セマンティックWebオントロジ記述言語OWLによる関数従属性の表現について

中村　大介† 　岩井原瑞穂†† 　上林弥彦 *††

† 京都大学工学部情報学科　〒 606-8501 京都市左京区吉田本町
†† 京都大学大学院情報学研究科　〒 606-8501 京都市左京区吉田本町
E-mail: †daisuke@db.soc.i.kyoto-u.ac.jp, ††{iwaihara, yahiko}@i.kyoto-u.ac.jp

あらまし　OWL（Web Ontology Language）はセマンティック Web におけるオントロジ記述言語である。OWL は記述論理をベースにしているため、高度なモデル化機能を有し、スキーマやそれらの一貫性制約を表現することができ、同値性、無矛盾性、冗長性、充足可能性の検査・検証を行うことができる。しかし OWL を用いてどのくらいまでデータベースの複雑な意味制約を表現することができるかは明確となっていない。そこで本稿では関係モデルにおける関数従属性に着目し、OWL を用いてどのくらい複雑な関数従属性を表現できるかを議論する。さらに OWL を関数従属性が扱えるように拡張した OWL_{fd} を定義し、それの推論の計算量についても議論する。関数従属性を OWL で表現しておくと従属性の検査を行うことができ、異種の関係スキーマを統合する際に有用である。
キーワード　情報統合、セマンティック Web、OWL、記述論理、関数従属性

Representing Functional Dependencies by the OWL Web Ontology Language

Daisuke NAKAMURA†, Mizuho IWAIHARA††, and Yahiko KAMBAYASHI††

† School of Infomatics, Faculty of Engineering, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501 JAPAN
†† Department of Social Infomatics Graduate School of Informatics, Kyoto University
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501 JAPAN
E-mail: †daisuke@db.soc.i.kyoto-u.ac.jp, ††{iwaihara, yahiko}@i.kyoto-u.ac.jp

Abstract　The OWL Web Ontology Language is developed for describing ontologies of Semantic Web. Since OWL is based on description logics, it can model schemata and their integrity constraints, and examine equivalence, consistency, redundancy, and satisfiability. However, it is not clear how much complex constraints on a database OWL can express. In this paper we discuss how functional dependencies of the relational model can be represented in OWL. We extend OWL with functional dependencies, and also discuss complexity of reasoning in the resulting OWL, called OWL_{fd}. Encoding functional dependencies by OWL is useful for integrating heterogeneous schemata because we can examine integrity constraints and dependencies.
Key words　Information Integration, Semantic Web, OWL, Description Logics, Functional Dependency

1. はじめに

現在インターネット上には膨大な量の情報が存在するため、その中から有用な情報を得ることは非常に困難になってきている。またキーワードの組み合わせだけでは探すことのできない情報も存在する。そのような問題に対して近年注目されている技術がセマンティック Web である。セマンティック Web ではインターネット上の情報に対して明示的な意味を与え、機械がその情報を自動で処理し、統合して用いたりすることでユーザに有益な情報やサービスを提供することを目的としている。それを実現するための中心技術がオントロジ記述言語 OWL である。OWL は 2002 年 3 月に最初の提案がなされ、現在 W3C において推奨仕様が公開されている。OWL にはインターネット上の情報の語彙や語彙間の関連を記述することでメタ情報を利用した精密な検索、XML 文書の容易な交換、大量の情報の中での自動的な制約検査、インターネット上のサービスの自動検索と

---

*2004 年 2 月 6 日逝去

いったことが期待されており、これを実現するための論理機構が提供されている。将来的にはより表現力の高い論理機構の導入や推論の信頼性を構築できる能力の追加などが考えられている。この OWL を用いることでインターネット上に分散している情報を統合して効率的に用いることができる。このとき分散したデータを統合するため、データ間に矛盾が生じる場合がある。そのため OWL はデータ間に矛盾が生じた場合に自動で検出することができる機能を有している。

これは Web 文書だけに限らず、電子商取引において企業間でやり取りされる XML 文書や、データベースの統合といった場合にも応用できる。各企業または各システムで独自にスキーマが定義されているため、実際は同じデータを表現していても異なるように見えたりする。これに対しオントロジを用いて各システムで用いられている表現や語彙を関係付けることで統合を実現できる。同様に異種のシステムを統合した場合、データの一貫性が保たれないという場合が生じる。関係データベースを統合した場合を考えると、それまで保たれていた一貫性制約（参照一貫性やキー制約や従属性）が満たされないという場合が生じる。これらの制約を OWL で記述しておけば統合の際に矛盾が生じた場合は自動で検出することができる。しかし OWL はどのくらい複雑な一貫性制約を記述できるかは明確でない。

OWL がこのような情報の統合や矛盾の検出を行えるのは、明確な意味論が定義された記述論理という言語を基礎においているためである。記述論理は構造化された情報を扱う一階述語論理の部分集合である。記述論理は明確な意味論が定義されているため、概念の充足可能性や概念間の包摂関係などを形式的な手法で判定することができる。さらに選言（$\sqcup$）や否定（$\neg$）を自由に使えるという高い表現力を持ちながら、それらの推論が決定可能であるという特徴を持つ。これをデータ管理の点で考えると、データモデルの設計、管理、データの検索、情報統合などに推論機構を導入して同値性、無矛盾性、冗長性、充足可能性の検査ができる。[3] では $\mathcal{ALUNI}$ というクラスの記述論理を用いてフレームモデル、実体関連モデル、オブジェクト指向モデルをモデル化している。[1] ではデータ管理における記述論理の有用性について述べられており、[3] と同様に実体関連モデルのモデル化について述べられている。[4] [7] では $\mathcal{DLR}$ というクラスの記述論理を用いて関係データベースにおける一貫性制約をモデル化している。そこではスキーマの一貫性や冗長性、実体関連モデルのエンティティや関係の包摂関係の判定について述べられており、クエリー言語としての記述論理についても議論されている。$\mathcal{DLR}$ の推論は ExpTime で決定可能であることも述べられている。しかし $\mathcal{DLR}$ はキーや関数従属性に関しては表現力が低く、単一属性のキーは表現できるが、複数属性のキーは表現できないという問題があった。そこで [5] [6] では推論の計算量を ExpTime 完全に保ったまま、キー属性や関数従属性を表現できるように $\mathcal{DLR}$ を拡張している。[6] は左辺が単一属性の関数従属性を含んでいないときに限り、推論が ExpTime 完全で決定可能であると述べている。

OWL はこれらの記述論理とは表現力が異なるためにどのくらいまで複雑なデータベースの一貫性制約を表現することができるか明確となっていない。そこで本稿では関係モデルにおける関数従属性に着目し、OWL を用いてどのくらいまで複雑な関数従属性を表現することができるかについて議論する。さらに OWL を関数従属性が扱えるように拡張した OWL_{fd} を定義し、それの推論の計算量についても議論する。これにより情報統合の際により正確な一貫性制約の検査を行い、矛盾を自動で検出できるようになる。

2. では関数従属性について簡単に説明し、3. ではオントロジ記述言語 OWL の概要を述べる。それを用いて 4. で関数従属性をどのくらいまで表現できるかについて議論する。5. では OWL を関数従属性を扱えるように拡張した OWL_{fd} の表現力や計算量について議論する。

2. 関数従属性

ここでは、関係データベースにおける関数従属性についてまとめる。関係データベースでは各種整合性制約を記述するための手段として、従属性の概念が用いられる。その最も基本的なものが関数従属性であり、以下のような形をしている。

関数従属性

関係 $R(\cdots, X, \cdots, Y, \cdots)$（$X$ と Y は属性集合）が与えられたとき、その任意のインスタンス中の任意の 2 つのタプルに関して、その X の値が等しいならば Y の値も必ず等しいという制約を関数従属性といい、$X \to Y$ と書く。

直感的には $X \to Y$ は属性 X の値が決まると、属性 Y の値が一意に決まることを意味している。関数従属性はデータベースを設計する際に重要な情報となり、これを満たさない場合は制約違反として検出される。以降この関数従属性を OWL を用いて表現することになる。

3. オントロジ記述言語 OWL

OWL（Web Ontology Language）記述論理を基礎においたオントロジ記述言語で、共有性、発展性、相互運用性、矛盾の検出といった機能を持っている。記述論理は表現力や計算量が形式的な手法で研究されていて、記述論理を基礎におくことでその意味論と推論方法が保証されている。3.1 において基礎となっている記述論理について、3.2 においてサブセットである OWL Lite と OWL DL について説明する。

3.1 記 述 論 理

3.1.1 記述論理の概要

記述論理（Description Logics）は概念階層や構造的な知識表現を扱う論理体系であり、変数や関数のない第 1 階述語論理の部分集合である。第 1 階述語論理と違い、充足判定問題が決定可能である。記述論理では構造化された知識を概念（concept）とロール（role）で表し、それぞれ述語論理では単項述語、2 項述語に対応する。基本概念（atomic concept）によって記号が与えられ、提供されているコンストラクタを用いて複合概念（complex concept）が定義される。具体的には論理結合子: $\sqcap$（連言）、$\sqcup$（選言）、$\neg$（否定）および量化子: $\forall$（全称記号）、$\exists$（存在記号）などがある。

例えば $\text{Person} \sqcap \forall \text{child.Male}$ は概念 Person のインスタンス

コンストラクタ名		構文	意味論
概念名	$\mathcal{AL}$	A	$A^I \subseteqq \Delta^I$
トップ	$\mathcal{AL}$	$\top$	Δ^I
ボトム	$\mathcal{AL}$	$\bot$	ϕ
連言	$\mathcal{AL}$	$C \sqcap D$	$C^I \cap D^I$
全称限量	$\mathcal{AL}$	$\forall R.C$	$\{a \mid \forall b.(a,b) \in R^I \rightarrow b \in C^I\}$
存在限量	$\mathcal{E}$	$\exists R.C$	$\{a \mid \exists b.(a,b) \in R^I \wedge b \in C^I\}$
選言	$\mathcal{U}$	$C \sqcup D$	$C^I \cup D^I$
否定	$\mathcal{C}$	$\neg C$	$\Delta^I \setminus C^I$
数値制約	$\mathcal{N}$	$\geqslant nR$	$\{a \mid \mid \{b \mid (a,b) \in R^I\} \mid \geqq n\}$
		$\leqslant nR$	$\{a \mid \mid \{b \mid (a,b) \in R^I\} \mid \leqq n\}$
数値限量制約	$\mathcal{Q}$	$\geqslant nR.C$	$\{a \mid \mid \{b \mid (a,b) \in R^I \wedge b \in C^I\} \mid \geqq n\}$
		$\leqslant nR.C$	$\{a \mid \mid \{b \mid (a,b) \in R^I \wedge b \in C^I\} \mid \leqq n\}$
nominal	$\mathcal{O}$	$\{a_1, \cdots, a_n\}$	$\{{a_1}^I, \cdots, {a_n}^I\}$
ロール名		P	$P^I \subseteqq \Delta^I \times \Delta^I$
逆	$\mathcal{I}$	R^-	$\{(a,b) \mid (b,a) \in R^I\}$
連言	$\mathcal{R}$	$Q \sqcap R$	$Q^I \cap R^I$
結合	$\mathcal{R}$	$Q \circ R$	$\{(a,b) \mid \exists c.(a,c) \in Q^I, (c,b) \in R^I\}$

表 1　DL のコンストラクタ [12]

でロール child を通じて概念 Male にだけ関連付けられている個体（individuals）の集合、つまり子供を持つ場合はすべて男の子であるような人を表している。それに対し ∃child はロール child を通じてある個体に関連付けられているようなすべての個体、つまり子供を持っている個体の集合を表している。またその他に数量限定によって、ロール R によるリンク先の個体数を制限できる。例えば $\geqslant$ 2child は、子供が 2 人以上いる個体の集合を表している。記述論理はこれらのコンストラクタを用い、概念を構築していく。表 1 は記述論理で用いられる主なコンストラクタとその意味論を示している。このとき I は解釈といい、対象領域 Δ^I と解釈関数 $\cdot^I$ との対 $(\Delta^I, \cdot^I)$ である。解釈関数 $\cdot^I$ はすべての概念名に対して Δ^I の部分集合を割り当て、すべてのロール名に $\Delta^I \times \Delta^I$ を割り当てる。

記述論理の知識ベースは T-Box と A-Box から構成される。C, D を概念としたとき、T-Box は $C \equiv D$ または $C \sqsubseteq D$ という形をした言明の集合である。$C \sqsubseteq D$ は必要条件だけの言明である。概念名を A で表したとき、言明 $A \equiv C$ を概念定義（concept definition）と呼び、言明 $A \sqsubseteq C$ をプリミティブな概念包含（primitive concept inclusion）と呼ぶ。それに対して、言明 $C \equiv D$ を概念等式（concept equation）と呼び、言明 $C \sqsubseteq D$ を概念包含（concept inclusion）と呼ぶ。A-Box は $C(a), R(a,b)$ という形をした概念とロールに関する言明の集合である。ただし a, b は個体とする。$C(a)$ は個体 a が概念 C の要素であることを宣言している。$R(a,b)$ は個体 a, b の対 (a,b) がロール R の要素であることを宣言している。T-Box と A-Box の充足可能性は、次のように定義される。解釈 I が与えられたとき、T-Box の言明 $C \equiv D (C \sqsubseteq D)$ に対して、${C^I} = {D^I} ({C^I} \subseteqq {D^I})$ であるとき、I は $C \equiv D (C \sqsubseteq D)$ を充足する。A-Box の言明 $C(a)$ に対して、${a^I} \in {C^I}$ のとき、I は $C(a)$ を充足する。A-Box の言明 $R(a,b)$ に対して、$({a^I}, {b^I}) \in {R^I}$ のとき、I は $R(a,b)$ を充足する。I が T-Box $\mathcal{T}$(A-Box $\mathcal{A}$) のすべての言明を充足するなら

クラス	計算量
$\mathcal{AL}$, $\mathcal{ALN}$	P
$\mathcal{ALE}$, $\mathcal{ALR}$, $\mathcal{ALER}$	NP
$\mathcal{ALU}$, $\mathcal{ALUN}$	co-NP
$\mathcal{ALC}$, $\mathcal{ALCN}$, $\mathcal{ALCR}$, $\mathcal{ALCNR}$	PSPACE
$\mathcal{ALCQI}$	ExpTime

表 2　DL 言語の計算量 [11] [12]

ば、I は $\mathcal{T}(\mathcal{A})$ のモデルであるという。

3.1.2　記述論理の推論

記述論理の重要な推論タスクは、上で述べた知識ベース (T-Box や A-Box) に対して行われる。T-Box $\mathcal{T}$ に関する推論は以下のようなものがある。

（1） 概念の充足可能性

$\mathcal{T}$ のモデルのうち概念 C のモデルとなるものが存在するかを判定する。

（2） 概念間の包摂（subsumption）、同値（equivalence）

$\mathcal{T}$ のすべてのモデル I で ${C^I} \subseteqq {D^I}$（C は D に包摂されるといい、$\mathcal{T} \models C \sqsubseteq D$ と書く）もしくは ${C^I} = {D^I}$（C と D は同値であるといい、$\mathcal{T} \models C \equiv D$ と書く）となるかを判定する。

A-Box $\mathcal{A}$ に関する推論は以下のようなものがある。

（1） 無矛盾性

$\mathcal{A}$ が充足可能であるか判定する。

（2） 概念のインスタンス

$\mathcal{A}$ のすべてのモデル I が $C(a)$ を充足するか判定する。つまり固体 a が概念 C のインスタンスであるかを調べる。

さらに記述論理は用いるコンストラクタによりさまざまなクラスが存在し、それぞれに対し充足可能性判定問題の計算の複雑さが知られている。まとめると表 2 のようになる。

記述論理の計算の複雑さに関する研究は最初多項式時間で計算可能かどうかということが問題であった。しかし [8] において推論に最悪指数時間かかるような表現力の高い記述論理でも、実際は許容できるパフォーマンスを得られることを現実のアプリケーションを用いて実証されている。これにより記述論理の計算の複雑さの評価は決定可能かそうでないかということが重要になった。

3.2　OWL Lite と OWL DL

OWL はクラスとプロパティを用いてオントロジを記述する。クラスとプロパティは記述論理ではそれぞれ概念とロールに対応する。OWL では様々なクラスコンストラクタを用いることで複雑なクラスを定義することができる。例えば前節で例にとった”子供を持つ場合はすべて男の子であるような人”は図 1 のように表現することができる。

```
<owl:Class>
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Person"/>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#child"/>
      <owl:allValuesFrom rdf:resource="#Male"/>
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

図 1　OWL のクラスの例

OWL Lite and OWL DL	DL syntax
intersectionOf	$C_1 \sqcap \cdots \sqcap C_n$
unionOf(*)	$C_1 \sqcup \cdots \sqcup C_n$
complementOf(*)	$\neg C$
oneOf(*)	$\{i_1, \cdots, i_n\}$
allValuesFrom	$\forall P.C$
someValuesFrom	$\exists P.C$
hasValue(*)	$\exists P.\{i\}$
minCardinality	$\geqslant nR$
maxCardinality	$\leqslant nR$
cardinality	$= nR$
subClassOf	$C \sqsubseteq D$
equivalentClass	$C \equiv D$
disjointWith(*)	$C \equiv \neg D$
subPropertyOf	$Q \sqsubseteq R$
equivalentProperty	$Q \equiv R$
domain	$\top \sqsubseteq \forall P^-.C$
range	$\top \sqsubseteq \forall P.C$
inverseOf	$P \equiv Q^-$
TransitiveProperty	$P^+ \sqsubseteq P$
SymmetricProperty	$P \equiv P^-$
FunctionalProperty	$\top \sqsubseteq \leqslant 1P.\top$
InverseFunctionalProperty	$\top \sqsubseteq \leqslant 1P^-.\top$

表 3　OWL Lite, OWL DL と DL の対応 [9]

また OWL ではあるクラスがあるクラスのサブクラスである（owl:subClassOf）ということや、等しい（owl:equivalentClass）ことや、交わりがない（owl:disjointWith）ということなどを表現できる。これらは記述論理では T-Box に対応する。

OWL DL は OWL Full の部分集合で OWL Full はクラスやデータタイプやプロパティや個体を区別する必要はないが、OWL DL ではそれらを明確に区別する必要がある。OWL Lite は OWL DL の部分集合で、実装しやすい機能に限定されている。さらに OWL Lite では cardinality が 0 または 1 に限られている。表 3 は OWL Lite と OWL DL の提供するコンストラクタと公理である。(*) がついているコンストラクタや公理は OWL Lite では用いることができない。OWL DL や OWL Lite に課されている制約は [13] に詳しい。OWL DL, OWL Lite は記述論理に対応しているので既存の記述論理推論システムを用いて推論を行うことができる。

[9] では OWL Lite における ontology entailment という問題が $\mathcal{SHIF}(\mathcal{D})$（$\mathcal{SHIF}$ というクラスの記述論理に concrete domain を追加したもの）の充足可能性判定問題に帰結でき、ExpTime に含まれることが示されている。同様に OWL DL における ontology entailment が $\mathcal{SHION}(\mathcal{D})$ の充足可能性問題に帰結でき、NExpTime で計算できることが示されている。つまり OWL Lite と OWL DL における推論は決定可能である。次章においてこの OWL を用いた関数従属性の表現について議論する。

4. OWL による関数従属性の表現

ここでは 2. で述べた関数従属性を OWL を用いて表現する。まず最初に関係スキーマを OWL で表現する。その後で関係スキーマにおける関数従属性を表現する。以降見やすいように記述論理の記法を用いて表記する。つまり OWL Lite と OWL DL についてのみ考える。

4.1 OWL による関係スキーマの表現

関係 $R(A_1, \cdots, A_n)$ を OWL で表現する。関係 R をクラスに対応させ、属性 $A_i (i = 1, \cdots, n)$ もクラスとする。クラス R とクラス A_i をプロパティ a_i で結ぶ。つまり reification [7] と呼ばれる次の式で n 項関連を複数の 2 項関連を用いて表わす。このときタプルがひとつ決まれば属性の値も一意に決まるので、各プロパティ a_i は functional なプロパティとする。

$$R \sqsubseteq \forall a_1.A_1 \sqcap \forall a_2.A_2 \sqcap \cdots \sqcap \forall a_n.A_n$$

$$\top \sqsubseteq \leqslant 1a_i.\top \quad \text{for each } i \in \{1, \ldots, n\}$$

例えば関係 Student(Name, Address, Phone) は

$$\mathsf{STUDENT} \sqsubseteq \forall \mathsf{name.NAME} \sqcap \forall \mathsf{address.ADDRESS} \sqcap \forall \mathsf{phone.PHONE}$$

$$\top \sqsubseteq \leqslant 1\mathsf{name}.\top,\ \top \sqsubseteq \leqslant 1\mathsf{address}.\top, \top \sqsubseteq \leqslant 1\mathsf{phone}.\top$$

で表わされる。(図 2) ただし STUDENT, NAME, ADDRESS, PHONE はクラスで name, address, phone は functional なプロパティである。

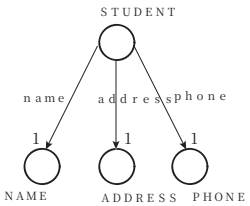

図 2　OWL による関係スキーマの表現

4.2 OWL による関数従属性の表現

ここでは 4.1 で表現した関係スキーマをもとに関数従属性を OWL を用いて表現する。

関数従属性 $X \rightarrow Y$ は属性集合 X が決まると属性集合 Y が一意に決まることを表わしている。したがってこれを OWL で表現する場合は基本的には、クラス X からクラス Y への functional なプロパティを定義してやればよい。具体的な例を挙げてみる。

[例 1] 関係 $R(ABC)$ に関数従属性 $A \rightarrow BC$ が存在する場合つまり左辺が単一属性の関数従属性の場合である。まず 4.1 より $R \sqsubseteq \forall a.A \sqcap \forall b.B \sqcap \forall c.C$, $\top \sqsubseteq \leqslant 1a.\top$, $\top \sqsubseteq \leqslant 1b.\top$, $\top \sqsubseteq \leqslant 1c.\top$ で $R(ABC)$ が表現できる。次に関数従属性 $A \rightarrow BC$ を表現する。このとき自明な関数従属性 $A \rightarrow A$ も成り立つのでアームストロングの公理系より関数従属性 $A \rightarrow ABC$ が成り立つ。これは属性 A が決まるとタプルが一意に決まることを意味している。つまりプロパティ a の逆のプロパティを functional なプロパティにすることで表現できる。プロパティ a の逆のプロパティを inv_a とすると以下の 3 式で表現できる。

$$inv\_a \equiv a^-,\ \top \sqsubseteq \leqslant 1inv\_a.\top,\ A \sqsubseteq \forall inv\_a.R$$

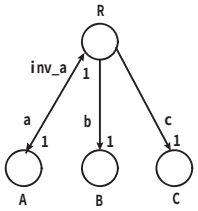

図 3　OWL による関数従属性の表現 (1)

図で表わすと図 3 のようになる。

[例 2] 関係 $R(ABC)$ に関数従属性 $AB \rightarrow C$ が存在する場合つまり左辺が複数属性の関数従属性の場合である。例 1 と同様にアームストロングの公理系を用いて、関数従属性 $AB \rightarrow ABC$ が得られる。クラス AB は存在しないので、図 4 のように作成する。つまり関係 R は $R \sqsubseteq \forall a.A \sqcap \forall b.B \sqcap \forall c.C \sqcap \forall ab.AB$, $\top \sqsubseteq \leqslant 1a.\top$, $\top \sqsubseteq \leqslant 1b.\top$, $\top \sqsubseteq \leqslant 1c.\top$, $\top \sqsubseteq \leqslant 1ab.\top$ で表現される。プロパティ ab の逆のプロパティを functional なプロパティにすることで関数従属性 $AB \rightarrow ABC$ は表現できる。プロパティ ab の逆のプロパティを inv_ab とすると以下の 3 式で表現できる。

$$inv\_ab \equiv ab^{-},\ \top \sqsubseteq \leqslant 1ab.\top,\ AB \sqsubseteq \forall a.A \sqcap \forall b.B \sqcap \forall inv\_ab.R$$

図で表わすと図 4 のようになる。

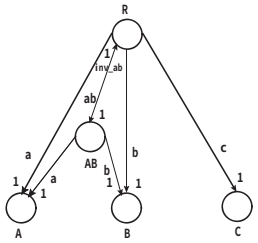

図 4　OWL による関数従属性の表現 (2)

このときクラス R のあるインスタンス i_1 はプロパティ ab を介して AB のインスタンス i_2 と関連付けられ、i_2 はプロパティ a を介して A のインスタンス i_3 と関連付けられているとする。このとき i_1 はプロパティ a を介して A のインスタンスである i_3 と関連付けられている必要がある。$i_1 \rightarrow i_3$ というパスは $R(ABC)$ を A に関して射影していて、$i_1 \rightarrow i_2 \rightarrow i_3$ というパスは $R(ABC)$ を AB に関して射影し、さらに A に関して射影している。元は同一タプルであるのでどちらのパスで射影されても、最後は同じ値にならなければならない。しかしこのように異なるパスを通って同一のインスタンスにたどり着くということは通常、ロールの合成を使って表現する。OWL ではロールの合成は提供されていないので表現できないと考えられる。さらにたとえロールの合成が使えたとしても望ましい結果が得られるわけではない。[2] ではロールの合成とロールの連言を含む記述論理を用いると、複数属性で構成されるキー属性を表現することはできるが、推論が決定不能になることが述べられている。つまり OWL が提供している機能では関数従属性を表すことはできず、また表現力の高い記述論理を用いると表すことができるが推論が決定不能になる。

以上をまとめると次のようになる。

- 左辺が単一属性の関数従属性は functional なプロパティを用いて OWL で表現することができる。
- 左辺が複数属性の関数従属性は OWL では表現できない。そのため関数従属性の推移律を用いた推論ができない。表現力の高い記述論理を用いると表現可能だが、推論が決定不能になる可能性がある。

5.　OWL の拡張

4.2 において、OWL は複数属性に関する関数従属性は表現できないことがわかった。そこでここでは OWL が関数従属性を表現できるように拡張を試みる。5.1 において OWL に関数従属性を扱う新しいコンストラクタを追加する。そのあとで 5.2 で拡張した OWL$_{fd}$ と記述論理 $\mathcal{DLFR}_{ifd}$ [6] と比較をし、OWL$_{fd}$ の推論の計算複雑さについて議論する。

5.1　OWL$_{fd}$

ここでは OWL に関数従属性を扱うコンストラクタ **fd** を追加する。**fd** は [2] でキー属性や関数従属性を扱うために追加されたコンストラクタで以下のような形をしている。

$$(\textbf{fd}\ C\ \{A_1 \cdots A_m\}\ B)$$

ただし C はクラス、$A_1, \cdots, A_m, B$ はプロパティである。また意味論は以下の式で定義される。

$$(\textbf{fd}\ \ C\ \ \{A_1 \cdots A_m\}\ B)^I = \{x | \forall y \in C^I : [{A_1}^I(x) = {A_1}^I(y) \wedge \cdots \wedge {A_m}^I(x) = {A_m}^I(y)] \rightarrow B^I(x) = B^I(y)\}$$

ただし $A_i(x), B(x)$ はそれぞれインスタンス x のプロパティ A_i, B のリンク先を表わしている。以下この **fd** を追加した OWL を OWL$_{fd}$ と定義する。特に OWL Lite に **fd** を追加したものを OWL Lite$_{fd}$、OWL DL に **fd** を追加したものを OWL DL$_{fd}$ と定義する。これを用いることで 4.2 で取り上げた例は次のように表現される。

[例 1] 関係 $R(ABC)$ に関数従属性 $A \rightarrow BC$ が存在する場合

$$R \sqsubseteq \forall a.A \sqcap \leqslant 1a \sqcap \forall b.B \sqcap \leqslant 1b \sqcap \forall c.C \sqcap \leqslant 1c$$

$$(\textbf{fd}\ R\ \{a\}\ b),\ (\textbf{fd}\ R\ \{a\}\ c)$$

[例 2] 関係 $R(ABC)$ に関数従属性 $AB \rightarrow C$ が存在する場合

$$R \sqsubseteq \forall a.A \sqcap \leqslant 1a \sqcap \forall b.B \sqcap \leqslant 1b \sqcap \forall c.C \sqcap \leqslant 1c$$

$$(\textbf{fd}\ R\ \{a\ b\}\ c)$$

5.2　OWL$_{fd}$ の計算の複雑さ

[2] では上と同じ **fd** コンストラクタを用いて関数従属性の推論を提供している。そこでは新しいコンストラクタを追加したにもかかわらず計算の複雑さは追加される前と同じに保たれている。しかし言語の表現力に問題がある。具体的には要素数制約（$\geqslant nR, \leqslant nR$）、inclusion axiom（$C \sqsubseteq D, Q \sqsubseteq R$）、逆ロー

$$
\begin{array}{c}
{\top_n}^I \subseteqq (\Delta^I)^n \\
P^I \subseteqq {\top_n}^I \\
(\$i/n : C)^I = \{t \in {\top_n}^I | t[i] \in C^I\} \\
(\neg R)^I = {\top_n}^I n R^I \\
(R_1 \sqcap R_2)^I = {R_1}^I \wedge {R_2}^I \\
{\top_1}^I = \Delta^I \\
A^I \subseteqq \Delta^I \\
(\neg C)^I = \Delta^I n C^I \\
(C_1 \sqcap C_2)^I = {C_1}^I \wedge {C_2}^I \\
(\exists [\$i] R)^I = \{d \in \Delta^I | \exists t \in R^I . t[i] = d\} \\
(\leqslant k [\$i] R)^I = \{d \in \Delta^I || \{t \in {R_1}^I | t[i] = d\}| \leqq k\}
\end{array}
$$

表 4　$\mathcal{DLR}$ のコンストラクタ [6]

ル（R^-）は提供されていない。そのため **fd** を OWL に追加し OWL$_{fd}$ を定義したがこれは単に関数従属性が表現できるようになったにすぎず **fd** を用いた推論をどのように行い、またそれがどのくらいの計算の複雑さになるかは明確でない。そこでここでは 5.1 で定義した OWL$_{fd}$ と、キー属性と関数従属性に関するコンストラクタを追加した記述論理 $\mathcal{DLR}_{ifd}$ [6] と比較し、提供する機能の違いや計算の複雑さについて議論する。そこでまず 5.2.1 において $\mathcal{DLR}_{ifd}$ の概要について説明し、5.2.2 において比較を行う。

5.2.1　$\mathcal{DLR}_{ifd}$ の概要

$\mathcal{DLR}$ [4] は n 項関係を扱えるように一般化された記述論理のクラスでデータベーススキーマやクエリーのモデル化に適している。$\mathcal{DLR}$ は概念（concept）と n 項関係（n-ary relation）で構成される。概念、n 項関係はそれぞれ次の構文に従って構成される。

$$C \rightarrow \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists [\$i] R \mid (\leqq k [\$i] R)$$

$$R \rightarrow \top_n \mid P \mid \$i/n : C \mid \neg R \mid R_1 \sqcap R_2$$

ただし C, A はそれぞれ任意の概念、基本概念を表わしていて、R, P はそれぞれ任意の関係、基本関係を表わしている。また i は関係の i 番目のコンポーネントを表わしており、n は関係の arity を表わしている。k は非負の整数である。$\mathcal{DLR}$ の T-Box は $R_1 \sqsubseteq R_2, C_1 \sqsubseteq C_2$ の形をした inclusion assertion と呼ばれる式の集合である。

これらの意味論は表 4 に示す。ただし $t[i]$ はタプル t の i 番目のコンポーネントを表わしている。また関係に対する $\neg$ コンストラクタは関係の差を表わしており、補集合ではない。

${R_1}^I \subseteqq {R_2}^I ({C_1}^I \subseteqq {C_2}^I)$ のとき、解釈 I は $R_1 \sqsubseteq R_2 (C_1 \sqsubseteq C_2)$ を充足するという。T-Box $\mathcal{T}$ のすべての式を充足する解釈を $\mathcal{T}$ のモデルといい、T-Box がモデルを持つとき T-Box は充足可能であるという。T-Box $\mathcal{T}$ が ${R^I}({C^I})$ が空でないようなモデル I を許すとき関係 R(概念 C) は充足可能であるという。T-Box $\mathcal{T}$ のすべてのモデルが α を充足するとき、α は論理的に含意されるという。$\mathcal{DLR}$ の充足可能性と論理的含意は ExpTime 完全であることが知られている。$\mathcal{DLR}$ は要素数制約を用いることでキーを表現できる。しかし単一属性がキーであることは表わせるが、複数属性がキーであることは表現できない。そこで [6] では複数属性のキー属性や関数従属性を表わすことができるように次のようなコンストラクタが追加された $\mathcal{DLR}_{ifd}$ が定義されている。

概念に対するキー制約

$$(\mathbf{id}\ C\ [\$i_1]R_1, \cdots, [\$i_h]R_h)$$

ただし C は概念で、各 R_j は関係、各 $\$i_j$ は R_j のひとつのコンポーネントを表わしている。

関係に対する関数従属性

$$(\mathbf{fd}\ R\ \$i_1, \cdots, \$i_h \rightarrow \$j)$$

ただし R は関係で、$\$i_1, \cdots, \$i_h$ はすべてのタプルを一意に決める R のコンポーネントである。これらの意味論は次のように定義される。

概念に対するキー制約

$\forall a, b \in {C^I}, \forall t_1, s_1 \in {R_1}^I, \cdots, \forall t_h, s_h \in {R_h}^I$ に対して次の 3 つの式が成り立つときに $a = b$ も成り立つ場合、解釈 I は $(\mathbf{id}\ C\ [\$i_1]R, \cdots, [\$i_h]R_h)$ を充足するという。

$$a = t_1[i_1] = \cdots = t_n[i_n]$$

$$b = s_1[i_1] = \cdots = s_n[i_n]$$

$$t_j[i_k] = s_j[i_k] \quad \text{for each } i \in \{1, \ldots, h\}, \text{ and for each } k \neq i_j$$

関係に対する関数従属性

$\forall t, s \in R^I$ に対して次の式が成り立つときに $t[j] = s[j]$ も成り立つ場合、解釈 I は $(\mathbf{fd}\ R\ \$i_1, \cdots, \$i_h \rightarrow \$j)$ を充足するという。

$$t[i_1] = s[i_1], \cdots, t[i_h] = s[i_h]$$

これらのコンストラクタを含んだ $\mathcal{DLR}$ を $\mathcal{DLR}_{ifd}$ と呼び、左辺が複数属性である関数従属性だけを扱う場合は $\mathcal{DLR}_{ifd}$ における充足可能性、論理的含意は ExpTime 完全に含まれることが示されている。逆に左辺が単一属性でキー制約ではない関数従属性を扱う場合（つまり冗長な関係スキーマの場合）は決定不能になる。左辺が単一属性でもキー制約である場合は関係 R の i 番目のコンポーネントがキーであることは $\top_1 \sqsubseteq (\leqslant 1 [\$i] R)$ で表現できる。

$\mathcal{DLR}_{ifd}$ ではこれらのコンストラクタを用いて関係モデルをモデル化する。

[例] 生徒の科目の成績と、さらにその科目を担当している教官の情報を格納している Exam(Student, Course, Prof, Grade) という関係は以下の式で表される。このとき生徒と科目が決まると担当教官と成績が一意に決まるという関数従属性 Student, Course→ Prof, Grade が存在するものとする。

$$\mathsf{Exam} \sqsubseteq (\$1/4 : \mathsf{Student}) \sqcap (\$2/4 : \mathsf{Course}) \sqcap (\$3/4 : \mathsf{Prof}) \sqcap (\$4/4 : \mathsf{Grade})$$

$$(\mathbf{fd}\ \mathsf{Exam}\ \$1, \$2 \rightarrow \$3),\ (\mathbf{fd}\ \mathsf{Exam}\ \$1, \$2 \rightarrow \$4)$$

このとき Student, Course, Prof, Grade は概念で、Exam は 4 項関係である。

次に OWL$_{fd}$ とこの $\mathcal{DLR}_{ifd}$ との比較を行う。

5.2.2 OWL$_{fd}$ と $\mathcal{DLR}_{ifd}$ との比較

ここでは OWL Lite$_{fd}$, OWL DL$_{fd}$ と 5.2.1 で説明した $\mathcal{DLR}_{ifd}$ との比較を行う。つまり表 3 で示されている OWL Lite, OWL DL のコンストラクタと **fd** は $\mathcal{DLR}_{ifd}$ ではどのように表現するか考えてみる。$\mathcal{DLR}_{ifd}$ で表現できるコンストラクタだけを持つ OWL$_{fd}$ の部分集合で左辺が複数属性の関数従属性のみを扱っている場合は、$\mathcal{DLR}_{ifd}$ に変換することで 5.2.1 で述べたように推論を ExpTime で行うことができる。表 5 は OWL Lite$_{fd}$ で提供されているコンストラクタが $\mathcal{DLR}_{ifd}$ でどのような表現になるかを示している。特に OWL$_{fd}$ の **fd** を $\mathcal{DLR}_{ifd}$ に帰着させる場合は次のようになる。OWL$_{fd}$ の **fd** は reification という手法で n 項関係を図 3 のように複数の二項関係で構成したものに対して適用される。それに対し、$\mathcal{DLR}_{ifd}$ では n 項関係を直接関係として表し、それに対し **fd** が適用される。そこで例えば OWL$_{fd}$ で $R \sqsubseteq \forall a.A \sqcap \leqslant 1a \sqcap \forall b.B \sqcap \leqslant 1b \sqcap \forall c.C \sqcap \leqslant 1c$ のように表現されていて (**fd** $R\ \{a\underline{\}}\ c$) という関数従属性が存在する場合は、$\mathcal{DLR}_{ifd}$ では $R' \sqsubseteq (\$1/3 : A) \sqcap (\$2/3 : B) \sqcap (\$3/3 : C)$ という関係を構築し、(**fd** R' \$1 \$2 → \$3) で関数従属性を表すというようにすると $\mathcal{DLR}_{ifd}$ に帰着することができる。ただし関係 R' に対して reification を適用したものが概念 R である。表 5 より OWL Lite$_{fd}$ から TransitiveProperty（$\mathcal{T}$）を除くと（それを OWL Lite$_{fd}$-$\mathcal{T}$[注1]と定義する）、$\mathcal{DLR}_{ifd}$ に帰着することが可能になる。つまり OWL Lite$_{fd}$-$\mathcal{T}$ の推論は左辺が単一属性であるキー制約でない関数従属性を含まない限り、推論を ExpTime で行うことができる。ExpTime 完全であるかは明確でない。また TransitiveProperty を除かない場合の計算複雑さについては明確ではない。

次に OWL DL$_{fd}$ が提供するコンストラクタについても考えてみる。oneOf（$\mathcal{O}$）, hasValue, TransitiveProperty($\mathcal{T}$) を除いた OWL DL$_{fd}$（OWL DL$_{fd}$-$\mathcal{OT}$ と定義する）は左辺が単一属性であるキー制約でない関数従属性を含まない限り、推論を ExpTime で行うことができる。またこれらを除かない場合の計算複雑さは明確でない。そこで次に OWL DL$_{fd}$-$\mathcal{OT}$ に oneOf を追加した OWL DL$_{fd}$-$\mathcal{T}$ の計算複雑さについて議論する。

5.2.3 OWL DL$_{fd}$-$\mathcal{T}$ の計算複雑さ

ここでは OWL DL$_{fd}$-$\mathcal{T}$ の計算複雑さについて議論する。[6] では $\mathcal{DLR}_{ifd}$ に nominal[注2] を追加すると、どのような形の関数従属性であっても決定不能になることが示されている。そのため $\mathcal{DLR}_{ifd}$+nominal をすべて OWL DL$_{fd}$-$\mathcal{T}$ に帰着することができれば OWL DL$_{fd}$-$\mathcal{T}$ の推論は決定不能になることが示される。そこでまず表 6 の写像 α を用いて $\mathcal{DLR}_{ifd}$ の知識ベース $\mathcal{K}$ を OWL DL$_{fd}$-$\mathcal{T}$ のオントロジ $\alpha(\mathcal{K})$ に変換し、さらに次の式を追加する。

$$\begin{aligned}
\top &\sqsubseteq A_{\top_1} \sqcup \cdots \sqcup A_{\top_{n_{max}}} \\
\top &\sqsubseteq \leqslant 1F_i.\top \quad \text{for each } i \in \{1, \ldots, n_{max}\} \\
\forall F_i.\bot &\sqsubseteq \forall F_{i+1}.\bot \quad \text{for each } i \in \{1, \ldots, n_{max}\}
\end{aligned}$$

(注1)：OWL Lite$_{fd}$ から TransitiveProperty($\mathcal{T}$) を除いた (−) ことを意味する。
(注2)：OWL では oneOf に対応する。

OWL Lite$_{fd}$, OWL DL$_{fd}$	DL syntax	DLR_{ifd}
intersectionOf	$C \sqcap D$	$C \sqcap D$
someValuesFrom	$\exists P.C$	$\exists[\$1](P \sqcap \$2/2 : C)$
allValuesFrom	$\forall P.C$	$\neg\exists[\$1](P \sqcap \$2/2 : \neg C)$
maxCardinality	$\leqslant nR$	$\leqslant n[\$1]R$
minCardinality	$\geqslant nR$	$\neg(\leqslant (n-1)[\$1]R)$
cardinality	$= nR$	$(\leqslant n[\$1]R) \sqcap (\neg(\leqslant (n-1)[\$1]R))$
subClassOf	$C \sqsubseteq D$	$C \sqsubseteq D$
equivalentClass	$C \equiv D$	$C \sqsubseteq D, D \sqsubseteq C$
subPropertyOf	$P \sqsubseteq Q$	$P \sqsubseteq Q$
equivalentProperty	$P \equiv Q$	$P \sqsubseteq Q, Q \sqsubseteq P$
domain	$\top \sqsubseteq \forall P^-.C$	$\top_1 \sqsubseteq \neg\exists[\$2](P \sqcap (\$1/2 : \neg C))$
range	$\top \sqsubseteq \forall P.C$	$\top_1 \sqsubseteq \neg\exists[\$1](P \sqcap (\$2/2 : \neg C))$
inverseOf	$P \equiv Q^-$	$P \sqsubseteq (\$1/2 : \exists[\$2]Q) \sqcap (\$2/2 : \exists[\$1]Q)$ $(\$1/2 : \exists[\$2]Q) \sqcap (\$2/2 : \exists[\$1]Q) \sqsubseteq P$
TransitiveProperty	$P^+ \sqsubseteq P$	帰着できない
SymmetricProperty	$P \equiv P^-$	$P \sqsubseteq (\$1/2 : \exists[\$2]P) \sqcap (\$2/2 : \exists[\$1]P)$ $(\$1/2 : \exists[\$2]P) \sqcap (\$2/2 : \exists[\$1]P) \sqsubseteq P$
FunctionalProperty	$\top \sqsubseteqleqslant 1P.\top$	$\top_1 \sqsubseteqleqslant 1[\$1](P \sqcap (\$2/2 : \top_1))$
InverseFunctionalProperty	$\top \sqsubseteqleqslant 1P^-.\top$	$\top_1 \sqsubseteqleqslant 1[\$2](P \sqcap (\$1/2 : \top_1))$
FunctionalDependency	(**fd** $R\ \{a_1, \cdots, a_{m-1}\}\ a_m$)	(**fd** R' $\$1, \cdots, \$m-1 \rightarrow \$m$)
unionOf	$C \sqcup D$	$C \sqcup D$
complementOf	$\neg C$	$\neg C$
oneOf	$\{i_1, \cdots, i_n\}$	帰着できない
hasValue	$\exists P.\{i\}$	帰着できない
disjointWith	$C \sqsubseteq \neg D$	$C \sqsubseteq \neg D$

表 5　OWL Lite$_{fd}$, OWL DL$_{fd}$ と $\mathcal{DLR}_{ifd}$ の対応

$$\begin{aligned}
A_{\top_n} &\equiv \exists F_1.A_{\top_1} \sqcap \cdots \sqcap \exists F_n.A_{\top_1} \sqcap \forall F_{n+1}.\bot \\
&\qquad \text{for each } n \in \{2, \ldots, n_{max}\} \\
A_P &\sqsubseteq A_{\top_n} \quad (P \text{ は各 n 項基礎関係}) \\
A &\sqsubseteq A_{\top_1} \quad (A \text{ は各基礎概念})
\end{aligned}$$

この変換は 4.1 でも述べたように reification [7] という手法を再現している。つまり知識ベース $\mathcal{K}$ 内のタプルはオントロジ $\alpha(\mathcal{K})$ ではそれぞれのタプルの i 番目のコンポーネントに結びつく functional なプロパティ F_i を持つ個体によって表現される。例えば 5.2.1 の例で出てきた関係 Exam は表 6 に従って次のように変換される。

$$\begin{aligned}
\mathsf{A_{Exam}} &\sqsubseteq A_{\top_4} \sqcap \forall F_1.\mathsf{Student} \sqcap \forall F_2.\mathsf{Course} \\
&\qquad \sqcap \forall F_3.\mathsf{Prof} \sqcap \forall F_4.\mathsf{Grade} \\
(\textbf{fd}\ \ \mathsf{A_{Exam}}\ \ \{F_1\ F_2\}\ F_3),\ &(\textbf{fd}\ \mathsf{A_{Exam}}\ \{F_1\ F_2\}\ F_4) \\
\top &\sqsubseteq A_{\top_1} \sqcup \cdots \sqcup A_{\top_4} \\
\top &\sqsubseteq \leqslant 1F_i.\top \quad \text{for each } i \in \{1, \ldots, 4\} \\
\forall F_i.\bot &\sqsubseteq \forall F_{i+1}.\bot \quad \text{for each } i \in \{1, \ldots, 4\} \\
A_{\top_n} &\equiv \exists F_1.A_{\top_1} \sqcap \cdots \sqcap \exists F_n.A_{\top_1} \sqcap \forall F_{n+1}.\bot \\
&\qquad \text{for each } n \in \{2, \ldots, 4\} \\
A &\sqsubseteq A_{\top_1} \quad (A \text{ は各基礎概念})
\end{aligned}$$

表 6 より $\mathcal{DLR}_{ifd}$+nominal が OWL DL$_{fd}$-$\mathcal{T}$ に完全には変換さ

$\mathcal{DLR}_{ifd}$	OWL DL$_{fd}$
$\alpha(\top_n)$	$A_{\top_n}$
$\alpha(P)$	A_P
$\alpha((\$i/n:C))$	$A_{\top_n} \sqcap \forall F_i.\alpha(C)$
$\alpha(\neg R)$	$A_{\top_n} \sqcap \neg\alpha(R)$
$\alpha(R_1 \sqcap R_2)$	$\alpha(R_1) \sqcap \alpha(R_2)$
$\alpha(\top_1)$	$A_{\top_1}$
$\alpha(A)$	A
$\alpha(\neg C)$	$A_{\top_1} \sqcap \neg\alpha(C)$
$\alpha(C_1 \sqcap C_2)$	$\alpha(C_1) \sqcap \alpha(C_2)$
$\alpha(\{a_1, \cdots, a_m\})$	$\{a_1, \cdots, a_m\}$
$\alpha(\exists[\$i]R)$	$\exists F_i^{-}.\alpha(R)$
$\alpha(\leqslant k[\$i]R)$	帰着できない
$\alpha((\mathbf{id}\ C\ [\$i_1]R_1, \cdots, [\$i_h]R_h))$	不明
$\alpha((\mathbf{fd}\ P\ \$i_1 \cdots \$i_h \rightarrow \$j))$	$(\mathbf{fd}\ A_P\ \{F_{i_1}, \cdots, F_{i_h}\}\ F_j)$

表 6　$\mathcal{DLR}_{ifd}$+nominal から OWL DL$_{fd}$-$\mathcal{T}$ への変換

れていないことがわかる。$\alpha(\leqslant k[\$i]R)$ は実際は $\leqslant kF_i^{-}.\alpha(R)$ に変換されるのだが、OWL DL$_{fd}$-$\mathcal{T}$ での数値制約は $\leqslant P$ という形（表 1 では $\mathcal{N}$ に対応）しか許しておらず、$\leqslant P.C$(表 1 では $\mathcal{Q}$ に対応) という形は許していないので”帰着できない”としている。$\mathcal{N}$ は $\mathcal{Q}$ の特別な場合であるため表現力は低い。そのためこれが決定可能性に影響する可能性も考えられる。表 2 より $\mathcal{Q}$ を含む記述論理は ExpTime に含まれるが、N を含む記述論理はそうでないことからも決定可能になる可能性があることがわかる。また $\mathcal{DLR}_{ifd}$+nominal の **id** というコンストラクタは OWL DL$_{fd}$-$\mathcal{T}$ ではどのような表現に対応するかは明確となっていない。そのためこの点からも OWL DL$_{fd}$-$\mathcal{T}$ は決定不能となるとはいえず、決定可能である可能性もある。

以上をまとめると次のようになる。

- $\mathcal{DLR}_{ifd}$ の $\leqslant k[\$i]R$ という表現は OWL DL$_{fd}$-$\mathcal{T}$ では表現できない。この機能が決定可能性にどのくらい影響しているかは現段階では明確でないが、決定可能になることも考えられる。
- $\mathcal{DLR}_{ifd}$ のコンストラクタ **id** が OWL DL$_{fd}$-$\mathcal{T}$ ではどういう表現に対応するか明確でない。

上記のことが明確になれば OWL DL$_{fd}$-$\mathcal{T}$ の推論の計算複雑さもわかる。

6. まとめと今後の課題

本稿ではオントロジ記述言語 OWL を用いて関係モデルにおける関数従属性のモデル化を試みた。OWL は複雑な関数従属性を表わせるほどの表現力はもっておらず、関数従属性を表現するためには特別なコンストラクタを追加する必要があることを述べた。さらにその関数従属性を表現するためのコンストラクタ **fd** を OWL に追加し、その推論や計算量について議論した。OWL Lite$_{fd}$ は TransitiveProperty というコンストラクタを除くと（OWL Lite$_{fd}$-$\mathcal{T}$）、$\mathcal{DLR}_{ifd}$ というクラスの記述論理に帰着することができ、推論が ExpTime で行われることを確かめた。また OWL DL$_{fd}$ はさらにそれから oneOf, hasValue を除く（OWL DL$_{fd}$-$\mathcal{OT}$）と $\mathcal{DLR}_{ifd}$ に帰着することができ、推論が ExpTime で行われることを確かめた。さらに OWL DL$_{fd}$-$\mathcal{OT}$ に oneOf を追加した OWL DL$_{fd}$-$\mathcal{T}$ の計算量についても議論した。しかし現段階では不明な点もあり決定可能かどうかはわからないが、計算の複雑さに大きく影響を与えると考えられる Q を OWL DL$_{fd}$-$\mathcal{T}$ は含まないため、決定可能になる可能性もあることを述べた。どのコンストラクタも除かない純粋な OWL Lite$_{fd}$ や OWL DL$_{fd}$ の計算量についてもいまだ明確ではない。今後の課題として次のようなことがあげられる。

- OWL DL$_{fd}$-$\mathcal{OT}$ に oneOf を追加したもの（OWL DL$_{fd}$-$\mathcal{T}$）の計算複雑さ
- OWL Lite$_{fd}$-$\mathcal{T}$ に TransitiveProperty を追加したもの（OWL Lite$_{fd}$）の計算複雑さ
- OWL DL$_{fd}$-$\mathcal{OT}$ に TransitiveProperty を追加したもの（OWL DL$_{fd}$-$\mathcal{O}$）の計算複雑さ
- path functional dependency [10] を扱うコンストラクタの追加

文　献

[1] A. Borgida, ”Description Logics in Data Management”, IEEE TKDEvol.7, no.5, pages 671–682, 1995.

[2] Alex Borgida and Grant Weddell, ”Adding Uniqueness Constraints to Description Logics(Preliminary Report)”, Proceedings of the 5th International Conference on Deductive and ObjectOriented Databases(DOOD97), volume 1341 of LNCS, pages 85–102. Springer, 1997.

[3] D. Calvanese, M. Lenzerini, D. Nardi, ”A Unified Framework for Class-Based Representation Formalisms”, Proc. of KR-94, pages 109–120, 1994.

[4] D.Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, ”Description Logic Framework for Information Integration”, In Proc. of the 6th Int. Conf. on the Principles of Knowledge Representaion and Reasoning (KR-98), pages 2–13, 1998.

[5] D. Calvanese, G. De Giacomo, M. Lenzerini, ”Keys for Free in Description Logics”, Proc. of the 2000 Description Logic Workshop (DL2000), pages 79–88, CEUR Electronic Workshop Proceedings.

[6] D. Calvanese, G. De Giacomo, M. Lenzerini, ”Identification Constraints and Functional Dependencies in Description Logics”, Proc. of the 17th Int. Joint Conf. on Artificial Intelleigence (IJCAI2001).

[7] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P. Patel-Schneider: The Description Logic Handbook, CAMBRIDGE, 2003.

[8] Horrocks, I. R.: Using an Expressive Description Logic: FaCT or Fiction?, KR’98: Principles of Knowledge Representation and Reasoning , pp. 636–645 (1998).

[9] Ian Horrocks and Peter F. Patel-Schneider, ”Reducing OWL Entailment to Description Logic Satisfiability”, 2nd International Semantic Web Conference (ISWC-03), 2003.

[10] Vitaliy L. Khizder, David Toman, Grant Weddell, ”On Decidability and Complexity of Description Logics with Uniqueness Constraints”, Proc. of the8th International Conference on Database Theory (ICDT 2001).

[11] 兼岩憲, 佐藤健, ”DL: Description Logics”, 人工知能学会誌, Vol.18, No.1, pages.73–82, 2003 .

[12] 中西, 三浦, 塩谷, ”記述論理による UML の表現”, データベースワークショップ（DBWS’2003）, 情報処理学会データベース研究会, 2003.

[13] ”OWL Web Ontology Language Reference”. http://www.w3.org/TR/2003/CR-owl-ref/