an acyclic TBox. In the propositional case, this corresponds to admitting additional variables for defining abbreviations. In the terminology of Cadoli et al. (Cadoli *et al.* 1999), we thus move from logical equivalence to query equivalence. In this way, we obtain updates that are polynomial in the size of the original ABox.

In the following, we assume that the set of concept names is partitioned into a set of *primary* concept names and a set of *auxiliary* concept names. The latter are used only for defining abbreviations in a TBox. A *concept definition* is of the form $A \equiv C$, where A is an auxiliary concept name and C is a concept. An *(acyclic) TBox* $\mathcal{T}$ is a finite set of concept definitions with unique left-hand sides and without cyclic definitions (Baader *et al.* 2003), page 52. We call a concept name A *defined* in a TBox $\mathcal{T}$ and write $A \in \mathsf{def}(\mathcal{T})$ if A occurs on the left-hand side of a concept definition in $\mathcal{T}$. A *knowledge base (KB)* is a pair $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$ such that every auxiliary concept name used in $\mathcal{K}$ is in $\mathsf{def}(\mathcal{T})$. An interpretation $\mathcal{I}$ *satisfies* a concept definition $A \equiv C$ if $A^{\mathcal{I}} = C^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$, written $\mathcal{I} \models \mathcal{T}$, if $\mathcal{I}$ satisfies all concept definitions in $\mathcal{T}$. An interpretation $\mathcal{I}$ is a *model* of a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, written $\mathcal{I} \models \mathcal{K}$, if $\mathcal{I}$ is a model of $\mathcal{T}$ and $\mathcal{A}$. The set of all models of a KB $\mathcal{K}$ is denoted $M(\mathcal{K})$.

An *update* $\mathcal{U}$ is a simple and consistent ABox that does not use auxiliary concept names. We disallow auxiliary concept names because they can be defined in a TBox and thus allowing them is equivalent to admitting updates with complex concepts. Let $\mathcal{U}$ be an update and let $\mathcal{I}$ and $\mathcal{I}'$ be interpretations that agree on the interpretation of individual names. We define an update relation $\mathcal{I} \Longrightarrow^{p}_{\mathcal{U}} \mathcal{I}'$ (where p stands for "primary") as in Definition 1, but restrict the condition on concept names to primary concept names. This restriction is not harmful since we require auxiliary concept names that are used in a knowledge base to be defined in the TBox, and this means that their extension is uniquely determined by the extensions of the primary concept names and role names. Still, as a result of the restriction, the relation $\Longrightarrow^{p}_{\mathcal{U}}$ is not functional (in contrast to the case without TBoxes).

Definition 23 (Knowledge Base Update). Let $\mathcal{K}_1$ and $\mathcal{K}_2$ be knowledge bases, $\mathcal{K}_i = (\mathcal{T}_i, \mathcal{A}_i)$, and $\mathcal{U}$ an update. Then $\mathcal{K}_2$ *is a result of updating* $\mathcal{K}_1$ *with* $\mathcal{U}$ if

$$M(\mathcal{K}_2) = \{\mathcal{I}' \mid \exists \mathcal{I} \in M(\mathcal{K}_1) : \mathcal{I} \Longrightarrow^{p}_{\mathcal{U}} \mathcal{I}' \wedge \mathcal{I}' \models \mathcal{T}_2\}.$$

In this case, we write $\mathcal{K}_1 * \mathcal{U} \equiv_p \mathcal{K}_2$.

Observe that the TBox of the updated KB $\mathcal{K}_2$ can contain new abbreviations, i.e., definitions $A \doteq C$ with A an auxiliary concept names that does not occur in $\mathcal{K}_1$. Since there is more than a single way to define such abbreviations, the result of updating a knowledge base is not unique up to logical equivalence. However, we have this uniqueness when restricting our attention to what the updated ABox expresses regarding the primary concept names and role names, only.

In the equality in Definition 23, the conjunct $\mathcal{I}' \models \mathcal{T}_2$ has no impact on the "$\subseteq$" direction since all models of $\mathcal{K}_2$ are models of $\mathcal{T}_2$ anyway. For the "$\supseteq$" direction, the conjunct is essential: dropping it would mean to require that *every* possible interpretation of the auxiliary concept names in $\mathcal{I}'$ satisfies $\mathcal{T}_2$. Moreover, since $\mathcal{T}_2$ is part of the updated knowledge base $\mathcal{K}_2$, interpretations not satisfying $\mathcal{T}_2$ are irrelevant.

We now establish a relationship between updates of ABoxes and updates of knowledge bases. Let $\mathcal{T}$ be an acyclic TBox, and C a concept. The concept $C^{\mathcal{T}}$ obtained from C by exhaustively replacing defined concept names in C with their definitions from $\mathcal{T}$ is called the *unfolding of* C *w.r.t.* $\mathcal{T}$. If $\mathcal{A}$ is an ABox, then the *unfolding of* $\mathcal{A}$ *w.r.t.* $\mathcal{T}$ is the ABox $\mathcal{A}^{\mathcal{T}}$ obtained by replacing each concept assertion $C(a)$ in $\mathcal{A}$ with $C^{\mathcal{T}}(a)$. If $(\mathcal{T}, \mathcal{A})$ is a knowledge base, then the unfolding $\mathcal{A}^{\mathcal{T}}$ contains only primary concept names. The following lemma shows that updated knowledge bases are just updated ABoxes with abbreviations.

Lemma 24. *Let* $\mathcal{K}_1$ *and* $\mathcal{K}_2$ *be knowledge bases,* $\mathcal{K}_i = (\mathcal{T}_i, \mathcal{A}_i)$*, and* $\mathcal{U}$ *an update. Then*

$$\mathcal{K}_1 * \mathcal{U} \equiv_p \mathcal{K}_2 \quad \textit{iff} \quad \mathcal{A}_1^{\mathcal{T}_1} * \mathcal{U} \equiv \mathcal{A}_2^{\mathcal{T}_2}.$$

For the moment, the purpose of Lemma 24 is only to clarify the relation between ABox updates and knowledge base updates. Although we could compute knowledge base updates using Lemma 24 together with our construction for ABox updates, this would not help to obtain smaller updates.

Therefore, we now show how to directly construct updated knowledge bases in $\mathcal{ALCQIO}^{@}$ and its fragments. Let $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ be a knowledge base, and let $\mathcal{U}$ be an update. Diagrams for $\mathcal{U}$ and the sets $\mathfrak{D}$ and $\mathcal{D}_\mathcal{U}$ are defined as in the previous section. We use $\mathsf{sub}(\mathcal{K})$ to denote the set of all subconcepts of concepts occurring in $\mathcal{K}$. To construct the result of updating $\mathcal{K}$ with $\mathcal{U}$, we introduce a new concept name $A_C^\mathcal{D}$ for every diagram $\mathcal{D} \in \mathfrak{D}$ and every $C \in \mathsf{sub}(\mathcal{K})$. Let $\mathsf{trans}(C, \mathcal{D})$ denote the concept on the right-hand side of the clause for $C^{\mathcal{D}_\mathcal{U}}$ in Figure 4, with all subconcepts $E^\mathcal{D}$ replaced by the concept name $A_E^\mathcal{D}$. For example, $\mathsf{trans}(C \sqcap D, \mathcal{D}) = A_C^\mathcal{D} \sqcap A_D^\mathcal{D}$. For each diagram $\mathcal{D} \in \mathfrak{D}$, define a TBox

$$\mathcal{T}_{\mathsf{sub}}^\mathcal{D} := \{A_C^\mathcal{D} \equiv \mathsf{trans}(C, \mathcal{D}) \mid C \in \mathsf{sub}(\mathcal{K}) \setminus \mathsf{def}(\mathcal{T})\}.$$

Then, we define the TBox

$$\mathcal{T}' := \bigcup_{\mathcal{D} \in \mathfrak{D}} (\mathcal{T}_{\mathsf{sub}}^\mathcal{D} \cup \{A_A^\mathcal{D} \equiv A_C^\mathcal{D} \mid A \equiv C \in \mathcal{T}\}).$$

For every $\mathcal{D} \in \mathfrak{D}$, let

$$\begin{aligned} \mathcal{A}_{\mathcal{D}_\mathcal{U}} \quad := \quad & \{A_C^\mathcal{D}(a) \mid C(a) \in \mathcal{A}\} \cup \ \cup \\ & \{r(a,b) \mid r(a,b) \in \mathcal{A} \wedge \neg r(a,b) \notin \mathcal{D}_\mathcal{U}\} \cup \\ & \{\neg r(a,b) \mid \neg r(a,b) \in \mathcal{A} \wedge r(a,b) \notin \mathcal{D}_\mathcal{U}\} \end{aligned}$$

Now we can define the ABox $\mathcal{A}'$ by setting

$$\mathcal{A}' = \bigvee_{\mathcal{D} \in \mathfrak{D}} \bigwedge \mathcal{A}_{\mathcal{D}_\mathcal{U}} \cup \mathcal{D}_\mathcal{U} \cup (\mathcal{D} \setminus \neg \mathcal{D}_\mathcal{U}).$$

and finally assemble the updated knowledge base by setting $\mathcal{K}' := (\mathcal{T}', \mathcal{A}')$. It can be proved that this knowledge base is as required:

Lemma 25. $\mathcal{K} * \mathcal{U} \equiv_p \mathcal{K}'$

We now formulate the main result on updates with acyclic TBoxes. In constrast to updates without TBoxes, updated knowledge bases are polynomial in the size of the original KB. Thus, Lemma 24 implies that we can use acyclic TBoxes to obtain a more succinct presentation of updated ABoxes. In the following, the size $|\mathcal{T}|$ of a TBox $\mathcal{T}$ is $\sum_{A \equiv C \in \mathcal{T}} |C|$, and the size $|\mathcal{K}|$ of a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ is the sum of $|\mathcal{T}|$ and $|\mathcal{A}|$.

Theorem 26.
Let $\mathcal{L} \in \{\mathcal{ALCO}^{@}, \mathcal{ALCIO}^{@}, \mathcal{ALCQO}^{@}, \mathcal{ALCQIO}^{@}\}$. Then there are polynomials p_1, p_2, and q such that, for every $\mathcal{L}$-knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ and every update $\mathcal{U}$, there is an $\mathcal{L}$-knowledge base $\mathcal{K}'$ such that

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

It is important to note that Theorem 26 is true only if we assume unary coding of numbers: with binary coding, already the translation C^U results in an exponential blowup in the size of the original ABox since we have $|(\leqslant\ n\ r\ C)^\mathcal{U}| \in \mathcal{O}(2^n)$. Thus, the updated ABox will not be polynomial in the size of the original one.

As in the case without TBoxes, it can be shown that iterated updates do not produce a blowup of the size of updated ABoxes that is worse than the blowup produced by a single update.

Theorem 27. *There are polynomials p_1, p_2 such that the following holds: for all knowledge bases $\mathcal{K}_0, \ldots, \mathcal{K}_n$ and updates $\mathcal{U}_1, \ldots, \mathcal{U}_n$, if $\mathcal{K}_i$ is the ABox computed by our algorithm when $\mathcal{K}_{i-1}$ is updated with $\mathcal{U}_i$, for $0 < i \leq n$, then*

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

Small Updates in $\mathcal{ALCQIO}^{+}$

We have argued above that, if the update contains no role assertions, then updated $\mathcal{ALCQIO}^{@}$ ABoxes are polynomial in the size of the original ABox even without introducing TBoxes. Intuitively, updates with only concept assertions do not lead to an exponential blowup because we have available nominals, the @-operator, and the Boolean operators on concepts. 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 nominals. In this section, we explore the possibility of constructing updated ABoxes in a language in which such constructors are available. The language we consider is of almost the same expressive power as C^2, the two-variable fragment of first-order logic with counting quantifiers (Lutz, Sattler, & Wolter 2001).

Denote by $\mathcal{ALCQIO}^{+}$ the description logic extending $\mathcal{ALCQIO}^{@}$ by means of the role constructors $\cap$ (role intersection), $-$ (set-theoretic difference of roles), and $\{(a, b)\}$ (nominal roles). In this language, complex roles are constructed starting from role names and nominal roles, and then applying $\cap$, $-$, and the inverse role operator $\cdot^-$. The interpretation of complex roles is as expected:

- $\{(a, b)\}^\mathcal{I} = \{(a^\mathcal{I}, b^\mathcal{I})\}$, for all $a, b \in \mathsf{N_I}$;
- $(r_1 \cap r_2)^\mathcal{I} = r_1^\mathcal{I} \cap r_2^\mathcal{I}$;
- $(r_1 - r_2)^\mathcal{I} = r_1^\mathcal{I} - r_2^\mathcal{I}$.

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 (Pacholski, Szwast, & Tendera 2000; Pratt-Hartmann 2005). We now formulate the main result of this section:

Theorem 28. *There are 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 18. For $\mathcal{ALCQIO}^+$, the construction of the concepts $C^{\mathcal{U}}$ is much simpler: it suffices to replace every concept name A in C with

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

and every role name r in C with

$$r \cup \bigcup_{\neg r(a,b) \in \mathcal{U}} \{(a,b)\} \setminus (\bigcup_{r(a,b) \in \mathcal{U}} \{(a,b)\}).$$

The concepts $C^{\mathcal{U}}$ are of size polynomial in the size of C and $\mathcal{U}$. The ABox $\mathcal{A}'$ can then be constructed in the same way as in the proof of Theorem 18 and is polynomial in the size of $\mathcal{A}$, but exponential in the size of the update $\mathcal{U}$. ❑

Clearly, Theorem 28 is independent of the coding of numbers, and, also with iterated updates, updated ABoxes remain polynomial in the size of the original ABox. An alternative to working with a description logic such as $\mathcal{ALCQIO}^+$ is to work directly in the two-variable fragment with counting C^2. Then, a result analogous to Theorem 28 is easily obtained.

Outlook

There are two obvious directions for future work. The first direction is to alleviate the syntactic restriction posed on concepts appearing in updates in a controlled way. For example, research on propositional updates containing *disjunctions* (Zhang & Foo 2000; Herzig 1996; Lin 1996; Thielscher 2000a) suggests the feasibility of ABox updates with Boolean combinations of concept names. We conjecture that natural generalizations of the semantics proposed in the propositional case lead to useful notions of an ABox update under which the updates are still computable. The second direction for future work is to incorporate cyclic TBoxes into our framework. However, this direction appears to be considerably more difficult than the first one. As discussed in (Baader *et al.* 2005), it is not even clear if a satisfactory semantics can be defined in this case.

Acknowledgements We would like to thank Franz Baader and Michael Thielscher for ideas and discussions. The third author was supported by the DFG Graduiertenkolleg 334. The fourth author was partially supported by UK EPSRC grant no. GR/S63182/01.

References

Areces, C., and de Rijke, M. 2001. From description logics to hybrid logics, and back. In *Advances in Modal Logics Volume 3*. CSLI Publications.

Areces, C.; Blackburn, P.; Hernandez, B. M.; and Marx, M. 2003. Handling Boolean ABoxes. In *Proc. of the 2003 Int. Workshop on Description Logics*.

Baader, F.; McGuiness, D. L.; Nardi, D.; and Patel-Schneider, P. 2003. *The Description Logic Handbook: Theory, implementation and applications*. Cambridge University Press.

Baader, F.; Lutz, C.; Milicic, M.; Sattler, U.; and Wolter, F. 2005. Integrating description logics and action formalisms: First results. In *Proc. of the Twentieth National Conf. on AI (AAAI-05)*.

Borgida, A. 1996. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence* 82(1 - 2):353–367.

Cadoli, M.; Donini, F. M.; Liberatore, P.; and Schaerf, M. 1999. The size of a revised knowledge base. *Artificial Intelligence* 115(1):25–64.

Forbus, K. D. 1989. Introducing actions into qualitative simulations. In *Proc. of the Int. Joint Conf. on AI (IJCAI'89)*, 1273–1279. Morgan Kaufman.

Herzig, A. 1996. The PMA revisited. In *Proc. of the 5th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'96)*, 40–50. Morgan Kaufmann.

Kurtonina, N., and de Rijke, M. 1999. Expressiveness of concept expressions in first-order description logics. *Artificial Intelligence* 107(2):303–333.

Lin, F. 1996. Embracing causality in specifying the indeterminate effects of actions. In *Proc. of the 14th National Conf. on AI (AAAI-96)*, 670–676. MIT Press.

Liu, H.; Lutz, C.; Milicic, M.; and Wolter, F. 2005. Updating aboxes. LTCS-Report 05-10, Technical University Dresden. see http://lat.inf.tu-dresden.de/research/reports.html.

Lutz, C.; Sattler, U.; and Wolter, F. 2001. Modal logics and the two-variable fragment. In *Annual Conf. of the European Association for Computer Science Logic (CSL'01)*, LNCS. Paris, France: Springer Verlag.

M.P.Shanahan. 1997. *Solving the Frame Problem*. MIT Press.

Pacholski, L.; Szwast, W.; and Tendera, L. 2000. Complexity results for first-order two-variable logic with counting. *SIAM Journal on Computing* 29(4):1083–1117.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison-Wesley.

Pratt-Hartmann, I. 2005. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language, and Information* 14(3):369–395.

Reiter, R. 2001. *Knowledge in Action*. MIT Press.

Scherl, R., and Levesque, H. 2003. Knowledge, action, and the frame problem. *Artificial Intelligence* 144(1):1–39.

Thielscher, M. 2000a. Nondeterministic actions in the fluent calculus: Disjunctive state update axioms. In *Intellectics and Comput. Logic*. Kluwer Academic. 327–345.

Thielscher, M. 2000b. Representing the knowledge of a robot. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'00)*, 109–120. Morgan Kaufmann.

Winslett, M. 1990. *Updating Logical Databases*. Cambridge, England: Cambridge University Press.

Zhang, Y., and Foo, N. 2000. Updating knowledge bases with disjunctive information. *Computational Intelligence* 16:1–22.