Consequence Finding in $\mathcal{ALC}$

Meghyn Bienvenu

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

1 Introduction

Research on reasoning in description logics has traditionally focused on the standard inference tasks of (un)satisfiability, subsumption, and instance checking. In recent years, however, there has been a growing interest in so-called non-standard inference services (cf. [1]) which prove useful in the creation, evolution, and utilisation of description logic knowledge bases (KBs). In this paper, we propose *consequence finding* as a new non-standard reasoning task for description logics.

As its name suggests, consequence finding is concerned with the generation of a subset of the implicit consequences of a KB. This topic has been extensively studied in propositional logic (cf. [2]) where it has been shown to be relevant to a number of areas of AI, among them knowledge compilation, abduction, and non-monotonic reasoning. In the context of description logics, we view consequence finding as a tool to enable knowledge engineers and end users alike to better understand and access the contents of a description logic KB. We consider two possible applications:

Ontology Management The creation and maintenance of description logic KBs is a difficult and time-consuming task. Consequence finding might prove a useful aid in ontology design and evolution by allowing the knowledge engineer to have a better idea of both what information is contained in the KB and how additions may affect it. For instance, it would allow the knowledge engineer to check the relationships holding between a set of atomic concepts, or to evaluate the impact of adding a piece of new information to the KB.

Vague Queries An end user may not always have a specific query in mind but instead a general idea of the information that interests her. Consequence finding could help render DL KBs more accessible to users by allowing them to pose vague queries about the contents of the KB. Possible queries might be "Give me all the facts concerning penguins" or "Tell me what is known about Bob's students".

Consequence finding in DLs could take one of many forms depending on the nature of the input (concept, ABox, TBox, KB) and the type of consequences desired (concepts, assertions, axioms). In this paper, we investigate consequence finding for concept expressions in the standard description logic $\mathcal{ALC}$. This seemed a suitable starting point for our investigation as concept expressions are more basic than assertions and axioms, and $\mathcal{ALC}$ is a well-known and reasonably expressive description logic.

Our paper is organized as follows. In Section 2, we generalize the propositional notion of prime implicates to $\mathcal{ALC}$ concepts in order to provide a formal definition of the notion of a relevant consequence. More refined variants of prime implicates are then introduced to allow for more directed consequence finding, and the properties of these different forms of prime implicates are studied. In Section 3, we give sound and complete algorithms for computing prime implicates of concepts. In Section 4, we provide some first results concerning the complexity of prime implicate recognition. We conclude the paper with a discussion of future work.

2 $\mathcal{ALC}$ Prime Implicates

One of the first questions that presents itself when we talk about consequence finding is which consequences to generate? We obviously cannot generate *all* of the consequences, because even the simplest propositional formula has infinitely many consequences. Even if we restrict ourselves to one consequence per equivalence class, we still produce a lot of clearly irrelevant or redundant consequences. Since one of the aims of consequence finding is to make the contents of the KB more accessible to users, we clearly need a way of focusing in on the relevant subset of consequences.

In propositional logic, the solution lies in considering only the strongest clausal consequences of a formula, which are known as its *prime implicates*. By focusing on clauses, we avoid redundancies, and by only considering the logically strongest consequences, we eliminate weaker, irrelevant consequences. As every formula can be rewritten as a conjunction of clauses, and every clausal consequence of a formula is entailed by some prime implicate of the formula, prime implicates provide a complete picture of a formula's consequences.

In order to generalize the notion of prime implicates to $\mathcal{ALC}$ concepts, we need to come up with a suitable definition of $\mathcal{ALC}$ clausal concepts. In [3], a number of different potential definitions are compared, and two are singled out as being most suitable for the purposes of consequence finding.

The first definition is inspired by the notion of modal atom proposed in [4]. It defines literal concepts as the concepts in NNF that cannot be decomposed propositionally:

Definition 1. *Literal concepts, clausal concepts, and cubal concepts are defined as follows (where A is an atomic concept and R an atomic role):*

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

The second definition defines literal concepts as those concepts in NNF that cannot be decomposed *modally*, resulting in a more restrictive clausal form.

Definition 2. *Literal, clausal, and cubal concepts are defined as follows:*

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

Both definitions yield the standard notion of literals, clauses, and cubes when restricted to the propositional fragment of $\mathcal{ALC}$. They can also be shown to satisfy a number of properties of the propositional definition:

Proposition 1. *For both Definition 1 and Definition 2, we have:*

1. *Clausal and cubal concepts are simply unions and intersections of literal concepts.*
2. *The negation of a literal concept is equivalent to a literal concept. Negations of clausal (resp. cubal) concepts are equivalent to cubal (resp. clausal) concepts.*
3. *Every concept C is equivalent to a finite intersection of clausal concepts $D = D_1 \sqcap ... \sqcap D_n$ such that (a) D has the same depth as C, and (b) the size of D is at most singly exponential in the size of C. Likewise every concept is equivalent to a finite union of cubal concepts.*

Definition 1 has the advantage of yielding a more compact representation while Definition 2 provides a more fine-grained decomposition of concepts. To simplify the presentation, we will henceforth consider only the second definition, but all of our results hold equally well with respect to the first definition.

Now that we have selected a definition of clausal concepts, we can define prime implicates in exactly the same manner as in propositional logic:[1]

Definition 3. *A clausal concept Cl is an implicate of a concept C if and only if $\models C \sqsubseteq Cl$. A clausal concept 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 gives the standard notion of prime implicates, in which all of the clausal consequences of a concept are considered, but for many applications, only some of the consequences are of interest. This motivates the introduction of more refined notions of prime implicates in which additional restrictions are placed on implicates. In this paper, we consider the following three refined versions of prime implicates:

C-prime implicates: the C-prime implicates of a concept D are defined as the most specific clausal concepts which subsume $C \sqcap D$ but do not subsume C

only-$\mathcal{L}$-prime implicates: the only-$\mathcal{L}$-prime implicates of a concept D are the most specific clausal concepts which both subsume D and contain only those atomic concepts and roles in $\mathcal{L}$

about-$\mathcal{L}$-prime implicates: the about-$\mathcal{L}$-prime implicates of a concept D are the most specific clausal concepts which subsume D and which contain non-trivially all symbols in $\mathcal{L}$ (i.e. such that all equivalent concepts contain all symbols in $\mathcal{L}$)

We have chosen to study these particular variants because they seem interesting from an applications point of view. Once appropriately extended to TBox axioms, C-prime implicates could be used by the knowledge engineer to see how a new axiom will affect the ontology $\mathcal{O}$ under construction ("What are all of the $\mathcal{O}$-prime implicates of $axiom$?"), whereas only-$\mathcal{L}$-prime implicates could allow him to explore the relationships between a set S of atomic concepts ("What are all of the S-prime implicates of $\mathcal{O}$?"). Finally, about-$\mathcal{L}$-prime implicates can be used to generate all the consequences on a particular topic of interest and are thus very useful for vague querying. It is this type of prime implicate which would enable the user to find all the information concerning penguins.

[1] The dual notion of prime implicants can also be straightforwardly defined, but here we consider only prime implicates as they are the most relevant to purposes of the current paper.

Example 1. Consider the following concept expression $\mathcal{Q}$:

$$A \sqcap (B \sqcup C) \sqcap \exists R.\top \sqcap \forall R.(B \sqcap (A \sqcup C)) \sqcap \forall R.(B \sqcup D)$$

The prime implicates of Q are A, $B \sqcup C$, $\exists R.(B \sqcap A) \sqcup \exists R.(B \sqcap C)$, $\forall R.B$, and $\forall R.(A \sqcup C)$. There is just one only-$\{A\}$-prime implicates of $\mathcal{Q}$, the atomic concept A. There are three about-$\{A\}$-prime implicates of $\mathcal{Q}$: A, $\exists R.(B \sqcap A) \sqcup \exists R.(B \sqcap C)$, and $\forall R.(A \sqcup C)$. The $\mathcal{Q}$-prime implicates of $\forall R.\neg C$ are $\forall R.\neg C$, $\forall R.A$, and $\exists R.A \sqcap B$.

It is not hard to see that standard prime implicates can be recovered as special cases of C-, only-$\mathcal{L}$-, and about-$\mathcal{L}$-prime implicates. The following proposition further clarifies the relationship between standard and refined prime implicates:

Proposition 2.

1. *Every C-prime implicate of a concept D is a prime implicate of $C \sqcap D$.*
2. *Every about-$\mathcal{L}$-prime implicate of a concept D is a prime implicate of D.*
3. *An only-$\mathcal{L}$-prime implicate of a concept D may not be a prime implicate of D.*

To see why (3) holds, consider the concept $\exists R.A$ which is an only-$\{A, R\}$-prime implicate of $\exists R.(A \sqcap (\forall R.B))$ but not a standard prime implicate.

We now consider some important properties of our notions of prime implicates:

Proposition 3. *The set of prime implicates satisfy the following properties:*

Finiteness *The number of standard, C-, only-$\mathcal{L}$-, and about-$\mathcal{L}$-prime implicates of a concept is finite modulo logical equivalence.*

Covering *Every standard, C-, only-$\mathcal{L}$-, or about-$\mathcal{L}$-implicate of a concept subsumes respectively some standard, C-, only-$\mathcal{L}$-, or about-$\mathcal{L}$-prime implicate of the concept.*

Distribution *The prime implicates (respectively C-, only-$\mathcal{L}$-prime implicates) of a concept $C_1 \sqcup ... \sqcup C_n$ are equivalent to the logically strongest unions of the prime implicates (respectively C-, only-$\mathcal{L}$-prime implicates) of the C_i.*

Finiteness ensures that the prime implicates of a concept can be finitely represented, which is of course essential from a computational perspective. **Covering**, as its name suggests, implies that every implicate of a concept is "covered" by one of its prime implicates. For the standard definition of prime implicates, **Covering** implies that the set of prime implicates of a concept is equivalent to the concept itself, thus guaranteeing that no information is lost in replacing a formula by its prime implicates. For $\mathcal{L}$-concepts, **Covering** implies that the set of $\mathcal{L}$-prime implicates is equivalent to the uniform interpolant[2] of the concept with respect to $\mathcal{L}$. **Distribution** is at the heart of several prime implicate generation algorithms in propositional logic, and we will exploit this property in our own algorithms presented in the next section. Notice however that **Distribution** does not hold for about-$\mathcal{L}$-prime implicates since some disjuncts of about-$\mathcal{L}$-concepts may not contain all (or any) of the symbols in $\mathcal{L}$.

[2] We recall the *uniform interpolant* of a concept D with respect to a signature $\mathcal{L}$ is the most specific concept built from $\mathcal{L}$ which subsumes D.

3 Prime Implicate Generation

Figure 1 presents algorithms for computing the prime implicates of a concept for the different forms of prime implicates that we have defined. Our algorithms make use of the following auxilliary functions:

- **DNF-1** and **DNF-2** which return a set of satisfiable cubal concepts (w.r.t. Definitions 1 and 2 respectively) whose union is equivalent to the input concept
- **INT**$_{\mathcal{L}}$ which returns a finite union of cubal concepts (w.r.t. Definition 2) which is equivalent to the uniform interpolant of the input concept w.r.t. $\mathcal{L}$

We remark that we can implement these functions so that the size of the output of these functions is at most singly exponential in the size of the input concept.

Function PI(X): *generates standard prime implicates of X*
Step 1: If X is unsatisfiable, return $\{\bot\}$. Otherwise, set $\mathcal{T} = \textbf{DNF-1}(X)$.
Step 2: For each $T \in \mathcal{T}$: let L_T be the set of propositional literals in T, let $\mathcal{E}_T^R$ be the set of concepts E such that $\exists R.E$ is in T, and let $\mathcal{U}_T^R$ be the intersection of concepts U such that $\forall R.U$ is in T. Set

$$\Delta(T) = L_T \cup \bigcup_R (\{\forall R.P | P \in \textbf{PI}(\mathcal{U}_T^R)\} \cup \{ \bigsqcup_{H_i \in \textbf{DNF-2}(E \sqcap \mathcal{U}_T^R)} \exists R.H_i | E \in \mathcal{E}_T^R\}$$

Step 3: Set CANDIDATES $= \{\bigsqcup_{T \in \mathcal{T}} G_T \,|\, G_T \in \Delta(T)\}$.
Step 4: For each $C_j \in$ CANDIDATES: if $\models C_k \sqsubseteq C_j$ for some $C_k \in$ CANDIDATES with $k > j$, then delete C_j from CANDIDATES.
Step 5: Return CANDIDATES.

We now show how to modify **PI** to generate other types of prime implicates. For only-$\mathcal{L}$-prime implicates, we define $\Delta(T)$ as follows where $L_T^{\mathcal{L}} = L_T \cup \{\top\} \setminus \{(\neg)A | A \notin \mathcal{L}\}$:

$$\Delta(T) = L_T^{\mathcal{L}} \cup \bigcup_{R \in \mathcal{L}} (\{\forall R.P | P \in \textbf{PI}(\mathcal{U}_T^R)\} \cup \{\bigsqcup_i \exists R.H_i \,|\, \textbf{INT}_{\mathcal{L}}(E \sqcap \mathcal{U}_T^R) = \bigsqcup_i H_i \text{ and } E \in \mathcal{E}_T^R\})$$

For C-prime implicates, we eliminate from $\Delta(T)$ all implicates of C, and for about-$\mathcal{L}$-prime implicates, we remove from CANDIDATES all concepts which are not about-$\mathcal{L}$-concepts. Notice that for C- and about-$\mathcal{L}$-prime implicates, the modifications only affect the outermost call to **PI**; recursive calls use the same algorithm as for standard prime implicates.

Fig. 1. Algorithms for prime implicate generation.

Our algorithms all work in a similar manner. In Step 1, we check whether the input concept X is unsatisfiable, outputting $\bot$ if this is the case. For satisfiable X, we set $\mathcal{T}$ equal to a set of satisfiable concepts whose union is equivalent to X. We know from the distribution property (Proposition 3) that every prime implicate of X is equivalent to some union of prime implicates of the concepts in $\mathcal{T}$. In Step 2, we construct a set $\Delta(T)$ of clausal concepts for each $T \in \mathcal{T}$ in such a way that every prime implicate of T is equivalent to some element in $\Delta(T)$[3]. This means that in Step 3, we are guaranteed that every prime implicate of the input concept is equivalent to some candidate prime implicate in CANDIDATES. During the comparison phase in Step 4, non-prime candidates are eliminated, and exactly one prime implicate per equivalence class is retained.

The algorithms differ in their definition of $\Delta(T)$ in Step (2). For standard prime implicates, we set $\Delta(T)$ equal to the propositional literals in T (L_T) plus the strongest $\forall$-literal concepts implied by T ($\bigcup_R(\{\forall R.P | P \in \mathbf{PI}(\mathcal{U}_T^R)\})$ plus the strongest $\exists$ clausal concepts implied by T ($\{\bigsqcup_{H_i \in \mathbf{DNF\text{-}2}(E \sqcap \mathcal{U}_T^R)} \exists R.H_i | E \in \mathcal{E}_T^R\}$, i.e. the concepts $\exists R.(E \sqcap \mathcal{U}_T^R)$ put into clausal form). It can be shown that every standard prime implicate of T must be equivalent to one of the elements in $\Delta(T)$ (note however that some elements in $\Delta(T)$ may not be prime implicates). For only-$\mathcal{L}$-prime implicates, we modify $\Delta(T)$ in order to ensure that the elements of $\Delta(T)$ contain only the symbols in $\mathcal{L}$ (and that they include all only-$\mathcal{L}$-prime implicates of T). For C-prime implicates, we remove from $\Delta(T)$ all implicates of C. Finally, for about-$\mathcal{L}$-prime implicates, we use the same $\Delta(T)$ as for standard prime implicates, but we eliminate from CANDIDATES all clausal concepts which are not about-$\mathcal{L}$-concepts.

Proposition 4. *The prime implicate generation algorithms presented in Figure 1 are sound, complete, and always terminate.*

Our algorithms correspond to the simplest possible implementation of the distribution property, and it is well-known that naive implementations of the distribution property are computationally infeasible even for propositional logic. More efficient versions of our algorithms can be obtained using techniques developed for propositional logic, cf. [2]. For instance, instead of generating all of the candidate concepts and *then* comparing them, we can build them incrementally, comparing them as we go.

By performing induction on the depth of the input concept, it is possible to place an upper bound on the size of the prime implicates generated by our algorithms:

Proposition 5. *The size of the smallest representation of a standard, C-, only-$\mathcal{L}$-, and about-$\mathcal{L}$-prime implicate of a concept is at most singly exponential in the size of the concept.*

The following proposition shows that this bound is optimal.

Proposition 6. *The size of the smallest representation of a standard, C-, only-$\mathcal{L}$-, or about-$\mathcal{L}$-prime implicate of a concept can be exponential in the size of the concept.*

Proof. Consider the concept $(\bigsqcap_{i=1}^{n}(\forall R.(A_{i1} \sqcup A_{i2})) \sqcap \exists R.\top$ and its prime implicate $\bigsqcup_{i_k \in \{1,2\}} \exists R.(A_{1i_1} \sqcap ... \sqcap A_{ni_n})$.

4 Prime Implicate Recognition

Prime implicate recognition consists in deciding whether a given concept is a prime implicate of another. The purpose of this section is to study the complexity of this decision problem for the different notions of prime implicates that we have introduced.

[3] A worked-out example of Step 2 for standard prime implicates can be found in [3].

It is not hard to see that this decision problem must be at least as difficult as unsatisfiability: a concept is unsatisfiable just in the case that it has $\bot$ as a prime implicate (irrespective of the notion of prime implicate considered).

Proposition 7. *Recognition of standard, C-, only-$\mathcal{L}$-, and about-$\mathcal{L}$-prime implicates are all* PSPACE*-hard problems.*

In order to obtain an upper bound, we exploit Proposition 5 which tells us that there is some polynomial p such that for every concept C the size of its prime implicates is bounded by $2^{p(|C|)}$. This leads to a simple non-deterministic procedure for determining if a clausal concept Cl is a prime implicate of a concept C. We simply guess a clausal concept W of size at most $2^{p(|C|)}$ and check whether W is an implicate of C which is subsumed by Cl and does not subsume Cl. If this is the case, then Cl is not a prime implicate (we have found a more specific implicate of C), otherwise, there exists no stronger implicate, so Cl is indeed a prime implicate.

Proposition 8. *Recognition of standard, C-, only-$\mathcal{L}$-, and about-$\mathcal{L}$-prime implicates are all in* EXPSPACE[4].

The following proposition improves on the above complexity bounds.

Proposition 9. *We have the following:*

1. *Standard prime implicate recognition is in* EXPTIME.
2. *C-prime implicate recognition is in* EXPTIME[5].
3. *about-$\mathcal{L}$-prime implicates recognition is in* NEXPTIME.
4. *only-$\mathcal{L}$-prime implicate recognition is* CONEXPTIME*-hard.*

Proof. To demonstrate (1), we have constructed an algorithm for deciding standard prime implicate recognition in single-exponential time. Our algorithm first checks that the clausal concept is indeed an implicate and then verifies that each of the component literals is as specific as possible. Refer to [3] for more details.

(2) follows directly from (1) since C-prime implicates are just standard prime implicates which are not implied by C (by Proposition 2).

(3): We can check whether a concept Cl is an about-$\mathcal{L}$-prime implicate of a concept D in three steps: first, we check that the Cl contains all symbols in $\mathcal{L}$ (linear time), next, we verify that Cl is indeed a standard prime implicate of D (exponential time by (1)), and finally, we ensure that each symbol in $\mathcal{L}$ appears non-trivially in Cl. For this last step, it suffices to show that for each symbol $s \in \mathcal{L}$ the uniform interpolant UI_s of Cl over $Sig(Cl) \setminus \{s\}$ is not subsumed by Cl. This can be accomplished in non-deterministic exponential time by guessing and verifying an open branch of the tableaux of $UI_s \sqcap \neg Cl$ for each s.

For (4), we reduce the conservative extension decision problem for $\mathcal{K}$ (proven CONEXPTIME-complete in [5]) to the only-$\mathcal{L}$-prime implicate recognition problem. We recall that a formula $\phi_1 \wedge \phi_2$ is a conservative extension of ϕ_1 if for every formula ψ

[4] While the proof given here is non-constructive, we do have constructive exponential-space algorithms which we were unable to include for lack of space.

[5] Here we assume that the concept C is considered as part of the input

with $var(\psi) \subseteq var(\phi_1)$ we have $\phi_1 \wedge \phi_2 \models \psi$ implies $\phi_1 \models \psi$. The reduction is straightforward: $\phi_1 \wedge \phi_2$ is a conservative extension of a cubal formula ϕ_1 if and only if $\exists R.f(\phi_1)$ is a $var(\phi_1) \cup \{R\}$-prime implicate of $\exists R.(f(\phi_1 \wedge \phi_2))$, where f is the standard mapping between $\mathcal{K}$-formulae and $\mathcal{ALC}$ concepts. This is sufficient to show CoNEXPTIME-hardness since the conservative extension problem remains CoNEXPTIME-hard even when ϕ_1 is restricted to be a cubal formula.

We leave the determination of the exact complexity classes of the different prime implicate recognition tasks as an interesting open problem.

5 Future Work

While there are several possible continuations to this work, the question that interests us most is the appropriate extension of consequence finding to ABox assertions and TBox axioms. This is a non-trivial task as while the definition of clausal forms of assertions and axioms, and thus of assertion and axiom prime implicates, is rather straightforward, we lose some of the nice properties enjoyed by concept prime implicates. In particular, prime implicate axioms and assertions do not in general satisfy the covering property since there may be infinite sequences of increasingly more specific assertions or axioms. This is a familiar problem as these infinite sequences are also responsible for the inexistence of most specific concepts of ABox individuals in many common DLs (cf. [6]) and the lack of uniform interpolation for $\mathcal{ALC}$ TBoxes [7]. There appear to be two possible solutions to this problem. The simplest solution consists in bounding the depth of the assertions/axioms to be generated. A second more elegant possibility is to enrich the language by fixpoint constructs (cf. [8]) so that these infinite sequences of assertions/axioms might be finitely represented.

References

1. Baader, F., Küsters, R.: Non-standard inferences in description logics: The story so far. In Gabbay, D.M., Goncharov, S.S., Zakharyaschev, M., eds.: Mathematical Problems from Applied Logic I. Logics for the XXIst Century. Vol. 4 IMS. Springer-Verlag (2006) 1–75
2. Marquis, P.: Consequence Finding Algorithms. In: Handbook on Defeasible Reasoning and Uncertainty Management Systems. Volume 5. Kluwer (2000) 41–145
3. Bienvenu, M.: Prime implicates and prime implicants in modal logic. In: Proc. AAAI-07. (2007) To appear.
4. Giunchiglia, F., Sebastiani, R.: A sat-based decision procedure for ALC. In: Proc. KR'96. (1996) 304–314
5. Ghilardi, S., Lutz, C., Wolter, F., Zakharyaschev, M.: Conservative extensions in modal logic. In: Proc. AiML'06. (2006)
6. Küsters, R., Molitor, R.: Approximating most specific concepts in logics with existential restrictions. AI Communications (15) (2002) 47–59
7. Ghilardi, S., Lutz, C., Wolter, F.: Did I damage my ontology? A case for conservative extensions in description logics. In: Proc. KR'06. (2006)
8. de Giacomo, G., Lenzerini, M.: A uniform framework for concept descriptions in description logics. Journal of Artifical Intelligence Research **6** (1997) 87–110