Applications and Extensions of PTIME Description Logics with Functional Constraints

David Toman and Grant Weddell
D. R. Cheriton School of Computer Science, University of Waterloo, Canada

Abstract

We review and extend earlier work on the logic CFD, a description logic that allows terminological cycles with universal restrictions over functional roles. In particular, we consider the problem of reasoning about concept subsumption and the problem of computing certain answers for a family of attribute-connected conjunctive queries, showing that both problems are in PTIME. We then consider the effect on the complexity of these problems after adding a concept constructor that expresses concept union, or after adding a concept constructor for the bottom class. Finally, we show that adding both constructors makes both problems EXPTIME-complete.

1 Introduction

The need to provide better guarantees on performance has prompted a resurgence of interest in ontology languages for which various reasoning tasks have PTIME complexity. The latest version of the OWL, web ontology standard [W3C, 2008] reflects this trend by including so-called profiles. Notably, two of these are based on the description logics (DLs) EL++ [Baader et al., 2005] and DL-Lite [Calvanese et al., 2007], each of which has PTIME complexity for the fundamental task of deciding concept subsumption. In this paper, we consider a third DL with PTIME complexity for this problem, called CFD [Khizder et al., 2000], that can also serve as the basis for an OWL profile.

CFD is designed for ontologies that require an ability to express universal restrictions over functional roles, that have terminological cycles and that also require more general ways of expressing identification constraints such as composite keys and functional dependencies. This combination of features makes CFD another option to DL-Lite for integrating ontological knowledge about formatted data sources such as those conforming to underlying relational database schemata, a very common circumstance for many web applications. To illustrate, one can say the following about a hypothetical human resources data source in CFD:

∀x : EMP(x) → BOSS(Sup(x)),
∀x : EMP(x) → ∀y : EMP(y) →
(Ct(x) = Ct(y) ∧ Pn(x) = Pn(y)) → x = y, (1)
∀x : BOSS(x) → EMP(x) and
∀x : BOSS(x) → DIRECTOR(Sup(x)).

The statements assert that an employee is supervised by a boss and has a unique phone number within a given city, and also that a boss is an employee that is more specifically supervised by a director. In contrast, EL++ is not able to capture (abstract) functionality and functional restrictions such as keys, the first two statements for example, and DL-Lite is not able to capture additional conditions on attributes or roles that might be satisfied by subclasses, as in the case of the last statement about the supervision of bosses. However, both EL++ and DL-Lite are able to capture disjointness conditions which is not possible in CFD, as we show. Also note that Calvanese et al. [Calvanese et al., 2008] have recently extended DL-Lite with a path-based variety of identification constraints. Consequently, DL-Lite is now able to capture the second statement in (1).

We review and extend the earlier work on CFD, first considering the problem of reasoning about concept subsumption with respect to a CFD TBox, and then considering the problem of computing certain answers for a large subclass of conjunctive queries with respect to a CFD knowledge base consisting of both a TBox and ABox. In both cases, we consider the effect on the complexity of these problems with respect to the addition of concept constructors for expressing concept union and the bottom (unsatisfiable) class. In particular, in Sections 2 and 3, we show the following for CFD (with a TBox that may contain terminological cycles):

1. The problem of determining if a particular concept subsumption, called a posed question, is logically implied by the TBox is in PTIME. Our proof is based on a reduction of the problem to a Horn formulation over a finite universe, a much simpler approach than the procedure outline in [Khizder et al., 2000], and one that enables a more transparent consideration of extensions to CFD that follow.
2. This problem is coNP-complete if one adds a concept constructor that expresses concept union to CFD.
3. The problem is PSPACE-complete if, alternatively, one adds a concept for the bottom class.
4. The problem is ultimately EXPTIME-complete if one adds concept constructors for both concept union and the bottom class.

We also consider extending the expressiveness of posed questions to allow primitive negation, disjunction, general negation and inverse features, and show the effect of these extensions on the complexity of the implication problem, in particular when measured only in the size of the posed question.

We then consider the problem of computing certain answers to a family of attribute-connected conjunctive queries over an arbitrary CFD knowledge base. Such queries are more general than those that can be folded into a single concept description in more typical DLs and include, for example, queries with existential restrictions corresponding to foreign-key joins. We show how all of the above complexity bounds for the concept subsumption problem for CFD and its extensions transfer to this second problem.

The remainder of this section introduces CFD, in particular, the definitions related to the logical implication problem for inclusion dependencies that are needed for an understanding of the results in Sections 5 and 6. Additional definitions concerning ABox reasoning are left to the start of Section 4 in which we consider certain answer computation. We conclude in Section 5 and 6 with a discussion of related work and with summary comments.

1.1 The Description Logic CFD

A formal definition of the syntax and semantics of the logical implication problem of CFD follows. Note that CFD itself is based on attributes (also called features) instead of the more common case of roles (which are easily accommodated by reification [Toman and Weddell, 2005a]).

**Syntax**

\[
C ::= A \\
| C_1 \cap C_2 \\
| C \land C \\
| \forall \text{Pf}. C \\
| C : \text{Pf}_1, \ldots, \text{Pf}_k \rightarrow \text{Pf} \\
| \land_{i=1}^{k} (\text{Pf}_i)(x) = (\text{Pf}_i)(y) \\
| (\text{Pf}_1 = \text{Pf}_2)
\]

**Semantics:** “(·)\text{CFD}”

\[
D ::= C \\
| D_1 \cap D_2 \\
| D_1 \land D_2 \\
| \forall \text{Pf}. D \\
| C : \text{Pf}_1, \ldots, \text{Pf}_k \rightarrow \text{Pf} \quad \{ x \in (C) \} \\
| \land_{i=1}^{k} (\text{Pf}_i)(x) = (\text{Pf}_i)(y) \\
| (\text{Pf}_1 = \text{Pf}_2)
\]

\[
E ::= C \\
| \perp \\
| E_1 \land E_2 \\
| \forall \text{Pf}. E \\
| (\text{Pf}_1 = \text{Pf}_2) \\
| (x : (\text{Pf}_1)(x) = (\text{Pf}_2)(x))
\]

**Figure 1:** Syntax and Semantics of CFD.

**Definition 1 (the logic CFD)** Let F and A be disjoint sets of (names of) attributes and primitive concepts, respectively. A path expression Pf is a word in F* with the usual convention that the empty word is denoted by Id and concatenation by “·”. Concept descriptions are defined by the grammar on the left-hand-side of Figure 1. A concept produced by the “C : Pf_1, \ldots, Pf_k \rightarrow Pf” production of this grammar is called a path functional dependency (PFD). In addition, any occurrence of a PFD must adhere to one of the following two forms:

1. \( C : \text{Pf}_1, \ldots, \text{Pf}_i, \ldots, \text{Pf}_k \rightarrow \text{Pf} \) or
2. \( C : \text{Pf}_1, \ldots, \text{Pf}_i, \ldots, \text{Pf}_k \rightarrow \text{Pf}.f \) \hspace{1cm} (2)

An inclusion dependency C is an expression of the form \( C \subseteq D \). (Note the distinction of concept descriptions that can appear on the left- and right-hand sides of C.) A terminology (TBox) \( T \) consists of a finite set of inclusion dependencies. A posed question Q is an expression of the form \( E_1 \subseteq E_2 \).

The semantics of expressions is defined with respect to a structure \( (\Delta, \mathcal{T}) \), where \( \Delta \) is a domain of “objects” and \( (\cdot)\text{CFD} \) an interpretation function that fixes the interpretation of primitive concepts A to be subsets of \( \Delta \) and attributes f to be total functions \( (f)\text{CFD} : \Delta \rightarrow \Delta \). The interpretation is extended to path expressions by interpreting the empty word (Id) as the identity function \( \lambda x.x \), concatenation as function composition, and to derived concept descriptions C, D and E as defined on the right-hand-side of Figure 1.

An interpretation satisfies an inclusion dependency \( C \subseteq D \) (resp. a posed question \( E_1 \subseteq E_2 \)) if \( (C)\text{CFD} \subseteq (D)\text{CFD} \) (resp. \( (E_1)\text{CFD} \subseteq (E_2)\text{CFD} \)).

The logical implication problem asks if \( T \models Q \) holds; that is, for a posed question Q, if Q is satisfied by any interpretation that satisfies all inclusion dependencies in T.

The conditions imposed on PFDs in (2) distinguish, for example, PFDs of the form \( C : f \rightarrow \text{Id} \) and \( C : f \rightarrow g.h \) from PFDs of the form \( C : f \rightarrow g,f \). The strategic advantage of these conditions is that chase procedures can employ a simple saturation strategy for PFDs that will “fire” them unconditionally, a process that would otherwise not terminate. Indeed, we rely on this in presenting a PTIME procedure for the implication problem in the next section.

Note that the conditions are still satisfied by PFDs that correspond to arbitrary keys or functional dependencies in the sense of the relational model. Indeed, we have found that the conditions do not appear to hinder any modeling utility of CFD for ontologies that occur in practice. Moreover, relaxing the boundary conditions, such as allowing PFDs of the form \( C : f \rightarrow g.f \), already leads to undecidability [Toman and Weddell, 2006].

To illustrate, consider our introductory example relating to a hypothetical human resources data source. The statements in (1) can be captured in CFD as follows:

\[
\text{EMP} \subseteq \forall \text{Sup.Boss}.
\]

\[
\text{EMP} \subseteq \text{EMP} : \text{Ct}, \text{Pn} \rightarrow \text{Id},
\]

\[
\text{Boss} \subseteq \text{EMP} \text{ and } \text{Boss} \subseteq \forall \text{Sup.Director}.
\]

In particular, the i_th statement in (1) is captured by the i_th inclusion dependency in (3) above.

There are two final things to note about the grammar. First, in presenting our PTIME procedure in the next section and again when considering the addition of the bottom class to CFD in Section 3, we presume each inclusion dependency in a TBox \( T \) has the form \( A_1 \cap \cdots \cap A_k \subseteq D \) for \( A_i \) a primitive
concept. Second, although the grammar does not allow a PFD to appear directly in a posed question, the following lemma establishes that an indirect check for the logical implication of such concepts can be easily simulated.

**Lemma 2** Let $T$ be a $\text{CFD}$ terminology and $E_1 \subseteq E_2 : \text{Pf}_1, \ldots, \text{Pf}_k \rightarrow \text{Pf}$ be a posed question in an extension of $CFD$ (that allows such a construct in posed questions). Then there is a posed question $Q$ in $CFD$ such that $T \models E_1 \subseteq E_2 : \text{Pf}_1, \ldots, \text{Pf}_k \rightarrow \text{Pf}$ iff $T \models Q$.

Proof (sketch): Define $Q$ to be the subsumption

$$\forall f,E_1 \cap \forall g.E_2 \cap \left( \bigcap_{i=1}^{k} \left( (f. \text{Pf}_i = g. \text{Pf}_i) \right) \right) \subseteq (f. \text{Pf} = g. \text{Pf})$$

where $f$ and $g$ are attributes not occurring in $T$. □

2 The Polynomial Case

We first show that the logical implication problem for $\text{CFD}$ is in PTIME. Our proof is based on encoding a given problem as a collection of Horn clauses. The reduction introduces terms that correspond to path expressions, and relies on the fact that the number of required terms is polynomial in the size of the problem itself.

**Definition 3 (Expansion Rules)** Let $T$ and $Q$ be a $\text{CFD}$ terminology and a posed question, respectively. We write $\text{CON}(T, Q)$ to denote the set of all subconcepts appearing in $T$ and $Q$, define $\text{PF}(T, Q)$ as

$${\text{PF}}(\text{Pf}, \text{Pf}') \mid \text{Pf} \text{ is a prefix of a path expression in } Q \text{ and } \text{Pf}' \text{ is a path expression in } T \text{ or Id},$$

write $C_{\cdot}$ to denote unary predicates for $C \in \text{CON}(T, Q)$, and introduce the binary predicate $E$, with all predicates ranging over the universe $\text{PF}(T, Q)$. The expansion rules for a given terminology $T$, denoted $R(T)$, are defined in Figure 2.

![Figure 2: Expansion Rules.](image)

A goal for each concept $E$ is a set of ground assertions de-

**Theorem 4** Let $T$ be a $\text{CFD}$ terminology and $Q$ a posed question of the form $E_1 \subseteq E_2$. Then $T \models Q$ iff $R(T) \cup \{C_{E_1}(Id)\} \models G_{E_2}$ or $R(T) \cup \{C_{E_1}(Id)\} \models C_{\cdot}(\text{Pf})$ for $\text{Pf} \in \text{PF}(T, Q)$.

Proof (sketch): The least model of $R(T) \cup \{C_{E_1}(Id)\}$, if one exists, can be extended to a model of $T$. Since the original model is the least model, it suffices to determine whether $E_2$ subsumes $E_1$ (in the extension of this model).

Since the expansion rules are Horn clauses over a finite universe $\text{PF}(T, Q)$ of polynomial size, we have the following:

**Corollary 5** Let $T$ and $Q$ denote a terminology and posed question in $\text{CFD}$. Then the implication problem $T \models Q$ is decidable in PTIME.

Proof (sketch): The least model of $R(T) \cup \{C_{E_1}(Id)\}$ can be obtained by using a bottom-up construction of the least fix-point of the rules in time polynomial in $|T| + |Q|$. □

In practice, elements of this set can be constructed on demand by using additional Horn rules in such a way that only path expressions needed to confirm subsumption or non-subsumption are generated.

3 Intractable Extensions

We now consider the consequences of the various extensions to $\text{CFD}$ outlined in our introductory comments. The extensions add options for the right-hand-sides of inclusion dependencies and are illustrated in Figure 3. We begin by considering the case for concept union. Subsection 3.2 then considers an alternative addition of a constructor for the bottom class, and Subsection 3.3 considers the addition of both constructors. Finally, in Subsection 3.4, we consider consequences of various extensions to a posed question.
To handle disjunction in right-hand sides of inclusion dependencies, we extend the expansion rules with “or” rules as follows:

\[
R^{\cup}(T) := R(T) \cup \{C_{C_1 \sqcup C_2}({\sf Pf}) \rightarrow C_{C_1}({\sf Pf}) \text{ or } C_{C_2}({\sf Pf})\}.
\]

This rule is added to the expansion rules used for the polynomial case. The intuition behind using this rule is to non-deterministically find a model with an object that belongs to the left-hand side of the posed question but which does not belong to the right-hand side.

**Theorem 6** Let \( T \) be a \( CFD^\cup \) terminology and \( Q \) a posed question of the form \( E_1 \sqsubseteq E_2 \). Then

\[ T \models Q \text{ iff } G_{E_2} \subseteq M \text{ or } C_{\bot}({\sf Pf}) \in M \text{ for Pf} \in PF(T, Q) \text{ in each minimal model } M \text{ of } R^{\cup}(T) \cup \{C_{E_1}({\sf Id})\}.
\]

Proof (sketch): The proof is similar to the proof of Theorem 4 augmented with a non-deterministic application of the above rule. Thus non-subsumption can be detected using polynomially many non-deterministic guesses.

**Corollary 7** Let \( T \) and \( Q \) denote a terminology and posed question in \( CFD^\cup \). Then the implication problem \( T \models Q \) is decidable and coNP-complete.

Proof (sketch): To obtain the upper bound, simply apply rules of the form “\( C_{C_1 \sqcup C_2}({\sf Pf}) \rightarrow C_{C_1}({\sf Pf}) \text{ or } C_{C_2}({\sf Pf})\)” nondeterministically during the construction of a minimal (but not necessarily unique) model of \( R^{\cup}(T) \cup \{C_{E_1}({\sf Id})\} \) that yields an counterexample to \( T \models Q \); a construction similar to Theorem 4. Non-implication can be detected in a polynomial number of guesses (is in NP). To obtain the lower bound, we reduce 2+2-SAT to non-implication \( T \not\models Q \) in \( CFD^\cup \). □

### 3.2 The Disjointness Case: \( CFD^\sqcap \)

We now show that the logical implication problem for \( CFD \) becomes PSPACE-complete if the bottom class is allowed in inclusion dependencies. This is also accomplished by appealing to our construction in Section 2, expanding in particular on the issue of “completing” the construction of counterexample interpretations.

**Definition 8 (Satisfiable Atomic Type)** Let \( T \) denote a finite set of primitive concepts. An atomic type over \( T \) is the concept \( \forall A \in T. A \). We say that \( T \) is satisfiable with respect to a \( CFD^\sqcap \) terminology \( T \) if there is an interpretation \( I_T \) such that \( I_T \models T \) and \( \forall A \in T. A \not\models \emptyset \).

**Lemma 9** Let \( T \) be a \( CFD^\sqcap \) terminology and \( T \) a finite set of primitive concepts. The problem of determining if \( T \) is satisfiable with respect to \( T \) is decidable and in \( \text{PSPACE} \).

Note that it is easy to construct terminologies (using only \( \sqcap \) and \( \forall . \)) in which a contradiction is found only after exponentially many steps (of \( f \)). We use a reachability search algorithm to detect these situations. Since the graph of all atomic types \( T \) is exponential in the size of the terminology, the reachability of an unsatisfiable atomic type starting from \( T \) can be determined in \( \text{PSPACE} \).

**Theorem 10** Let \( T \) be a \( CFD^\sqcap \) terminology and \( Q \) a posed question of the form \( E_1 \sqsubseteq E_2 \). Then \( T \models Q \iff \)

1. \( C_{\bot}({\sf Pf}) \in M \) for some Pf \( \in PF(T, Q) \), or
2. the set \( \{A \in \text{CON}(T, Q) \mid C_{A}({\sf Pf}) \in M, A \text{ primitive}\} \) is unsatisfiable w.r.t. \( T \) for some Pf \( \in PF(T, Q) \), or
3. \( G_{E_2} \subseteq M \)

holds in the minimal model \( M \) of \( R(T) \cup \{C_{E_1}({\sf Id})\} \).

Proof (sketch): The construction of a counterexample follows the construction in the proof of Theorem 4. However, the completion of the interpretation by infinite trees now depends on the existence of such trees and must be checked using Lemma 9. Note that this also relies on the fact that any interpretation \( I_T \) that exists for an atomic type \( T \) can be made into a tree interpretation.

**Corollary 11** Let \( T \) and \( Q \) denote a terminology and posed question in \( CFD^\sqcap \). Then the implication problem \( T \models Q \) is decidable and \( \text{PSPACE-complete} \).

Proof (sketch): To show hardness for \( \text{PSPACE} \), we reduce the (deterministic) Linear Bounded Automaton Acceptance [Garey and Johnson, 1979; Karp, 1972] to the implication problem in \( CFD^\sqcap \).

### 3.3 The Boolean Complete Case: \( CFD^{\sqcap,\sqcup} \)

We now show that adding both disjointness and coverage constraints leads to \( \text{EXPTIME} \) completeness. In particular, we show how to simulate axioms of the form

\[
\forall f_1.A_1 \cap \forall f_2.A_2 \sqsubseteq \forall f_3.A_3.
\]

(4)

Consider the following auxiliary inclusion dependencies relating to primitive concepts occurring in a logical implication problem for \( CFD \):

- Define \( T \) as a top-replacement concept and pairs of \( A_i \) and \( \bar{A}_i \) as disjoint partitions of the simulated top (and similarly for \( B \) and \( B \)):
  \[
  A_i \cap \bar{A}_i \subseteq \bot, \quad T \subseteq A_i \cup \bar{A}_i, \quad A_i \subseteq T, \quad \bar{A}_i \subseteq T.
  \]
- Propagate the top-replacement concept across all attributes:
  \[
  T \subseteq \forall f.T \text{ for all } f \in F.
  \]

Then the pair of inclusion dependencies

\[
B \subseteq \forall f_1.A_1 \cap \forall f_2.A_2 \text{ and } B \subseteq \forall f_3.A_3
\]

simulates the dependency (4) above, and, as a consequence, the decision problem is \( \text{EXPTIME}-\text{hard} \) [Toman and Weddell, 2005a]. To obtain a matching upper bound, we note that \( CFD^{\sqcap,\sqcup} \) is a fragment of \( DCFDE^- \) that is \( \text{EXPTIME}-\text{complete} \) [Khizder et al., 2007]. The latter logic is \( \text{Boolean-complete} \) and thus might be more convenient to use.
3.4 Extensions to the Posed Questions

In this final subsection, we consider several ways that the structure of posed questions for a CFD implication problem might be extended.

**Primitive Inequalities and Negations: \( \text{CFD}_{\neq, (\neg)} \)**

Extending the grammar for the posed questions with primitive negations, i.e., negated primitive concepts, \( \neg A \), and primitive inequalities, \( \langle \text{Pf}_1 \neq \text{Pf}_2 \rangle \), still enjoys a PTIME complexity bound for the logical implication problem.

**Theorem 12** Let \( T \) be a \( \text{CFD} \) terminology and \( Q = E_1 \subseteq E_2 \) that allows primitive negations and/or inequalities. Then \( T \models Q \) is decidable in PTIME.

**Proof (sketch):** We replace \( Q \) with a (linearly-sized) set of questions that do not use negations or inequalities. The result then follows from Theorem 4. \( \square \)

Similar PTIME results can be obtained for \( \text{CFD}_{\neq, \cup} \), i.e., when terminologies do not use disjunction and only when complexity is measured solely in \(|Q|\), i.e., under the data complexity assumption. For the extensions that allow disjunctions in the terminology, the data complexity increases to \( \Pi^p_2 \): the complexity results for \( \text{CFD}_{\neq} \) terminologies is also \( \Pi^p_2 \)-complete (for the same reasons). For terminologies that allow the \( \cup \) constructor, the complexity increases to \( \Pi^p_2 \) since matching must be tested in all minimal models of \( T \) and \( E_1 \).

**Theorem 13** Let \( T \) be a \( \text{CFD}_{\neq, \cup} \) terminology and \( Q = E_1 \subseteq E_2 \) that allows primitive negations and/or inequalities. Then \( T \models Q \) is coNP-complete.

**Proof (sketch):** We encode a propositional DNF formula using the concept \( E_1 \), force a valuation using disjunction in \( T \), encode evaluation of the boolean formula (also in \( T \)), and then test if every such valuation evaluates to true using \( E_2 \). \( \square \)

**Primitive Inequalities and Disjunction: \( \text{CFD}_{\neq, \cup} \)**

Adding disjunctions to posed questions with primitive negations and/or allowing general negations in posed questions leads to an increase in data complexity.

**Theorem 14** Let \( T \) be a \( \text{CFD} \) terminology and \( Q = E_1 \subseteq E_2 \) with disjunctions and inequalities. Then \( T \models Q \) is decidable and coNP-complete.

The coNP-completeness result can be lifted to the intractable extensions of \( \text{CFD}_{\neq} \) under the data complexity assumption.

**Inverse Features: \( \text{CFD}_{\text{inv}} \)**

Last, we consider adding the inverse feature constructor, \( \exists f^{-1}.E \), to the grammar for posed questions\(^1\). Intuitively, such an extension allows for enforcing and checking for existence of additional objects that are not reachable from the objects interpreting \( E_1 \) and \( E_2 \) in the posed questions.

To accommodate the new constructor, we extend the set \( \text{Pf}(T, Q) \) to accommodate the new paths that are induced by the new constructor; the extended set is defined as

\[
\{ \text{Pf}(o) \mid \text{Pf} \in \text{Pf}(T, Q) \} \cup \{ f_1, \text{Pf}(a_i) \mid \text{Pf} \in \text{Pf}(T, Q), \exists f_i^{-1}.E \text{ appears in } E_1 \cup E_2 \},
\]

where \( o \) and \( a_i \) are distinct constants. Intuitively, we create a new constant \( a_i \) for each instance of \( \exists f^{-1}.D \) constructor in the posed question. This set is still polynomial in the size of the original problem. Note also that the expansion rules naturally apply to this new set. Hence we can use similar techniques to solve the logical implication problem: we simply extend the definition of \( G_E \) to accommodate the new constructor by introducing the new constants \( a_i \) and by using equations to ensure their \( f_i \) attributes agree with the existing paths as required. However, note that allowing the \( \exists f^{-1}.D \) constructor in the \( E_2 \) concept of the posed question leads to an increase of complexity: to verify that \( E_2 \) is entailed by \( T \) and \( E_1 \), a subgraph isomorphism problem must be solved (one needs to match the new constants in \( E_2 \) to those that originated from \( E_1 \)).

**Theorem 15** Let \( T \) be a \( \text{CFD} \) terminology and \( Q = E_1 \subseteq E_2 \) with inverse attributes. Then \( T \models Q \) is decidable and \( \Pi^p_2 \)-complete.

The data complexity for \( \text{CFD}_{\neq} \) terminologies is also \( \Pi^p_2 \)-complete (for the same reasons). For terminologies that allow the \( \cup \) constructor, the complexity increases to \( \Pi^p_2 \) since matching must be tested in all minimal models of \( T \) and \( E_1 \).

The complexity results for \( \text{CFD} \) and its extensions are summarized in Figure 4: the superscripts denote the extensions allowed in the terminology \( T \) and the subscripts the extensions in the posed questions \( Q \).

4 Query Answering

In this section, we consider the problem of computing certain answers to a family of conjunctive queries over an arbitrary CFD knowledge base that augments a CFD TBox with an ABox.

**Definition 16 (The ABox consistency problem for CFD)** Let \( N \) be a set of (names of) individuals disjoint from \( F \) and \( A \). An ABox \( A \) consists of a finite set of assertions of the form \( C(a) \) or \( f(a) = b \), where \( C \) is a concept description, \( f \in F \) and \( \{a, b\} \subseteq N \).

An interpretation satisfies an ABox assertion \( C(a) \) (resp. \( f(a) = b \)) if \( (a)^2 \in (C)^2 \) (resp. \( (f)^2((a)^2) = (b)^2 \)).

The ABox consistency problem asks if \( T \cup A \) is consistent, that is, if there exists an interpretation that satisfies all inclusion dependencies in \( T \) and all assertions in \( A \).

We write Pf\((a) = b \) as shorthand for the equivalent set of primitive ABox assertions with “single use” intermediate individuals.

**Proposition 17** Let \( T \) be a \( \text{CFD}_{\text{inv}, \cup} \) terminology. Then

1. for every ABox \( A \) there is a concept \( E \) such that \( T \cup A \) is not consistent if and only if \( T \models E \sqsubseteq \bot \); and
2. for every equational concept \( E \) there is an ABox \( A \) such that \( T \models E \sqsubseteq \bot \) if and only if \( T \cup A \) is not consistent.

This proposition is a special case of a result for \( D_{\text{CFD}}^{\text{inv}, \cup} \) [Khizder et al., 2007]; we also use a similar representation of the ABox in the proof of Theorem 19 below. Note that, for \( \text{CFD} \) and \( \text{CFD}_{\text{inv}} \), inconsistency cannot be derived. The expansion rules from Definition 3, however, still allow us to construct (a representation) of a minimal model(s) of the TBox and ABox that we shall use to show whether or not a particular tuple of ABox individuals is a certain answer to a query with respect to the given TBox and ABox.
Conjunctive Queries.
The equational constructs in CFD posed questions allow us to characterize certain conjunctive queries as equational concepts. This provides a straightforward approach to query answering over ABoxes. Unfortunately, the equational construct is not powerful enough to capture all conjunctive queries.

**Definition 18** A conjunctive query \( Q \) is an expression

\[
q(x_{i_1}, \ldots, x_{i_k}) \leftarrow \bigwedge C_i(x_i) \land \bigwedge f_i(x_i) = x_j
\]

where \( C_i \) and \( f_i \) are CFD concepts and attributes, respectively. We call the variables \( x_{i_1}, \ldots, x_{i_k} \) distinguished.

A query graph for a conjunctive query \( Q \) is a directed graph with variables \( x_i \) of \( Q \) playing the role of nodes that are connected by (directed) edges \( (x_i, x_j) \) whenever \( f_i(x_i) = x_j \) is an atom in the query \( Q \).

We say that a conjunctive query \( Q \) is attribute-connected (resp. connected) if every variable in the query graph of \( Q \) is reachable from a distinguished variable along a directed (resp. undirected) path.

Although attribute-connected conjunctive queries disallow some forms of arbitrary conjunctive queries, they do admit non tree-shaped queries that cannot be folded into concept descriptions in common description logics.

We assume a standard definition of query answering under constraints utilizing the certain answer semantics, i.e., the ground instantiation of the head of the query is entailed by the underlying theory \( T \) extended with the query \( Q \) considered to be a universally quantified implication.

**Theorem 19** Let \( T \) be a CFD terminology, \( A \) an ABox, \( Q \) a attribute-connected conjunctive query, and \( \theta \) a substitution \( \langle a_1/x_{i_1}, \ldots, a_k/x_{i_k} \rangle \) for \( a_1, \ldots, a_k \) ABox objects. Then there are equational concepts \( E_A, E_{\theta}, \) and \( E_Q \) such that

\[
T \cup A \models Q\theta \iff T \models E_A \land E_{\theta} \equiv E_Q.
\]

In addition, \( E_A, E_{\theta}, \) and \( E_Q \) can be constructed from \( A \) and \( Q \) in polynomial time.

Proof (sketch): Let \( A \) be an ABox expressing assertions about the set of individuals \( \{a_1, \ldots, a_l\} \). We associate an equational concept \( E_A \) as follows:

\[
E_A = \left( \bigwedge_{C(a_i) \in A} \forall h_i.C \right) \cap \left( \bigwedge_{j=1}^{l} \left( f(a_j) = a_i \in A, h_i.f = h_j \right) \right)
\]

where the attributes \( h_1, \ldots, h_l \) do not occur in \( T \cup A \) nor in \( Q \). Intuitively, these attributes represent the individual ABox objects.

For an attribute-connected query \( Q \), let \( Pf_f^j \) be a path that leads from a distinguished query variable \( x_i \) to the variable \( x_j \) in the query graph of \( Q \). At least one such path that starts from some distinguished variable \( x_i \) must exist for every variable \( x_j \) \((0 < j \leq n) \) in \( Q \) (for the distinguished variables, the path can play the role of nodes that are connected by (directed) edges whenever \( f_i(x_i) = x_j \) is an atom in the query \( Q \)).

We define the concept \( E_{\theta} \) to capture the substitution \( \theta = \langle a_1/x_{i_1}, \ldots, a_k/x_{i_k} \rangle \) as follows:

\[
E_{\theta} = \bigwedge_{0 \leq j \leq n} \left( h_i. Pf_f^j = x_j \right).
\]

To determine whether \( Q\theta \) is a certain answer to \( Q \) (under \( T \cup A \)) we create an equational concept \( E_Q \) as follows:

\[
E_Q = \left( \bigwedge_{C(x_i) \in Q} \forall x_i.C \right) \cap \left( \bigwedge_{f(x_i) = x_j \in Q} \left( x_i.f = x_j \right) \right)
\]

It is now easy to show that the tuple \( \langle (h_1)^T(o), \ldots, (h_k)^T(o) \rangle \) can serve as certain answer to \( Q \) whenever \( o \in (E_Q)^T \) and vice versa. \( \square \)

This result allows transferring complexity bounds derived in Sections 2 and 3 for the implication problem in CFD and its extensions to answering of path-connected conjunctive queries over an ABox under the certain answer semantics.

A similar approach can be used to determine substitution for more general connected queries. However, the restriction to at least connected queries is essential to maintain the low complexity:

**Theorem 20** The problem of finding certain answers over CFD terminologies and ABoxes is PSPACE-complete even for queries of the form \( \exists x.C(x) \) for \( C \) a primitive concept.

Proof (sketch): Follows directly from the CFD\(^{\perp} \) case. \( \square \)

The restriction is also needed to maintain decidability in the presence of inequalities in queries [Calvanese et al., 1998].

5 Related Work

It has been shown that removing the boundary conditions imposed on right-hand-sides of PFDs in CFD makes its implication problem EXPTIME-complete [Khizder et al., 2001], and leads to undecidability of both the implication problems and ABox consistency problems for CFD\(^{\perp,\perp} \) [Khizder et al., 2007; Toman and Weddell, 2006]. It is also easy to see that allowing path agreements in terminologies makes the implication problem for CFD undecidable (by virtue of a straightforward reduction of the uniform word problem [Machtey and Young, 1978]). And it is interesting that: 1) the following two restricted cases have decidable decision problems:

- allowing arbitrary PFDs in terminologies, or
- allowing path agreements in the posed question;

but that 2) the combination of these two cases leads to undecidability [Khizder et al., 2007].

PFDs were first introduced and studied in the context of object-oriented data models [Ito and Weddell, 1994; Weddell, 1989]. Subsequently, an FD concept constructor was proposed and incorporated in Classic [Borgida and Weddell, 1997], an early DL with a PTIME reasoning procedure, without changing the complexity of its implication problem.
generalization of this constructor to unrestricted PFDs alone leads to \textsc{EXPTIME} completeness of the implication problem [Khizder et al., 2001], a complexity that remains unchanged in the presence of additional concept constructors common in rich DLs such as roles, qualified number restrictions, and so on [Toman and Weddell, 2001; 2004]. PFDs have also been applied to a number of applications in object-oriented schema diagnosis and synthesis [Biskup and Polle, 2000; 2003], in query optimization [DeHaan et al., 2003: Khizder et al., 2000] and in the selection of indexing for a database [Stanchev and Weddell, 2003]. Finally, we note that $\mathcal{C}\mathcal{F}D$ compliments $\mathcal{E}\mathcal{L}++$ [Baader et al., 2005] and DL-Lite [Calvanese et al., 2007] as an additional DL-based option for \textsc{PTIME} OWL profiles.

6 Summary

We have reviewed and extended earlier work on $\mathcal{C}\mathcal{F}D$, a dialect of description logic with universal restrictions over functional roles that allows terminological cycles in a given TBox. Notably, $\mathcal{C}\mathcal{F}D$ includes a concept constructor that can capture a general form of functional constraint that can in turn be used to express, among other things, TBox constraints for capturing knowledge that relates to object identification. $\mathcal{C}\mathcal{F}D$ complements existing dialects such as $\mathcal{E}\mathcal{L}++$ and DL-Lite with \textsc{PTIME} reasoning procedures. In particular, we have shown that both the concept implication problem and the problem of computing certain answers for a $\mathcal{C}\mathcal{F}D$ knowledge base for an important family of conjunctive queries have \textsc{PTIME} procedures. We have also considered the effect on the complexity of these problems following the addition of concept constructors for expressing concept union and the bottom class, showing for each possible extension that the complexity becomes progressively more intractable. Finally, we considered extending the expressiveness of posed questions to allow primitive negation, disjunction, general negation and inverse features, and have shown the effect of these extensions on the complexity of the associated implication problems when measured in the size of the posed question.

References


