A Knowledge Compilation Technique for $\mathcal{ALC}$ Tboxes

Ulrich Furbach and **Heiko Günther** and **Claudia Obermaier**
University of Koblenz

Abstract

Knowledge compilation is a common technique for propositional logic knowledge bases. A given knowledge base is transformed into a normal form, for which queries can be answered efficiently. This precompilation step is expensive, but it only has to be performed once. We apply this technique to knowledge bases defined in the Description Logic $\mathcal{ALC}$. We discuss an efficient satisfiability test as well as a subsumption test for precompiled concepts and Tboxes. Further we use the precompiled Tboxes for efficient Tbox reasoning. Finally we present first experimental results of our approach.

Introduction

Knowledge compilation is a technique for dealing with the computational intractability of propositional reasoning. It has been used in various AI systems for compiling knowledge bases offline into systems, that can be queried more efficiently after this precompilation. An overview about techniques for propositional knowledge bases is given in (Darwiche and Marquis 2002).

There are several techniques for Description Logics which are related to knowledge compilation techniques. An overview on precompilation techniques for Description Logics such as structural subsumption, normalization and absorption is given in (Horrocks 2003). To perform a subsumption check on two concepts, structural subsumption algorithms (Baader et al. 2003) transform both concepts into a normal form and compare the structure of these normal forms. However these algorithms typically have problems with more expressive Description Logics. Especially general negation, which is an important feature in the application of Description Logics, is a problem for those algorithms. In contrast to structural subsumption algorithms, our approach is able to handle general negation without problems. Normalization (Balsiger and Heuerding 1998) is a preprocessing technique for Description Logics, which eliminates redundant operators in order to determine contradictory as well as tautological parts of a concept. In many cases this technique is able to simplify subsumption and satisfiability problems. Absorption (Tsarkov and Horrocks 2006) is a technique which tries to eliminate general inclusion axioms from a knowledge base. Both absorption and normalization have the aim of increasing the performance of tableau based reasoning procedures. In contrast to absorption and normalization, our approach extends the use of preprocessing. We suggest to transform the concept into a normal form called linkless graph which allows an efficient consistency test. For this consistency test a tableau procedure is not necessary.

In the context of Description Logics, knowledge compilation has firstly been investigated in (Selman and Kautz 1996), where $\mathcal{FL}$ concepts are approximated by $\mathcal{FL}^-$concepts. More recently, (Bienvenu 2008) introduces a normal form called prime implicate normal form for $\mathcal{ALC}$ concepts which allows for a polynomial subsumption check. However up till now, prime implicate normal form has not been extended for Tboxes yet. In contrast to prime implicate normal form, our approach is able to precompile Tboxes.

In this paper we will consider the Description Logic $\mathcal{ALC}$ (Baader et al. 2003) and we adopt the notion of linkless formulas, as it was introduced in (Murray and Rosenthal 1993; 2003). In the first section we recall the basics of the Description Logic $\mathcal{ALC}$ and $\mathcal{ALE}$. The second section introduces the idea of our precompilation. After that we discuss an efficient satisfiability test for precompiled concepts and Tboxes. Further we introduce a method to efficiently answer certain subsumption queries from both precompiled concepts and Tboxes. At the end of this paper we present first experimental results of the implementation of our approach.

Preliminaries

First we introduce syntax and semantics of both the Description Logics $\mathcal{ALE}$ and $\mathcal{ALC}$. Complex $\mathcal{ALE}$ concepts C and D are built up from atomic concepts and atomic roles according to the following syntax rule:

$$C, D \rightarrow A \mid \top \mid \bot \mid \neg A \mid C \sqcap D \mid \exists R.C \mid \forall R.C$$

where A is an atomic concept and R is an atomic role. $\mathcal{ALC}$ has the additional rules $C, D \rightarrow \neg C \mid C \sqcup D$. Next we consider the semantics of $\mathcal{ALC}$ concepts. An interpretation $\mathcal{I}$ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a nonemty set which is the domain of the interpretation and $\cdot^{\mathcal{I}}$ is an interpretation function assigning to each atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to each atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. We extend the interpretation function to complex concepts by the following inductive definitions:

$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \quad \bot^{\mathcal{I}} = \emptyset \qquad (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}} \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \exists b\ (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} | \forall b\ (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$

A concept C is satisfiable, if there is an interpretation $\mathcal{I}$ with $C^{\mathcal{I}} \neq \emptyset$. We call such an interpretation a model for C. Further a terminological axiom has the form $C \sqsubseteq D$ or $C \equiv D$ where C, D are concepts and an axiom $C \sqsubseteq D$ ($C \equiv D$) is satisfied by an interpretation $\mathcal{I}$, if $C^{\mathcal{I}} \subset D^{\mathcal{I}}$ ($C^{\mathcal{I}} = D^{\mathcal{I}}$). A Tbox consists of a finite set of terminological axioms and is called satisfiable, if there is an interpretation satisfying all its axioms. Given an axiom $A \sqsubseteq B$ and a Tbox $\mathcal{T}$ we often want to know if $A \sqsubseteq B$ holds w.r.t. $\mathcal{T}$, which we denote by $A \sqsubseteq_{\mathcal{T}} B$ and call it a query to the Tbox $\mathcal{T}$. Further $A \sqsubseteq_{\mathcal{T}} B$ holds, if $A \sqsubseteq B$ is true in all models of $\mathcal{T}$. Another way to show that $A \sqsubseteq_{\mathcal{T}} B$ holds is to show that $\mathcal{T}$ together with $A \sqcap \neg B$ is unsatisfiable. In the following, unless stated otherwise, by the term concept, we denote $\mathcal{ALC}$ concepts given in NNF, i.e., negation occurs only in front of concept names. By *concept literal*, we denote a concept name or a negated concept name. Further by *literal* we denote a concept literal or a concept of the form $\exists R.C$ or $\forall R.C$. The size of a concept C is the number of concepts, roles and connectives used in C. For example the size of $A \sqcap \exists R.\neg B$) is 5.

Precompilation of $\mathcal{ALC}$ Concept Descriptions

The precompilation technique we use for an $\mathcal{ALC}$ concept C consists of two steps. In the first step we remove all so called *links* occurring in C. The notion of a link was firstly introduced for propositional logic (Murray and Rosenthal 1993). Intuitively links are contradictory parts of a concept which therefore can be removed preserving equivalence.

In the second step of the precompilation process we consider role restrictions. Given for example $C = \exists R.B \sqcap \forall R.D$. According to the semantics of $\mathcal{ALC}$ it follows from $x \in C^I$ that there is an individual y with $(x,y) \in R^I$ and $y \in (B \sqcap D)^I$. The concept $B \sqcap D$ is precompiled in the second step of the precompilation. This is repeated recursively until all concepts of reachable individuals are precompiled.

Definition 1. *For a given concept C, the set of its paths is defined as follows:*

$paths(\bot) = \emptyset$
$paths(\top) = \{\emptyset\}$
$paths(C) = \{\{C\}\}$, *if C is a literal*
$paths(C_1 \sqcup C_2) = paths(C_1) \cup paths(C_2)$
$paths(C_1 \sqcap C_2) = \{X \cup Y |$
$X \in paths(C_1)$ *and* $Y \in paths(C_2)\}$

The concept $C = \neg A \sqcap (A \sqcup B) \sqcap \forall R.(E \sqcap F)$ has the two paths $p_1 = \{\neg A, A, \forall R.(E \sqcap F)\}$ and $p_2 = \{\neg A, B, \forall R.(E \sqcap F)\}$. We typically use p to refer to both the path and the conjunction of the elements of the path when the meaning is evident from the context. We call a path inconsistent, if the conjunction of its elements is inconsistent. Further for a set of paths P, the set of minimal paths in P is defined as $minimal(P) = \{q \mid q \in P$ and $\neg \exists p \in P\ p \subset q\}$

Definition 2. *For a given concept C a* link *is a set of two complementary concept literals occurring in a path of C.*

Definition 3. *A concept C is called* linkless*, if C is in NNF and there is no path in C which contains a link.*

A link means that the formula has a contradictory part. Further if all paths of a formula contain a link, the formula is unsatisfiable. The special structure of linkless formulas in propositional logic allows us to consider each conjunct of a conjunction separately. Therefore satisfiability can be decided in linear time and it is possible to enumerate models very efficiently.

Removing Links

In propositional logic one possibility to remove links from a formula is to use *path dissolution* (Murray and Rosenthal 1993). The idea of this algorithm is to eliminate paths containing a link. The result of removing all links from a propositional logic formula F is called *full dissolvent* of F. Further path dissolution simplifies away all occurrences of $\top$ and $\bot$ in a formula. Note that in the worst case, the removal of links can cause an exponential blowup. Path dissolution can be used in the context of Description Logics as well. For this purpose we define a bijection between concepts and propositional logic formulas, which maps a concept C to a propositional logic formula $prop(C)$. This bijection maps each concept name A to a propositional logic variable a, further $\sqcap$ ($\sqcup$) to $\wedge$ ($\vee$), $\top$ ($\bot$) to $true$ ($false$) and $QR.C$ to a propositional logic variable Q_r_c with $Q \in \{\exists, \forall\}$. After mapping a concept C to a propositional formula $prop(C)$ we construct the full dissolvent of $prop(C)$. We then use $prop^{-1}$ to map the full dissolvent back to a concept. By $linkless(C)$ we denote the result of this. Note that if $prop(C)$ is unsatisfiable, $linkless(C) = \bot$.

Theorem 4. *Let C be a concept. Then* $linkless(C) \equiv C$.

Theorem 4 follows from the fact [Murray & Rosenthal 93], that path dissolution preserves equivalence in the propositional case.

Handling Role Restrictions

A linkless concept can still be inconsistent. Take $\forall R.B \sqcap \exists R.\neg B$ as an example. So it is not sufficient to remove links from a concept. Therefore, in the second step of the precompilation we consider role restrictions.

Definition 5. *Let C be a linkless concept, p a path in C. Further let* $\{\forall R.B_1, \ldots, \forall R.B_n\} \subseteq p$ *be the (possibly empty) set of all universal role restrictions w.r.t. R in p. Then the concept* $C' \equiv B_1 \sqcap \ldots \sqcap B_n$ *is called* potentially R-reachable *from C. If further* $\exists R.A \in p$*, the concept* $C'' \equiv A \sqcap B_1 \sqcap \ldots \sqcap B_n$ *is called* R-reachable *from C. Further p is called a path used to reach* C'' *(potentially reach* C'*) from C.*

Note that it is possible that a concept is (potentially) reachable from another concept via several paths. Since the removal of links preserves equivalence, we call both C'' and $linkless(C'')$ R-reachable from C. A concept C' is called (potentially) reachable from a linkless concept C, if it is (potentially) R-reachable from C for some role R. Further (potentially) reachable* is the transitive reflexive closure of the relation (potentially) reachable.

For example the following linkless concept: $C = (\exists R.(D \sqcup E) \sqcup A) \sqcap \forall R.\neg D \sqcap \forall R.E \sqcap B$ which has the two different paths $p_1 = \{\exists R.(D \sqcup E), \forall R.\neg D, \forall R.E, B\}$ and $p_2 = \{A, \forall R.\neg D, \forall R.E, B\}$. The concept $C' \equiv (D \sqcup E) \sqcap \neg D \sqcap E$ is reachable from C via path p_1 using $\{\exists R.(D \sqcup E), \forall R.\neg D, \forall R.E\}$.

Since a path p is inconsistent, if the conjunction of its elements is inconsistent, it is clear that p is inconsistent, iff there is a concept literal A with $A \in p$ and $\overline{A} \in p$ or there is an inconsistent concept which is reachable from C using p.

In the second step of the precompilation we precompile, i.e. remove all links from, all (potentially) reachable* concepts. The precompilation of all potentially reachable* concepts is necessary when we want to answer queries. For example the concept $\forall R.\neg D \sqcap \forall R.\neg E$ does not have any reachable concepts since no existential role restriction w.r.t. the role R is present. Asking a query to this concept can introduce the missing existential role restriction and can make a concept reachable. For example asking the query $\forall R.\neg D \sqcap \forall R.\neg E \sqsubseteq \forall R.(\neg D \sqcap \neg E)$ leads to checking the consistency of $\forall R.\neg D \sqcap \forall R.\neg E \sqcap \exists R.(D \sqcup E)$ which has a reachable concept.

Result of the Precompilation Process

The result of the precompilation of a concept C is a rooted directed graph (N, E) i.e a directed graph with exactly one source. The graph consists of two different types of nodes: path nodes PN and concept nodes CN. So $N = CN \cup PN$. Whereas each path node in PN is a set of paths in C and each node in the CN is a linkless concept. The set of edges is $E \subset (CN \times PN) \cup (PN \times CN)$. Since the concepts which are (potentially) reachable from a concept via a path p only depend on the role restrictions occurring in p, we regard all paths containing the same set of role restrictions as equivalent. A concept node C_i has a successor node for each set of equivalent paths in C_i and further there is an edge from each path node to the concept nodes of (potentially) reachable concepts. These edges are labeled by the role restrictions used to (potentially) reach the respective concept. Further each path node has exactly one preceding concept node, whereas a concept node can have more than one preceding path node.

Definition 6. *The precompilation of a concept C is a rooted directed graph (N, E), called linkless graph, with root $\mathit{linkless}(C)$ and for each set P_i of equivalent paths in C there is a subsequent path node. There is an edge from a path node P_i to the linkless graph of concept node C', if C' is (potentially) reachable from C via one of the paths in P_i. This edge is labeled by the set of role restrictions used to reach C' from C. Further for all path nodes P holds: $|\{C' | \langle C', P\rangle \in E\}| = 1$.*

Consider for example the linkless concept:

$$C \equiv (B \sqcap \neg E) \sqcup ((B \sqcup \neg A \sqcup (\exists R.A \sqcap A)) \sqcap \exists R.E \sqcap \forall R.\neg A)$$

Its linkless graph with root C is depicted in Fig. 1. C has four paths $p_1 = \{B, \neg E\}$, $p_2 = \{B, \exists R.E, \forall R.\neg A\}$, $p_3 = \{\neg A, \exists R.E, \forall R.\neg A\}$ and $p_4 = \{\exists R.A, A, \exists R.E, \forall R.\neg A\}$. There are three sets of equivalent paths: $\{p_1\}$, $\{p_2, p_3\}$ and $\{p_4\}$. For each set of equivalent paths, there is a successor path node. In the next step, reachable concepts are considered: for instance the concept $E \sqcap \neg A$ is reachable via the paths in the second set of paths using the role restrictions $\{\exists R.E, \forall R.\neg A\}$. Therefore there is an edge from the second path node to the concept node $E \sqcap \neg A$ with $\mathit{label}(\langle\{p_2, p_3\}, E \sqcap \neg A\rangle) = \{\exists R.E, \forall R.\neg A\}$. In the same way, the precompilation of all (potentially) reachable concepts are combined with the path nodes.

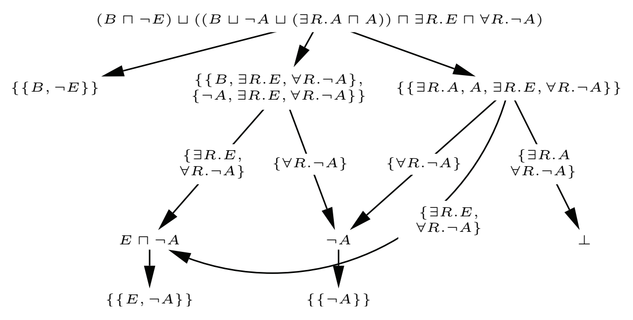
$(B \sqcap \neg E) \sqcup ((B \sqcup \neg A \sqcup (\exists R.A \sqcap A)) \sqcap \exists R.E \sqcap \forall R.\neg A)$
$\{\{B, \neg E\}\}$
$\{\{B, \exists R.E, \forall R.\neg A\}, \{\neg A, \exists R.E, \forall R.\neg A\}\}$
$\{\{\exists R.A, A, \exists R.E, \forall R.\neg A\}\}$
$\{\exists R.E, \forall R.\neg A\}$
$\{\forall R.\neg A\}$
$\{\forall R.\neg A\}$
$\{\exists R.A\ \forall R.\neg A\}$
$\{\exists R.E, \forall R.\neg A\}$
$E \sqcap \neg A$
$\neg A$
$\perp$
$\{\{E, \neg A\}\}$
$\{\{\neg A\}\}$

Figure 1: Example for a linkless graph

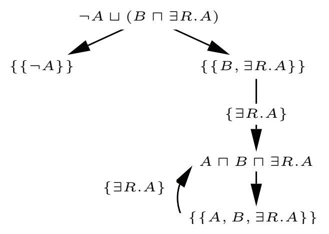
$\neg A \sqcup (B \sqcap \exists R.A)$
$\{\{\neg A\}\}$
$\{\{B, \exists R.A\}\}$
$\{\exists R.A\}$
$A \sqcap B \sqcap \exists R.A$
$\{\exists R.A\}$
$\{\{A, B, \exists R.A\}\}$

Figure 2: Example for a precompiled Tbox

Precompilation of General Tboxes

When answering queries with respect to a general Tbox it is necessary to restrict reasoning such that only models of this Tbox are considered. As described in (Baader et al. 2003) we transform a given Tbox $\mathcal{T} = \{C_1 \sqsubseteq D_1, \ldots, C_n \sqsubseteq D_n\}$ into a metaconstraint $\mathcal{M} = (\neg C_1 \sqcup D_1) \sqcap \ldots \sqcap (\neg C_n \sqcup D_n)$. The idea of the linkless graph can be directly extended to represent precompiled Tboxes. We simply construct the linkless graph for $\mathcal{M}$, but instead of just considering the concept nodes, each concept node C must also fulfill $\mathcal{M}$. So whenever there is a (potentially) reachable concept C, we precompile $C \sqcap \mathcal{M}$ instead of just C. Note that the precompilation of a Tbox typically contains cycles. For example the Tbox $\mathcal{T} = \{A \sqsubseteq B \sqcap \exists R.A\}$ can be transformed to $M_{\mathcal{T}} = \neg A \sqcup (B \sqcap \exists R.A)$. In Fig. 2 the result of the precompilation is depicted. There are two different paths in $M_{\mathcal{T}}$, which constitute the two subsequent path nodes. The path node $\{\{B, \exists R.A\}\}$ can be used to reach the concept $A \sqcap M_{\mathcal{T}}$ which is equivalent to the linkless concept $A \sqcap B \sqcap \exists R.A$, which therefore labels this concept node. Since $A \sqcap B \sqcap \exists R.A$ has the path $\{\{A, B, \exists R.A\}\}$, the graph contains a cycle. In the worst case there can be exponentially many reachable* concepts. Given r different roles each with n existential role restrictions, m universal role restrictions which are all nested with depth d, in the worst case the number of reachable* concepts is $r \cdot m \cdot 2^n \cdot d$. However in real world ontologies the number of reachable concepts is smaller. Furthermore precompiling a Tbox never increases the number of reachable concepts, contrariwise it usually decreases it. For example for the amino-acid [1] ontology $r = 5$, $d = 1$, $m = 3$ and $n = 5$. So in the worst case, there are 480 reachable* concepts. But in reality, before the precompilation there are 170 and after the precompilation 154 reachable concepts.

Next we will give an efficient consistency check for precompiled concepts and Tboxes.

Consistency

Definition 7. *Let C be a linkless concept and (N, E) its linkless graph. We call (N, E) inconsistent, if $C = \bot$ or for each P with $\langle C, P\rangle \in E$ there is a concept node C' which is reachable from C via one of the paths in P and the subgraph with root C' is inconsistent.*

Theorem 8. *Let C be a concept and (N, E) its linkless graph. Then holds: C is inconsistent iff (N, E) is inconsistent.*

By adding a label to each concept and path node in the linkless graph, it can be ensured that the consistency can still be checked in the presence of cycles.

Checking consistency of a linkless graph corresponds to searching in AND/OR graphs. (Mahanti, Ghose, and Sadhukhan 2003) presents a polynomial algorithm which is able to search AND/OR graphs in the presence of cycles.

Using the Linkless Graph to Answer Queries

In this section we will show that, given the precompilation of a concept C, subsumption queries $C \sqsubseteq D$ with $\neg D$ an $\mathcal{ALE}$ concept can be answered very efficiently.

In (Darwiche 2001) an operator called conditioning is used as a technique to answer queries from a precompiled knowledge base. The intuition of the conditioning operator is to consider $C \sqcap \alpha$ for a concept literal α and to simplify C according to α. Given for example $C = (B \sqcup E) \sqcap D$ and $\alpha = \neg B$, $C \sqcap \alpha$ can be simplified to $E \sqcap D \sqcap \neg B$. We will transfer the idea of this operator to linkless graphs.

Definition 9. *Let C be a linkless concept description, A be a consistent set of concept literals and $\alpha \in A$. Then C conditioned by A, denoted by $C|A$, is the concept description obtained by replacing each top level occurrences of α ($\overline{\alpha}$) in C by $\top$ ($\bot$) and simplifying the result according to the following simplifications:*

$\top \sqcap C = C$	$\top \sqcup C = \top$	$\bot \sqcap C = \bot$
$\bot \sqcup C = C$	$\exists R.\bot = \bot$	$\forall R.\top = \top$

Since this conditioning operator corresponds to the conditioning operator introduced in (Darwiche 2001), it is linear in the size of the concept C. From the way $C|\alpha$ is constructed, it follows that $C|\alpha \sqcap \alpha$ is equivalent to $C \sqcap \alpha$ and obviously $C|\alpha \sqcap \alpha$ is linkless.

[1] http://www.co-ode.org/ontologies/amino-acid/2006/05/18/amino-acid.owl

In the following we understand an $\mathcal{ALE}$ concept a set of its conjuncts. We now want to combine a precompiled concept with an $\mathcal{ALE}$ concept using the conditioning operator. For this it is essential to know how conditioning changes the set of paths in a concept.

Definition 10. *Let P be a set of paths and A a consistent set of concept literals. Then $P_{|A}$ is defined as:*

$$P_{|A} = \{p \setminus A' \mid p \in P \wedge p \cap A = A' \wedge \neg\exists\alpha \in p \text{ with } \overline{\alpha} \in A\}$$

Proposition 11. *Let C be a linkless concept and A a set of consistent concept literals. Then holds:*

$$minimal(paths(C|A)) = minimal(paths(C)_{|A})$$

Next we give an algorithm for the conditioning operator for an arbitrary node of a linkless graph.

Algorithm 12. *Let (N, E) be a linkless graph, $C \in N$ a concept node, B an $\mathcal{ALE}$ concept and β a literal. Then C conditioned by B w.r.t. (N, E) denoted by $condset((N, E), C, B)$ is the linkless graph calculated as follows:*

$$condset((N,E),C,B) = \\ = \begin{cases} condelem((N,E),C,\beta), \text{ if } B = \{\beta\} \\ condset((condelem((N,E),C,\beta), C|\beta \sqcap \beta, B'), \\ \text{if } \beta \text{ is a concept literal and } B = B' \cup \{\beta\}. \\ condset((condelem((N,E),C,\beta), C \sqcap \beta, B'), \\ \text{otherwise (with } B = B' \cup \{\beta\}). \end{cases}$$

Further $condelem((N, E), C, \beta)$ is the linkless graph calculated as follows:

1. *If β is a concept literal, substitute $C|\beta \sqcap \beta$ for C. Further for each P with $\langle C, P\rangle \in E$ substitute $minimal(P_{|\beta})$ for P and add β to all paths in $minimal(P_{|\beta})$. If $minimal(P_{|\beta}) = \emptyset$, remove its node and all its in- and outgoing edges. Return the resulting graph.*
2. *If $\beta = QR.D$ with $Q \in \{\exists, \forall\}$, substitute $C \sqcap QR.D$ for C and for each P with $\langle C, P\rangle \in E$ add $QR.D$ to all paths in P. Further:*
   - (a) *For all P whose paths do not contain a role restriction w.r.t. R except for $QR.D$: Create the linkless graph for D, add an edge from P to its root and label the edge with $\{QR.D\}$.*
   - (b) *For all P whose paths contain role restrictions w.r.t. R unlike $QR.D$:*
     - i. *For all P whose paths do not contain a universal role restriction w.r.t. R except for $QR.D$: Create the linkless graph for D, add an edge from P to its root and label the edge with $\{QR.D\}$.*
     - ii. *For all edges $\langle P, C'\rangle \in E$ whose label contains a role restriction w.r.t. R unlike $QR.D$,*
       - *If $Q = \forall$, add $\forall R.D$ to the label of $\langle P, C'\rangle$. Let (N', E') be the graph, resulting from the previous steps. If C' has only one ingoing edge, continue with the result of $condset((N', E'), C', D)$.*
       - *If $Q = \exists$ and $label(\langle P, C'\rangle)$ contains only universal role restrictions or $Q = \forall$ and C' has more than one ingoing edge then: Copy C', producing a new concept node C'' and create an edge $\langle P, C''\rangle$*

labeled with label($\langle P, C' \rangle$) $\cup \{QR.D\}$. For all P' with $\langle C', P' \rangle \in E$, copy P' to a new path node P'' and create an edge $\langle C'', P'' \rangle$. Further for each $C'_{P'}$ with $\langle P', C'_{P'} \rangle \in E$, create an edge $\langle P'', C'_{P'} \rangle$, label it with label($\langle P', C'_{P'} \rangle$). If $Q = \forall$ remove the edge $\langle P, C' \rangle$. Let (N', E') be the graph, resulting from the previous steps. Continue with the result of $condset((N', E'), C'', D)$.

iii. Return the resulting graph.

Algorithm 12 shows how to gradually condition a linkless graph by an $\mathcal{ALE}$ concept. However it is also possible to simultaneously calculate parts of the conditioning. If for example the $\mathcal{ALE}$ concept is $A \sqcap B \sqcap \neg C$ then the conditioning of a linkless graph with this concept can be calculated in one step since the literals all occur at the same level and therefore change the same parts of the linkless graph.

During the calculation of $condelem((N, E), C, \beta)$ the depth of nested role restrictions in β decreases, hence the calculation always terminates. If the role restrictions are nested with maximal depth d, the conditioning only affects concept nodes which are reachable in d steps from the root.

Case 2a corresponds to the precompilation of a part of the query, which in general can not be done in polytime. However it is reasonable to expect the query to be much smaller than the Tbox. Therefore it is not too harmful to do this precompilation during query time.

In order conjunctively combine a precompiled concept C with an $\mathcal{ALE}$ concept, we just have to condition the root node of the linkless graph for C with the $\mathcal{ALE}$ concept. We adapt our notation to Def. 9 and denote by $(N, E)|B$ the result of $condset((N, E), root(N, E), B)$.

Lemma 13. *Let C be a linkless $\mathcal{ALC}$ concept, (N, E) its linkless graph and B an $\mathcal{ALE}$ concept. Then $(N, E)|B$ is consistent, iff $C \sqcap B$ is consistent.*

Theorem 14. *Given a concept C, its linkless graph (N, E) and a subsumption query $C \sqsubseteq D$ with $\neg D$ an $\mathcal{ALE}$ concept. Then $C \sqsubseteq D$ holds, iff $(N, E)|\neg D$ is inconsistent.*

Theorem 14 follows directly from Lemma 13, since $C \sqsubseteq D$ holds iff $C \sqcap \neg D$ is inconsistent.

Let's now consider the concept C whose linkless graph is presented in Fig. 1. Asking the query $C \sqsubseteq B \sqcup \exists R.E$ leads to conditioning the linkless graph with $\neg B \sqcap \forall R.\neg E$. The result is the linkless graph depicted in Fig. 3, which is inconsistent. Therefore the subsumption query holds.

The linkless graph of a given Tbox $\mathcal{T}$ can be easily used to do Tbox reasoning. Let A be an $\mathcal{ALE}$ concept and B be a concept in NNF which is constructed only using the connectives disjunction and atomic negation. If we want to check whether a subsumption $A \sqsubseteq_{\mathcal{T}} B$ holds, we have to check the consistency of $A \sqcap \neg B \sqcap \mathcal{M}$ where $\mathcal{M}$ denotes the meta-constraint of $\mathcal{T}$. Assuming that we have the linkless graph for $\mathcal{M}$, we only have to condition it with $A \sqcap \neg B$. By performing a consistency check for the resulting graph, we can decide if the subsumption holds. Since in Tbox reasoning many queries are asked to the same Tbox, it is worthwhile to precompile the Tbox into a linkless graph.

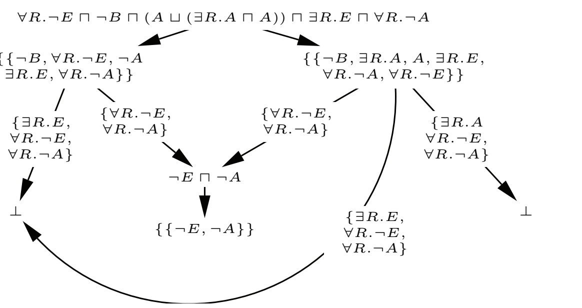
$\forall R.\neg E \sqcap \neg B \sqcap (A \sqcup (\exists R.A \sqcap A)) \sqcap \exists R.E \sqcap \forall R.\neg A$
$\{\{\neg B, \forall R.\neg E, \neg A, \exists R.E, \forall R.\neg A\}\}$
$\{\{\neg B, \exists R.A, A, \exists R.E, \forall R.\neg A, \forall R.\neg E\}\}$
$\{\exists R.E, \forall R.\neg E, \forall R.\neg A\}$
$\{\forall R.\neg E, \forall R.\neg A\}$
$\{\forall R.\neg E, \forall R.\neg A\}$
$\{\exists R.A \ \forall R.\neg E, \forall R.\neg A\}$
$\neg E \sqcap \neg A$
$\{\{\neg E, \neg A\}\}$
$\{\exists R.E, \forall R.\neg E, \forall R.\neg A\}$
$\perp$
$\perp$

Figure 3: Result of conditioning the linkless graph from Fig. 1 by $\neg B \sqcap \forall R.\neg E$.

Complexity of Query Answering

As described above, we transfer the query $A \sqsubseteq_{\mathcal{T}} B$ into an inconsistency test of $A \sqcap \neg B \sqcap \mathcal{M}$. We construct the linkless graph for $\mathcal{M}$ and use conditioning to answer the query. The way the structure of the query is restricted, ensures that we know $A \sqcap \neg B$ to be an $\mathcal{ALE}$ concept. Since the consistency of $\mathcal{ALE}$ concepts with respect to the empty Tbox can not be answered in polytime, we have to state the complexity of query answering using the linkless graph of a Tbox depending on the query.

Theorem 15. *Given a linkless graph of a Tbox $\mathcal{T}$ and a query $A \sqsubseteq_{\mathcal{T}} B$ such that $A \sqcap \neg B$ is $\mathcal{ALE}$, then holds: If $A \sqcap \neg B$ does not contain any role restrictions, the query can be answered in time linear to the size of the linkless Tbox and to the number of concept nodes in $\mathcal{T}$'s linkless graph.*

We owe this property to the fact in this case it's not necessary to extend the existing linkless graph.

In order to ease the investigation of the complexity of query answering for queries containing role restrictions, we assume that the Tbox under consideration is flattened (Rudolph, Krötzsch, and Hitzler 2008) i.e. all nested role restrictions are removed. This is done by replacing each occurrence of $\exists R.C$ ($\forall R.C$) in an axiom by $\exists R.C'$ ($\forall R.C'$) with C' a new concept name. Further we add the axioms $C' \sqsubseteq C$ and $C \sqsubseteq C'$. We repeat this transformation recursively until no nested role restrictions are left and for all role restriction $\exists R.B$ or $\forall R.B$, B is a newly introduced concept. This ensures that only concepts of the form $A \sqcap linkless(\mathcal{M})$ are (potentially) reachable, where A is a conjunction of newly introduced concept names and further that every concept which is reachable in more than one step is also reachable in exactly one step.

Theorem 16. *Let $\mathcal{T}$ be a flat Tbox, (N, E) its linkless graph and n be the number of path nodes, which follow $root(N, E)$. Further let $A \sqsubseteq_{\mathcal{T}} B$ be a subsumption query such that $A \sqcap \neg B$ is an $\mathcal{ALE}$ concept. Then holds: If $A \sqcap \neg B$ contains m role restrictions at the topmost level, each with a nesting depth d, then compared to (N, E), the graph $(N, E)|A \sqcap \neg B$ in the worst case has $m \cdot \sum_{i=1}^{d} n^i = -m + m \cdot \frac{1-n^{d+1}}{1-n}$ (for $n > 1$) additional concept nodes.*

In this case exponentially many new concept nodes are added to our graph. However this result is not too harmful

Ontology	Size: flat Tbox (NNF)	no. of roles	Size: linkless Tbox	removed links	reachable concepts	pot. reachable concepts
Foodswap	130	1	159	11	0	17
Ribosome	133	1	354	19	32	7
Nautilus-ex.	208	2	717	40	9	3
Koala	272	5	362	19	18	9
Amino-acid	1215	5	11024	130	154	88

Figure 4: Result of precompiling different ontologies.

since the number of concept nodes added is only exponential in the nesting depth of the query and it is reasonable to expect the nesting depth of the query to be low.

Implementation and Experiments

Based on the precompilation of Tboxes described above, we have implemented a prototypical knowledge compilation system. The implementation first flattens the Tbox as described in the previous section. We used different ontologies from the literature (Foodswap[1], Ribosome [2], Nautilus-exceptions [3], Koala [4] and Amino-acid [5]) to test our implementation. Some of the ontologies we used are not $\mathcal{ALC}$, but in order to be able to use them anyway, we ignore features not belonging to $\mathcal{ALC}$. Our implementation is not yet able to precompile the Dice [6] ontology, because of its size. But we are planning to optimize the removal of links as described in (Murray and Rosenthal 1993) in order to change that. Further, our system is able to answer queries from the linkless graph of a given Tbox. Since the removal of links from a Tbox in the worst case produces an exponential blowup, it is crucial to find out, if this blowup occurs when precompiling real ontologies. Fig. 4 gives information on that and shows that we removed 11 links form the Foodswap ontology, which caused the ontology only to grow from size 130 to 159. Further we removed 19 links from the Ribosome ontology which caused the Tbox size only to reduplicate. As Fig. 4 shows, for none of the precompiled ontologies, the feared exponential blowup occurred. Another point which is interesting is the number of different (potentially) reachable concepts depending on the number of different roles occurring in an ontology. Fig. 4 shows that, for the ontologies we considered, the number of reachable and potentially reachable worlds is manageable. Since the performance of query answering depends on the size of the linkless graph, our experiments confirm the fact, that the precompilation of a Tbox into a linkless graph is worthwhile.

[1]http://www.mindswap.org/dav/ontologies/commonsense/food/foodswap.owl

[2]http://www.co-ode.org/ontologies/bio-tutorial/Ribosome.owl

[3]http://www.co-ode.org/ontologies/bio-tutorial/Nautilus-exceptions.owl

[4]http://protege.stanford.edu/plugins/owl/owl-library/koala.owl

[5]http://www.co-ode.org/ontologies/amino-acid/2006/05/18/amino-acid.owl

[6]http://www.mindswap.org/ontologies/dice.owl

Conclusion and Future Work

We introduced a knowledge compilation technique for $\mathcal{ALC}$ concepts as well as for $\mathcal{ALC}$ Tboxes together with a query mechanism. We implemented our approach and first experiments led to promising results. Up till now we are only able to precompile $\mathcal{ALC}$ ontologies. One could argue that $\mathcal{ALC}$ is not expressive enough. This is why we want to investigate how to extend our approach to more expressive Description Logics like for example $\mathcal{SHOIN}$. Further we want to compare the performance of query answering from a precompiled Tbox by our implementation to existing DL reasoners.

Uniform interpolation is a helpful technique when different Tboxes have to be combined. Since linkless concepts are closely related to a normal form which allows efficient uniform interpolation, we expect the linkless graph of a concept to have this property too. When constructing the uniform interpolant for an $\mathcal{ALC}$ Tbox, things get more complicated, since uniform interpolants for $\mathcal{ALC}$ Tboxes need not exist. We are planning to focus our research on this area as well.

References

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *The Description Logic Handbook*. Cambridge University Press.

Balsiger, P., and Heuerding, A. 1998. Comparison of Theorem Provers for Modal Logics - Introduction and Summary. In *TABLEAUX*, volume 1397 of *LNCS*, 25–26. Springer.

Bienvenu, M. 2008. Prime Implicate Normal Form for ALC Concepts. In *Proceedings of the 21st International Workshop on Description Logics (DL2008)*.

Darwiche, A., and Marquis, P. 2002. A Knowlege Compilation Map. *Journal of Artificial Intelligence Research* 17:229–264.

Darwiche, A. 2001. Decomposable Negation Normal Form. *Journal of the ACM* 48(4).

Horrocks, I. 2003. Implementation and Optimization Techniques. In Baader et al. (2003), 306–346.

Mahanti, A.; Ghose, S.; and Sadhukhan, S. 2003. A Framework for Searching AND/OR Graphs with Cycles. *CoRR* cs.AI/0305001.

Murray, N., and Rosenthal, E. 1993. Dissolution: Making Paths Vanish. *J. ACM* 40(3):504–535.

Murray, N., and Rosenthal, E. 2003. Tableaux, Path Dissolution, and Decomposable Negation Normal Form for Knowledge Compilation. In *Proceedings of TABLEAUX 2003*, volume 1397 of *LNCS*. Springer.

Rudolph, S.; Krötzsch, M.; and Hitzler, P. 2008. Terminological Reasoning in SHIQ with Ordered Binary Decision Diagrams. In *Proceedings of the 23rd AAAI Conference on Artficial Intelligence (AAAI-08)*.

Selman, B., and Kautz, H. 1996. Knowledge Compilation and Theory Approximation. *J. ACM* 43(2):193–224.

Tsarkov, D., and Horrocks, I. 2006. FaCT++ Description Logic Reasoner: System Description. In *In Proc. of the Int. Joint Conf. on Automated Reasoning*, 292–297. Springer.