

Prime Implicate Normal Form for \mathcal{ALC} Concepts

Meghyn Bienvenu

IRIT, Université Paul Sabatier
31062 Toulouse Cedex, France
bienvenu@irit.fr

Abstract. In this paper, we present a new normal form for concept expressions in the description logic \mathcal{ALC} which is based on a recently introduced notion of prime implicate. We show that concepts in prime implicate normal form enjoy a number of desirable properties which make prime implicate normal form interesting from the viewpoint of knowledge compilation. In particular, we prove that subsumption between \mathcal{ALC} concepts in prime implicate normal form can be carried out in polynomial time using a simple structural subsumption algorithm reminiscent of those used for less expressive description logics. We provide a sound and complete algorithm for putting concepts into prime implicate normal form, and we investigate the spatial complexity of this transformation, showing there to be an at most doubly-exponential blowup in concept size. At the end of the paper, we compare prime implicate normal form to two other normal forms for \mathcal{ALC} that have been proposed in the literature, discussing the relative merits of the different approaches. Proofs of all results can be found in an accompanying technical report [1].

1 Introduction

Researchers have investigated a variety of strategies for coping with the high computational complexity of reasoning. Some have looked into restricted languages for which efficient reasoning is possible. Others have focused their efforts on the development of reasoning algorithms which perform well in practice, even if the worst-case complexity remains high. Still others have advocated the use of knowledge compilation [2], in which a knowledge base is put into a normal form which admits polytime querying, the idea being that the cost of the initial preprocessing will be offset by the computational savings made on later queries.

In the description logics community, the first two strategies have been privileged, while the third strategy, knowledge compilation, has remained largely unexplored. The likely explanation for this phenomenon is not a lack of interest on the part of this community but the simple fact that there have been no normal forms proposed in the literature which yield tractable reasoning for any reasonably expressive description logic.

Our paper aims to help remedy this situation by showing how prime implicate normal form, a well-studied normal form for propositional logic which has been influential in AI, can be extended to concept expressions in the description

logic \mathcal{ALC} . The starting point for this paper is our recent study [3, 4] of prime implicates for the modal logic \mathcal{K} , a known notational variant of \mathcal{ALC} . While the definition of prime implicates proposed in [3, 4] does not immediately yield a suitable notion of prime implicate normal form, it will play a key role in the definition we propose later in this paper. We will show that concepts in our normal form are much better behaved computationally than arbitrary \mathcal{ALC} concepts: it can be tested in constant time whether a concept in prime implicate normal form is satisfiable or tautologous and in quadratic time whether two concepts in prime implicate normal form are equivalent or if one subsumes the other. It is also easy to approximate concepts in prime implicate normal form over a sublanguage or up to a specified depth.

Our paper is organized as follows. In the first two sections, we recall the basics of the description logic \mathcal{ALC} and the notion of prime implicates in \mathcal{ALC} . In the following section, we propose a definition of prime implicate normal form for \mathcal{ALC} concepts, and we show that concepts in this form support a variety of polynomial-time queries and transformations. We then introduce an algorithm for putting concepts into prime implicate normal form and give some results concerning the spatial complexity of this transformation. At the end of the paper, we compare prime implicate normal form to two other normal forms for \mathcal{ALC} concepts, and then we conclude with a discussion of future work. Proofs have been omitted for lack of space. Refer to [1] for more details.

2 Preliminaries

In this section, we recall the syntax and semantics of the description logic \mathcal{ALC} as well as other useful notions and some necessary notation.

Concepts expressions in \mathcal{ALC} are built up from a set \mathcal{C} of atomic concepts and a set \mathcal{R} of atomic roles according to the following recursive definition:

$$C ::= \top \mid \perp \mid A \mid \neg C \mid C \sqcap C \mid C \sqcup C \mid \forall R.C \mid \exists R.C$$

where $A \in \mathcal{C}$ and $R \in \mathcal{R}$.

In analogy with classical logic, we will say that $C_1 \sqcap C_2$ is a *conjunction* (or intersection) of concepts, and we will call C_1 and C_2 *conjuncts* of $C_1 \sqcap C_2$. Likewise, we will say that $D_1 \sqcup D_2$ is a *disjunction* (or union) of concepts and that D_1 and D_2 are its *disjuncts*. Where convenient we will abuse notation and treat conjunction and disjunction as n -ary connectives. We will call a concept *propositional* if it does not contain any sub-concepts of the type $\forall R.C$ or $\exists R.C$. A concept is said to be in *negation normal form* (NNF) if negation only appears directly before atomic concepts. The *length* of a concept C , written $|C|$, is defined to be the total number of occurrences of atomic concepts and roles in C . The (role) *depth* of a concept C , noted $\delta(C)$, is defined to be the maximum number of nested $\exists R$ or $\forall R$ appearing in C . We define a *signature* to be any set of atomic roles and concepts. We define the *signature of a concept* C , written $\text{sig}(C)$, to be the set of atomic concepts and roles which appear in C .

The meaning of \mathcal{ALC} concepts is defined via a model-theoretic semantics. An interpretation (model) \mathcal{I} is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a non-empty set and $\cdot^{\mathcal{I}}$ is a function mapping each concept $A \in \mathcal{C}$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and each role $R \in \mathcal{R}$ to a relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. We extend $\cdot^{\mathcal{I}}$ to complex concepts as follows:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\perp^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b. (a, b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}}\} \\
(\exists R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b. (a, b) \in R^{\mathcal{I}} \text{ and } b \in C^{\mathcal{I}}\}
\end{aligned}$$

A concept C is said to be *satisfiable* if there is some interpretation \mathcal{I} for which $C^{\mathcal{I}} \neq \emptyset$. If there is no such model, then C is said to be *unsatisfiable*, and we write $\models C \sqsubseteq \perp$. A concept C is said to be *tautologous*, written $\models \top \sqsubseteq C$, just in the case that $\neg C$ is unsatisfiable. We say that a concept C is *subsumed* by D (or that D *subsumes* C), written $\models C \sqsubseteq D$, if for every model \mathcal{I} we have $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. Concepts C and D are said to be *equivalent*, written $\models C \equiv D$ if C subsumes D and D subsumes C .

3 Prime Implicates in \mathcal{ALC}

In this section, we define prime implicates for \mathcal{ALC} concepts and point out some of their key properties. All of the definitions and results in this section first appeared in [3] for the modal logic \mathcal{K} . We have adapted them to \mathcal{ALC} using the well-known correspondence [5] between formulae in \mathcal{K} and concept expressions in \mathcal{ALC} . For more details and for proofs of the results in this section, refer to [4].

Definition 1 (Literal/Clausal/Cubal Concepts). *We define literal, clausal, and cubal concepts as follows:*

$$\begin{aligned}
L &::= \top \mid \perp \mid A \mid \neg A \mid \forall R.D \mid \exists R.D \\
Cl &::= L \mid Cl \sqcup Cl \\
Cb &::= L \mid Cb \sqcap Cb \\
D &::= \top \mid \perp \mid A \mid \neg A \mid D \sqcap D \mid D \sqcup D \mid \forall R.D \mid \exists R.D
\end{aligned}$$

where $A \in \mathcal{C}$, $R \in \mathcal{R}$, and L , Cl , and Cb range respectively over the sets of literal concepts, clausal concepts, and cubal concepts.

In order to aid the presentation, we introduce the notation $Prop(Cl)$ to refer to the set of propositional literals which are disjuncts of the clausal concept Cl , and we will use $\exists R(Cl)$ (respectively $\forall R(Cl)$) to refer to the set of concepts C such that $\exists R.C$ (respectively $\forall R.C$) is a disjunct of Cl . For example, if $Cl = A \sqcup \exists R.A \sqcup \exists R.(B \sqcap \exists R.\top) \sqcup \forall S.B$, then $Prop(Cl) = \{A\}$, $\exists R(Cl) = \{A, B \sqcap \exists R.\top\}$,

$\exists S(Cl) = \forall R(Cl) = \emptyset$, and $\forall S(Cl) = \{B\}$. We will use the notation $Cl \setminus \{L\}$ to refer to the concept obtained from Cl by removing the disjunct L .

With a notion of clause in hand, we can define prime implicates just as in propositional logic:

Definition 2 (Prime implicate). *A clausal concept Cl is an implicate of a concept C if and only if $\models C \sqsubseteq Cl$. Cl is a prime implicate of C if and only if:*

1. *Cl is an implicate of C*
2. *If Cl' is an implicate of C such that $\models Cl' \sqsubseteq Cl$, then $\models Cl \sqsubseteq Cl'$*

This definition yields the standard notion of prime implicates when restricted to the propositional fragment of \mathcal{ALC} . It can also be shown to satisfy a number of properties of the propositional definition:

Proposition 1 (Finiteness). *The number of prime implicates of a concept is finite modulo logical equivalence.*

Proposition 2 (Covering). *Every implicate of a concept subsumes some prime implicate of the concept.*

Proposition 3 (Equivalence). *Every concept is equivalent to the intersection of its prime implicates.*

In [3], we proposed an algorithm for prime implicate generation. The algorithm, which we refer to by GEN-PI, has been shown to be sound and complete:

Proposition 4. *The algorithm GEN-PI always terminates, and it outputs exactly the set of prime implicates of the input concept.*

This algorithm will prove useful to us later on in the paper when we design a procedure for transforming a concept into an equivalent concept in prime implicate normal form.

4 Prime Implicate Normal Form

In this section, we introduce prime implicate normal form for \mathcal{ALC} concepts, show some of the nice properties it satisfies, and give an algorithm for putting concepts into prime implicate normal form. We also give some results concerning the size of concepts in prime implicate normal form.

4.1 Definition of Prime Implicate Normal Form

In propositional logic, a formula is said to be in prime implicate normal form if it is the conjunction of its prime implicates. We could define prime implicate normal form for \mathcal{ALC} concepts in exactly the same manner, but the normal form we obtain satisfies few of the nice properties of the propositional case. For example, under this definition, subsumption between two concepts in prime

implicate normal form is no easier than between arbitrary \mathcal{ALC} concepts. To see why, consider any pair of concepts C_1 and C_2 in negation normal form. The concepts $\exists R.C_1$ and $\exists R.C_2$ are their own prime implicates and hence are in prime implicate normal form according to the naive definition. As C_1 subsumes C_2 just in the case that $\exists R.C_1$ subsumes $\exists R.C_2$, we can reduce subsumption between arbitrary concepts in NNF to subsumption between concepts in prime implicate normal form. As the former problem is known to be PSPACE-complete, it follows that the latter is PSPACE-complete as well.

We remark, however, that the problem seems to come from the fact that the only the top layer of the concept is represented by its prime implicates, whereas the concepts appearing behind the role restrictions are left undecomposed. One idea then would be to require not only that the original concept be represented by its prime implicates but also that the sub-concepts appearing in the prime implicates be themselves represented by their prime implicates. This intuition is at the heart of our definition of prime implicate normal form for \mathcal{ALC} concepts:

Definition 3 (Prime Implicate Normal Form). *A concept C is in prime implicate normal form if and only if it satisfies one of the following conditions:*

1. $C = \perp$
2. $C = \top$
3. $\not\models C \sqsubseteq \perp$ and $\not\models \top \sqsubseteq C$ and $C = Cl_1 \sqcap \dots \sqcap Cl_p$ where
 - (a) each prime implicate of C is equivalent to some conjunct Cl_i
 - (b) every Cl_i is a prime implicate of C such that
 - i. if D is a disjunct of Cl_i , then $\not\models Cl_i \equiv Cl_i \setminus \{D\}$
 - ii. $|\exists R(Cl_i)| \leq 1$ for every role R
 - iii. if $E \in \exists R(Cl_i) \cup \forall R(Cl_i)$ for some R , then E is in prime implicate normal form
 - iv. if $E \in \exists R(Cl_i)$ and $F \in \forall R(Cl_i)$, then $\models E \sqsubseteq F$

Let us briefly go over the different points of the definition. The first two items state that all unsatisfiable concepts must be represented as \perp and all tautologous concepts must be represented as \top . All other concepts are to be represented as an intersection of their prime implicates, but we place some strong restrictions on how the prime implicates themselves are represented. First, we require that the prime implicates do not contain any unnecessary disjuncts (part (i) of condition (b)). We also stipulate that they contain at most one existential restriction per role (part (ii)) and that the concepts appearing behind the existential and universal restrictions be themselves in prime implicate normal form (part (iii)). Finally, we demand that if a prime implicate contains disjuncts $\exists R.E$ and $\forall R.F$ then E and F are such that $\models E \sqsubseteq F$ (part (iv)). This requirement may seem a little less intuitive than the others, but it ensures that our subsumption algorithm can treat the universal restrictions separately from the existential restrictions.

4.2 Properties of Our Normal Form

We will show later in the paper that our definition is well-founded by proving that every concept can be rewritten as an equivalent concept in prime implicate

normal form, but first we motivate the interest of doing so by exhibiting some of the nice properties of concepts in prime implicate normal form.

Tractable Querying The most important criterion when choosing a normal form for compilation is the set of polynomial time queries that the normal form supports. In [2], the authors enumerate a set of queries which they then use to compare different normal forms for propositional logic. Of the eight queries they consider, four are well-defined for \mathcal{ALC} : satisfiability-testing, tautology-testing, subsumption, and equivalence-testing. We show that for concepts in prime implicate normal form, all four queries are computable in polynomial time.

For satisfiability and tautology-testing, there is nothing to prove since by definition a concept C in prime implicate normal form is unsatisfiable just in the case that $C = \perp$ and is tautologous just in the case that $C = \top$. For subsumption and equivalence, we provide a structural subsumption algorithm Π -SUBSUME which decides subsumption between concepts in prime implicate normal form.

Function Π -SUBSUME(C_1, C_2): *decides if $\models C_1 \sqsubseteq C_2$*

1. If $C_1 = \perp$ or $C_2 = \top$, return **yes**.
 2. Return **no** if $C_1 = \top$ and $C_2 \neq \top$ or $C_2 = \perp$ and $C_1 \neq \perp$.
 3. For each conjunct G of C_2
 - Set *MatchFound* = *no*
 - For each conjunct H of C_1
 - Set *MatchFound* = *yes* if the following three conditions hold:
 - (a) $Prop(H) \subseteq Prop(G)$
 - (b) if $E \in \exists R(H)$, then Π -SUBSUME(E, E')=**yes** for some $E' \in \exists R(G)$
 - (c) if $F \in \forall R(H)$, then Π -SUBSUME(F, F')=**yes** for some $F' \in \forall R(G)$
 - If *MatchFound* = *no*, return **no**.
-
- Return **yes**.
-

We briefly explain the functioning of Π -SUBSUME. The first two steps treat limit cases where one or both of the concepts is unsatisfiable or tautologous. For all other pairs of concepts, we proceed to Step 3, in which we perform a structural comparison of the two concepts. We know from Proposition 3 that a concept C_1 is subsumed by a concept C_2 just in the case that C_1 is subsumed by each of the prime implicates of C_2 . Moreover, it follows from Proposition 2 that C_1 is subsumed by a prime implicate D of C_2 if and only if some prime implicate of C_1 is subsumed by D . As concepts in prime implicate normal form are conjunctions of their prime implicates, testing whether C_2 subsumes C_1 comes down to testing whether each conjunct of C_2 subsumes some conjunct of C_1 . If we hadn't placed any requirements on the form of the conjuncts, then this problem would be as hard as subsumption in general. But since C_1 and C_2 are in prime implicate normal form, their conjuncts have a particular structure which makes subsumption easy to test. We first check that the propositional literals in the first conjunct all appear in the second conjunct. We then call Π -SUBSUME on sub-concepts appearing in the two conjuncts in order to ensure that each of the existential and universal restrictions appearing in the first conjunct is subsumed

by an existential or universal restriction in the second. The algorithm performs these checks on each possible pair of conjuncts and returns **no** if it finds some conjunct of C_2 which does not subsume any conjunct of C_1 . If no such conjunct is found, the algorithm returns **yes** since every conjunct of C_2 has been shown to subsume a conjunct of C_1 , which means that C_2 subsumes C_1 .

Proposition 5. *If C_1 and C_2 are both in prime implicate normal form, then the algorithm Π -SUBSUME outputs yes on input (C_1, C_2) if and only if $\models C_1 \sqsubseteq C_2$.*

Proposition 6. *The algorithm Π -SUBSUME terminates in linear time in $|C_1| |C_2|$ (hence quadratic time in $|C_1| + |C_2|$) when given concepts C_1 and C_2 as input.*

Our algorithm requires that both input concepts be in prime implicate normal form. However, it is not always necessary for the second concept to be in prime implicate normal form to obtain polynomial time subsumption, as the following proposition demonstrates:

Proposition 7. *Let C be a concept in prime implicate normal form, and let D be a disjunction of propositional literals and concepts of the form $\exists R.Cl$ or $\forall R.Cl$ where Cl is a propositional clause. Then it can be decided in linear time in $|C|$ (and quadratic time in $|D|$) whether $\models C \sqsubseteq D$.*

The previous proposition can be extended to the entire class of concepts in NNF which do not contain conjunction while maintaining linear complexity in the first argument. Unfortunately, the complexity in the second argument is no longer polynomial since we need to test whether the query concept is a tautology and testing whether an arbitrary \mathcal{ALC} concept in NNF without conjunction is a tautology is an NP-complete problem [6].

We should also point out that the above category of concepts is just one example of a tractable class of query concepts. There are a variety of different syntactic conditions which can be placed on query concepts in order to guarantee polynomial subsumption.

Tractable Transformations Another criterion for choosing a normal form is the type of polynomial time transformations it permits. One of the most important transformations in propositional logic is forgetting (cf. [7]), in which we remove from a formula all reference to a given set of symbols while retaining as much of the formula's information as possible. Forgetting turns out to be closely related to the notion of uniform interpolation:

Definition 4 (\mathcal{L} -interpolant). *A concept C is said to be the uniform interpolant of a concept D with respect to the signature \mathcal{L} , or simply the \mathcal{L} -interpolant of D , if and only if $\text{Sig}(C) \subseteq \mathcal{L}$, $\models D \sqsubseteq C$, and $\models C \sqsubseteq E$ for every concept E such that $\models D \sqsubseteq E$ and $\text{Sig}(E) \subseteq \mathcal{L}$.*

Forgetting and uniform interpolation are two ways of looking at the same operation: the result of forgetting the signature S from a concept C is precisely the $\text{Sig}(C) \setminus S$ -interpolant of C .

We can show that \mathcal{L} -interpolants are easily computable when a concept is in prime implicate normal form. The algorithm is omitted for lack of space but is extremely simple: we traverse the concept removing those clausal subconcepts which contain a disjunct A or $\neg A$ with $A \notin \mathcal{L}$ or a disjunct of the form $\exists R.D$ or $\forall R.D$ with $R \notin \mathcal{L}$.

Proposition 8. *The \mathcal{L} -interpolant of a concept C in prime implicate normal form can be generated in linear time in the size of C .*

We also consider a different type of interpolation, in which instead of restricting our attention to concepts on a given signature we focus on concepts having less than a given depth:

Definition 5 (n -interpolant). *A concept C is the n -interpolant of a concept D if and only if $\delta(C) \leq n$, $\models D \sqsubseteq C$, and $\models C \sqsubseteq E$ for every concept E such that $\models D \sqsubseteq E$ and $\delta(E) \leq n$.*

The n -interpolant of a concept is easy to compute when the concept is in prime implicate normal form. We simply make a pass through the concept and when we reach a subconcept $\exists R.D$ or $\forall R.D$ appearing at level $n - 1$, we remove from D all conjuncts which are not propositional concepts.

Proposition 9. *The n -interpolant of a concept C in prime implicate normal form can be generated in linear time in $|C|$.*

4.3 Computing Prime Implicate Normal Form

We present below the algorithm PINF which transforms a given concept into an equivalent concept in prime implicate normal form.

Function $\text{PINF}(C)$: *returns a concept in prime implicate normal form equivalent to C*

1. If $\models C \sqsubseteq \perp$, return \perp . If $\models \top \sqsubseteq C$, return \top .
2. Set $\Sigma = \text{GEN-PI}(C)$.
3. For each P in Σ
 - (i) For each role R : if $\exists R(P) = \{D_1, \dots, D_m\}$ where $m > 1$, replace the disjuncts $\exists R.D_1, \dots, \exists R.D_m$ in P with the single disjunct $\exists R.(D_1 \sqcup \dots \sqcup D_m)$
 - (ii) For each role R : if $\exists R(P) = \{D\}$ and $\forall R(P) = \{E_1, \dots, E_n\}$, replace each disjunct $\forall R.E_i$ in P by $\forall R.(E_i \sqcup D)$
 - (iii) For each disjunct D in P : if $\models P \equiv P \setminus \{D\}$, replace P by $P \setminus \{D\}$.
 - (iv) For each disjunct $QR.D$ in P with $Q \in \{\exists, \forall\}$, replace $QR.D$ by $QR.\text{PINF}(D)$.
4. Return $\prod_{P \in \Sigma} P$.

The first step of our algorithm is to check whether the inputted concept is unsatisfiable or tautologous, in which case we return respectively \perp or \top . For all other concepts, we continue on to Step 2, where we use GEN-PI to generate the set of prime implicates of the inputted concept. We then modify the prime implicates so that they satisfy the conditions of Definition 3. We first check to see whether there are multiple existential restrictions for a single role, in which case we group them into a single existential restriction. We then make sure that

the concepts behind the universal restrictions are in the proper form by unioning them with the concept behind the existential restriction. We next check if each of the concepts in the clausal concept is necessary, and we remove all literal concepts which are found to be redundant. After that, we consider the concepts appearing behind a universal or existential restriction, and we put each of them into prime implicate normal form. Finally, in Step 4, we return the intersection of these modified prime implicates.

Proposition 10. *The algorithm PINF always terminates, and the concept it returns is a concept in prime implicate normal form which is equivalent to the inputted concept.*

It is well-known that in propositional logic the transformation to prime implicate normal form can result in a singly-exponential blowup in the size of the formula (cf. [8]). For \mathcal{ALC} concepts, the blowup can be doubly-exponential...

Proposition 11. *There exists a concept C such that the smallest equivalent concept in prime implicate normal form has length doubly-exponential in $|C|$.*

... but is never more than doubly-exponential:

Proposition 12. *Every concept C is equivalent to a concept in prime implicate normal form whose length is at most doubly-exponential in $|C|$.*

5 Related Work

Most of the subsumption algorithms that have been proposed for subpropositional description logics involve a normalization step in which concepts are put into some type of normal form. There has been relatively little work however on normal forms for more expressive description logics like \mathcal{ALC} which support disjunction. Two notable exceptions are the disjunctive form introduced for the mu-calculus in [9] and adapted to \mathcal{ALC} in [10] and the linkless normal form for \mathcal{ALC} recently proposed in [11].

5.1 Disjunctive Form

When restricted to propositional logic, the disjunctive form for \mathcal{ALC} concepts defined in [10] corresponds to standard disjunctive normal form (DNF). It follows that tautology-testing, subsumption, and equivalence-testing for concepts in disjunctive form must all be co-NP-hard since the problem of testing whether a formula in DNF is a tautology is known to be co-NP-complete. Satisfiability-testing remains polynomial [9]. Disjunctive form is better behaved when it comes to transformations: it is shown in [10] that the \mathcal{L} -interpolants of concepts in disjunctive form can be generated in linear time, and a similar result can be shown to hold for n -interpolants. As the transformation to disjunctive form involves an at most singly-exponential blowup [10], disjunctive form can be used to produce singly-exponential \mathcal{L} - and n -interpolants. Indeed, disjunctive form is used in [10] to prove the existence of singly-exponential-size \mathcal{L} -interpolants.

5.2 Linkless Normal Form

In [11], the authors show how linkless normal form can be extended from propositional formulae (cf. [12]) to concepts in \mathcal{ALC} . We can show that tautology-testing, subsumption, and equivalence-testing for linkless concepts are all co-NP-hard using a reduction to the DNF tautology problem. The authors of [11] have shown that satisfiability can be checked in linear time. They have also shown that the transformation to linkless normal form involves only a singly-exponential blowup in concept size and that subsumption can be carried out in linear time when the query concept is a disjunction of atomic literal concepts and of role restrictions followed by atomic literal concepts (this class is properly contained in the class of query concepts we introduced in Proposition 7). It was conjectured in [11] that \mathcal{L} -interpolants of concepts in linkless normal form can be easily generated, but the complexity of both \mathcal{L} - and n -interpolant generation is currently unknown.

5.3 Comparison

The results in this section suggest that prime implicate normal form is better suited than both disjunctive form and linkless normal form for the purposes of knowledge compilation, as prime implicate normal form supports the same class of polynomial transformations and a wider range of polynomial time queries. In particular, the fact that subsumption is polynomial between concepts in prime implicate normal form means that we can test whether an arbitrary query concept subsumes a concept in prime implicate normal form by first putting the (presumably small) query concept into prime implicate normal form and then using structural subsumption. For the other two normal forms, there is currently no procedure for posing arbitrary queries to compiled concepts.

On the other hand, disjunctive form and linkless normal form have the advantage of a lower spatial complexity. This means that if one is using a normal form for the sole purpose of generating \mathcal{L} - and n -interpolants, then disjunctive form is more appropriate since it produces singly-exponential-sized interpolants, whereas those obtained using prime implicate normal form may have doubly-exponential size.

6 Conclusion and Future Work

The main contribution of this paper is the introduction of prime implicate normal form as a new normal form for concepts in the description logic \mathcal{ALC} . We have shown that prime implicate normal form has a number of interesting properties which make it suitable for knowledge compilation, some of which are not satisfied by other normal forms proposed in the literature. We also provided an algorithm for transforming concepts into equivalent concepts in prime implicate normal form and proved that the transformation involves an at most doubly-exponential blowup in concept size.

In future work we would like to implement our prime implicate normal form transformation and our structural subsumption algorithm to see what kind of

performance they give in practice. This should help us to identify the type of situations in which the benefits gained by a concept being in prime implicate normal form outweigh the cost of putting it in this form.

Another interesting question for future research is how our normal form can be extended to handle even more expressive description logics. We expect that the extension to languages with nominals should be relatively straightforward, but that number restrictions and inverse roles will prove more challenging.

We also want to address what is probably the most important limitation of our work, namely the fact that our normal form treats concept expressions rather than TBoxes. We expect that the extension to TBoxes will be highly non-trivial, but we feel this is a question worth pursuing since it could potentially provide a new tool for dealing with the high complexity of TBox reasoning.

References

1. Bienvenu, M.: Prime implicate normal form for \mathcal{ALC} concepts. Technical Report IRIT/RR-2008-6-FR, IRIT, Université Paul Sabatier (2008) Available at <http://www.irit.fr/Meghyn.Bienvenu/papers/BienvenuRR-2008-6.pdf>.
2. Darwiche, A., Marquis, P.: A knowledge compilation map. *Journal of Artificial Intelligence Research* **17** (2002) 229–264
3. Bienvenu, M.: Prime implicates and prime implicants in modal logic. In: *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*. (2007) 397–384
4. Bienvenu, M.: Prime implicates and prime implicants in modal logic: Extended and revised version. Research report IRIT/RR-2007-17-FR, IRIT, Université Paul Sabatier, Toulouse (2007)
5. Schild, K.: A correspondence theory for terminological logics: preliminary report. In: *Proceedings of IJCAI-91, 12th International Joint Conference on Artificial Intelligence*. (1991) 466–471
6. Donini, F.M., Hollunder, B., Lenzerini, M., Marchetti Spaccamela, A., Nardi, D., Nutt, W.: The complexity of existential qualification in concept languages. *Artificial Intelligence* **53** (1992) 309–327
7. Lang, J., Liberatore, P., Marquis, P.: Propositional independence: Formula-variable independence and forgetting. *Journal of Artificial Intelligence Research* **18** (2003) 391–443
8. Chandra, A., Markowsky, G.: On the number of prime implicants. *Discrete Mathematics* **24** (1978) 7–11
9. Janin, D., Walukiewicz, I.: Automata for the modal μ -calculus and related results. In: *Proceedings of the Twentieth International Symposium on the Mathematical Foundations of Computer Science (MFCS'95)*. Volume 969 of *Lecture Notes in Computer Science*, Springer (1995) 552–562
10. ten Cate, B., Conradie, W., Marx, M., Venema, Y.: Definitorially complete description logics. In: *Proceedings of the Tenth International Conference on Principles of Knowledge Representation and Reasoning (KR06)*. (2006) 79–89
11. Furbach, U., Obermaier, C.: Knowledge compilation for description logics. In: *Proceedings of the 3rd Workshop on Knowledge Engineering and Software Engineering (KESE)*. (2007)
12. Murray, N., Rosenthal, E.: Dissolution: Making paths vanish. *Journal of the ACM* **40**(3) (1993) 504–535