

OWL による型付き単一化代入

Typed Unification based on OWL

小出 誠二*¹
Seiji Koide

武田 英明*²
Hideaki Takeda

*¹国立情報学研究所，総研大
National Institute of Informatics, Sokendai

*²国立情報学研究所
National Institute of Informatics

Unification is an elemental technique that is commonly used for reasoning in First-Order Logic systems, logic languages, and planners, etc. When we develop applications equipped with ontologies described in OWL, the unification between variables and individuals typed to OWL classes is required. In this paper, we extend conventional type-less unification to OWL typed unification. Since the data structure and variable binding mechanism is shared by tableau, satisfiability checking by tableau in OWL is naturally and intrinsically realized in the unification process. This technique will be useful to logics on top of the OWL layer.

1. はじめに

単一化 (unification) は、一階述語論理システム、Prolog のような論理型言語、計画システムなどで共通に用いられる推論の基礎技術である。OWL で書かれたドメインオントロジーを用いてアプリケーションを開発しようとするとき、OWL クラスで型付けされた論理変項を対象とした単一化代入が必要となる。我々は OWL をドメインオントロジー記述に用いたセマンティックウェブサービスの研究を進めているが、本報告ではウェブサービスの合成・分解への応用を目的として、OWL の型を考慮した単一化代入について報告する。

記述論理 (Description Logic, DL) による文は、一階述語論理の自由変項が 1 個のフラグメントに相当する。したがって、記述論理による二つの文を一緒にして推論を行おうとする場合にも、必然的に OWL の型を考慮した単一化代入が必要となる。W3C の提唱するルールやロジックのレイヤを、OWL レイヤの上に構築するためにも、OWL 文に論理変項を持ち込み、単一化代入によって論理レベルでの推論を可能にすることが必須である。

本論文では、記述論理と一階述語論理との関係の議論からはじめて、Russell と Norvig の示した単一化アルゴリズム [Russell 95] を OWL クラスと個物に適用できるように拡張し、OWL による型付き単一化のアルゴリズムと実装の結果について述べる。

2. 型付き単一化

単一化 (unification) とは、二つの文を引数にとり、それらを同一にする単一化代入 (unifier) を求めて、それを実行することで推論を進める技術である。型の無い単一化では、単純に論理変項同士あるいは論理変項と論理定項の間で代入が行われる。例えば、一階述語論理において次のルールがあったとしよう。

$$\text{Knows}(\text{John}, x) \Rightarrow \text{Hates}(\text{John}, x)$$

この表記における自由変項 x は、全称限量されているかのようには扱われる。今知識ベースに次のような文があると、

$$\begin{aligned} &\text{Knows}(\text{John}, \text{Jane}) \\ &\text{Knows}(y, \text{Leonid}) \\ &\text{Knows}(y, \text{Mother}(y)) \\ &\text{Knows}(x, \text{Elizabeth}) \end{aligned}$$

連絡先: 小出誠二, 国立情報学研究所総研大, 101-8430 東京都千代田区一ツ橋 2-1-2, koide@nii.ac.jp

ルールの前提と知識ベースの文を単一化することによって、次のような結果を得る。

$$\begin{aligned} \text{Unify}(\text{Knows}(\text{John}, x), \text{Knows}(\text{John}, \text{Jane})) &= \{x/\text{Jane}\} \\ \text{Unify}(\text{Knows}(\text{John}, x), \text{Knows}(y, \text{Leonid})) &= \{x/\text{Leonid}, y/\text{John}\} \\ \text{Unify}(\text{Knows}(\text{John}, x), \text{Knows}(y, \text{Mother}(y))) &= \{y/\text{John}, x/\text{Mother}(\text{John})\} \\ \text{Unify}(\text{Knows}(\text{John}, x), \text{Knows}(x', \text{Elizabeth})) &= \{x'/\text{Elizabeth}, x'/\text{John}\} \end{aligned}$$

最後の単一化では操作の前に変項の標準化が施されている。得られた結果をルールの帰結部に代入することにより、次の結論が得られる。

$$\begin{aligned} &\text{Hates}(\text{John}, \text{Jane}) \\ &\text{Hates}(\text{John}, \text{Leonid}) \\ &\text{Hates}(\text{John}, \text{Mother}(\text{John})) \\ &\text{Hates}(\text{John}, \text{Elizabeth}) \end{aligned}$$

型無しの単一化では、述語が一致するなど他の条件が満たされれば、論理変項はどんな変項や論理定項にも単一化が可能である。上記例では x は人間だけでなく、石や水にでも単一化され得る。そこで、これに型を持ち込み、論理定項も論理変項も次のように型付けされているとしよう。

表記	意味
$a:C$	$a \in C$, ここで a は OWL の個物
$x:C$	$x \in C$, ここで x は論理変項

C は OWL のクラスである。型付きの論理変項は、OWL クラスによって型付けされているので、個物の一種と見ることもできるが、それは単一化の操作において、その型の範囲内で任意の個物に同一化され得るものであり、その型と整合するものとは単一化されないとする。

今クラス *Human* がドメインオントロジーに定義されているとして、先のルールを次のように型付きで書き表す。

$$\begin{aligned} &\text{Knows}(\text{John}:\text{Human}, x:\text{Human}) \\ \Rightarrow &\text{Hates}(\text{John}:\text{Human}, x:\text{Human}) \end{aligned} \quad (1)$$

もし知識ベースが次のようであったなら、

$$\begin{aligned} &\text{Knows}(\text{John}:\text{Human}, \text{Jane}:\text{Human}) \\ &\text{Knows}(y:\text{Human}, \text{Leonid}:\text{Human}) \\ &\text{Knows}(y:\text{Human}, \text{Mother}(y):\text{Human}) \\ &\text{Knows}(x:\text{Human}, \text{Elizabeth}:\text{Cat}) \end{aligned}$$

型付き単一化によって、John は知っている人間は憎むが Elizabeth は憎まないように推論することができる。

3. 記述論理と一階述語論理

記述論理と一階述語論理との関係から型付き単一化について考察する．記述論理におけるクラス外延は一階述語論理における1項述語 (unary predicate) による原子文であり，ロール外延は2項述語 (binary predicate) による原子文である．

3.1 1項述語とクラス外延

先の型付きルールは一階述語論理では次のように書き表される．

$$Human(John) \wedge Human(x) \wedge Knows(John, x) \Rightarrow Hates(John, x)$$

一方，次のような一階述語論理式があったとき，通常の推論により $Likes(John, Elizabeth)$ と推論される．ここで知識ベースには $Cat(Elizabeth)$ はあるが $Human(Elizabeth)$ はないと仮定している．

$$Human(John) \wedge Cat(x) \wedge Knows(John, x) \Rightarrow Likes(John, x)$$

すなわち，

$$\begin{aligned} Knows(John: Human, x: Cat) \\ \Rightarrow Likes(John: Human, x: Cat) \end{aligned} \quad (2)$$

である．それでは，次のような論理式ではどうすべきであろうか．

$$Human(John) \wedge Pet(x) \wedge Knows(John, x) \Rightarrow Likes(John, x)$$

$$\begin{aligned} Knows(John: Human, x: Pet) \\ \Rightarrow Likes(John: Human, x: Pet) \end{aligned} \quad (3)$$

もしオントロジーにおいて $Cat \sqsubseteq Pet$ であることが分かっていたら，(2) 式の帰結と同様に $Likes(John, Elizabeth)$ が帰結されるべきであろう．すなわち $x: Pet$ と $Elizabeth: Cat$ の単一化は成功して $\{x: Cat/Elizabeth: Cat\}$ が返されるべきである．逆に， $x: Cat$ と $Elizabeth: Pet$ の単一化では $\{x: Cat/Elizabeth: Cat\}$ が返されるべきであり，代入の結果として $Elizabeth: Cat$ という特殊化がされるべきであろう．

本論文の目的とするところは，一般に型付き変項同士や型付き変項と型付き定項との間の単一化について，ウェブサービスの合成・分解に対してもっともらしい定式化を行おうとするところにある．

型付き単一化の定式化においては性格の異なる種々の定式化が可能であるが，我々ここでは知識の単調増加原則に則り，すべての操作は抽象クラスから特殊クラスへ具体化されるものとする．我々の考えるウェブサービス合成分解問題では，状態空間 (state space) 中に個物のみならず型付き変項があり，推論の進行に伴って状態空間中の変項とウェブサービスの入出力との間で単一化代入が行われ，状態空間中の変項は最初は抽象的なクラスから特殊なクラスに特殊化されていくものとしている．

なお，ここで提案する型付き単一化による推論能力には，色々な意味で限界がある．上記例で言えば，

$$\begin{aligned} Knows(John: Human, x: Baby) \\ \Rightarrow Likes(John: Human, x: Baby) \end{aligned} \quad (4)$$

という推論ルールに対して $Elizabeth: Human$ と $Elizabeth: Baby$ という知識があったとき， $Hates(John, Elizabeth)$ とすべきであろうか，それとも $Likes(John, Elizabeth)$ とすべきであろうか．今述語について何の知識もなければ，単純には両方とも成立する．一階述語論理においては推論ルールを書

きさえすれば，この問題を解くことはできるであろうが，記述論理の枠組みの内ではこれを扱うことはできない．OWL においてプロパティの包含関係を定義することはできるが， $Likes$ と $Hates$ が反意語の関係にあることは定義できず，その引数に対して disjoint であるように制約を設けることもできない．定義域制約と値域制約を論理変項として，一階述語論理においてそれを扱うことができるようにすれば，この問題を解決することができるかも知れないが，それは本論文の範囲を超える問題である．

3.2 2項述語とプロパティ外延

OWL あるいは記述論理では次のような表現がよく現れる．

$$Parent \equiv Human \sqcap \exists HasChild$$

一階述語論理でこれを表現すれば次のようになる．

$$Parent(x) \Leftrightarrow Human(x) \wedge \exists y. HasChild(x, y)$$

この文は1個の自由変項 x を有するが，論理変項 y については存在限量子でクローズされている． $HasChild(x, y)$ には知識ベース中の三つ組みにおいて，プレディケイトが $HasChild$ であるすべての三つ組み ($HasChild$ のプロパティ外延) が相当し，任意の x について ABox 中に $HasChild(x, y)$ となる y の個物が一つでも存在すれば， $\exists y. HasChild(x, y)$ は充足する．

さてここで次のようなルールを考えよう．

$$\begin{aligned} Human(x) \wedge HasChild(x, y) \Rightarrow Likes(x, y) \\ HasChild(x: Human, y: \top) \Rightarrow Likes(x: Human, y) \end{aligned} \quad (5)$$

今，次のような単一化が行われたとすると，

$$\begin{aligned} Unify(HasChild(x: Human, y), HasChild(John: Human, Elizabeth)) \\ = \{y/Elizabeth, x/John\} \\ Unify(HasChild(John: Human, y), HasChild(x: Human, Elizabeth)) \\ = \{y/Elizabeth, x/John\} \end{aligned}$$

いずれの場合も得られた結果について，(5) 式右辺に単一化代入を施して $Likes(John, Elizabeth)$ を得たい．しかし，次のようなルールであれば，

$$\begin{aligned} \exists y. Human(x) \wedge HasChild(x, y) \Rightarrow Likes(x, y) \\ \exists y. HasChild(x: Human, y: \top) \Rightarrow Likes(x: Human, y) \end{aligned} \quad (6)$$

実際に ABox 中に $HasChild(John, Elizabeth)$ という知識があった場合にのみ， $Likes(John, Elizabeth)$ という結果を得たい．(6) 式はスコレム関数化により存在限量された変項を削除しなければならない．

$$HasChild(x: Human, F(x)) \Rightarrow Likes(x: Human, F(x)) \quad (7)$$

すなわち，スコレム関数化により単一化代入は次のようになるが，

$$\begin{aligned} Unify(HasChild(x: Human, F(x)), HasChild(John, Elizabeth)) \\ = \{F(x)/Elizabeth, x/John\} \\ Unify(HasChild(John: Human, F(x)), HasChild(x, Elizabeth)) \\ = \{F(x)/Elizabeth, x/John\} \end{aligned}$$

これらのスコレム関数を含む代入に関しては，常にその代入が ABox に適合するかどうか，この場合では $HasChild(John, Elizabeth)$ が ABox にあるかどうかをチェックしなければならない．

4. 型付き単一化のアルゴリズム

4.1 型無し単一化のアルゴリズム

あとで型付き単一化に拡張することを目的に、ここでは型無し単一化のアルゴリズムについて記述する。Russel と Norvig の示した単一化アルゴリズム [Russell 95] を以下に示す。

function Unify(x, y) **returns** a substitution to make x and y identical, if possible

Unify-Internal($x, y, \{\}$)

function Unify-Internal(x, y, θ) **returns** a substitution to make x and y identical (given θ)

inputs: x , a varibale, constant, list, or compound
 y , a varibale, constant, list, or compound
 θ , the substitution built up so far

```
if  $\theta = \text{failure}$  then return failure
else if  $x = y$  then return  $\theta$ 
else if Variable?( $x$ ) then return Unify-Var( $x, y, \theta$ )
else if Variable?( $y$ ) then return Unify-Var( $y, x, \theta$ )
else if Compound?( $x$ ) and Compound?( $y$ ) then
  return Unify-Internal(Args( $x$ ),Args( $y$ ),
    Unify-Internal(Op( $x$ ),Op( $y$ ), $\theta$ ))
else if List?( $x$ ) and List?( $y$ ) then
  return Unify-Internal(Rest( $x$ ),Rest( $y$ ),
    Unify-Internal(First( $x$ ),First( $y$ ), $\theta$ ))
else return failure
```

function Unify-Var(var, x, θ) **returns** a substitution

inputs: var , a varibale
 x , any expression
 θ , the substitution built up so far

```
if  $\{var/val\} \in \theta$ 
  then return Unify-Internal( $val, x, \theta$ )
else if  $\{x/val\} \in \theta$ 
  then return Unify-Internal( $var, val, \theta$ )
else if  $var$  occurs anywhere in  $x$  /* occure-check */
  then return failure
else return add  $\{var/x\}$  to  $\theta$ 
```

4.2 型付き単一化への拡張

上記アルゴリズムにおける引数は型無しの論理変項や定項であるが、これを型付きにする。OWL を考慮した型付きの単一化において、個物の同一性を判定するために、owl:sameAs や owl:differentFrom は重要である。二つの名前が異なってもそれが同じ個物を指すという意味の owl:sameAs 関係を記号 \doteq で表す。同様に、クラスの等価性 (owl:equivalentClass) を \simeq で表す。二つの個物が異なるものであるという owl:differentFrom 関係を \neq と表記する。我々の目的に合致した型付きの引数を受け取る TUnify-Internal は次のようになる。

function TUnify-Internal($x:C, y:D, \theta$) **returns**

inputs: $x:C$, a varibale, constant, or compound typed to C
 $y:D$, a varibale, constant, or compound typed to D
 θ , the substitution built up so far

```
if  $\theta = \text{failure}$  then return failure
else if  $x \neq y$  then return failure
else if  $x \doteq y$  then return  $\theta$ 
else if Variable?( $x$ ) then
  return TUnify-Var( $x, y, \theta$ )
else if Variable?( $y$ ) then
  return TUnify-Var( $y, x, \theta$ )
else if Compound?( $x$ ) and Compound?( $y$ ) then
  return TUnify-Internal(Args( $x$ ),Args( $y$ ),
    TUnify-Internal(Op( $x$ ),Op( $y$ ), $\theta$ ))
else if List?( $x$ ) and List?( $y$ ) then
  return TUnify-Internal(Rest( $x$ ),Rest( $y$ ),
    TUnify-Internal(First( $x$ ),First( $y$ ), $\theta$ ))
else return failure
```

x と y が定項で、 \doteq も \neq も定義されていないとすると、上記アルゴリズム最後で単一化は失敗する。TUnify-Internal は外見上はほとんど Unify-Internal と変わるところはない。重要な拡張はすべて TUnify-Var にある。

function TUnify-Var($var:C, x:D, \theta$) **returns** a substitution

inputs: $var:C$, a varibale typed to C
 $x:D$, any expression typed to D
 θ , the substitution built up so far

```
if  $\{var:C/val:E\} \in \theta$  then
  if Unsatisfiable?( $C \sqcap E$ ) then return failure
  else if  $C \simeq E$  then
    return TUnify-Internal( $val, x, \theta$ )
  else if  $C \sqsubseteq E$  then
    return TUnify-Internal( $val:C, x, \theta$ )
  else if  $E \sqsubseteq C$  then
    return TUnify-Internal( $val, x, \theta$ )
  else
    return TUnify-Internal( $val:(C \sqcap E), x, \theta$ )
else if  $\{x:D/val:E\} \in \theta$ 
  if Unsatisfiable?( $D \sqcap E$ ) then return failure
  else if  $D \simeq E$  then
    return TUnify-Internal( $var, val, \theta$ )
  else if  $D \sqsubseteq E$  then
    return TUnify-Internal( $var, val:D, \theta$ )
  else if  $E \sqsubseteq D$  then
    return TUnify-Internal( $var, val, \theta$ )
  else
    return TUnify-Internal( $var, val:(D \sqcap E), \theta$ )
else if  $var:C$  occurs anywhere in  $x:D$  /* occure-check */
  then return failure
else if Unsatisfiable?( $C \sqcap D$ ) then return failure
else if  $C \simeq D$  then return add  $\{var:C/x:D\}$  to  $\theta$ 
else if  $C \sqsubseteq D$  then return add  $\{var:C/x:C\}$  to  $\theta$ 
else if  $D \sqsubseteq C$  then return add  $\{var:D/x:D\}$  to  $\theta$ 
else return add  $\{var:(C \sqcap D)/x:(C \sqcap D)\}$  to  $\theta$ 
```

Unsatisfiable? $(C \sqcap D)$ は、概念 C と概念 D が disjoint の関係にあれば真である。先の例では Cat と $Human$ が disjoint であるという知識があれば、この単一化は成功しない。一方、 $Cat \sqsubseteq Pet$ という知識があれば、(2) 式でも (3) 式でも $Elizabeth: Cat$ が θ に追加される。もし $Human$ と $Baby$ が disjoint でもなければ、包含関係にもないという場合には、 $Elizabeth: Human$ という知識と (3) 式から、 $Elizabeth: Baby$ という知識がなくても、 $Human \sqcap Baby$ という概念が新しく生成され、 $Likes(John, Elizabeth: (Human \sqcap Baby))$ が帰結される。

このような単一化は、はたしてやりすぎであろうか。開世界仮説 (Open World Assumption) に立つセマンティックウェブにおいては、クラスに対しての owl:disjointWith 記述は極めて重要である。この記述が無い二つの概念は常に同一のものを指しているのかも知れないという可能性を否定できない。ここで我々が提案する単一化の方式は、セマンティックウェブでのこのような特徴に配慮し、owl:disjointWith 記述が必要十分にオントロジーに記述されていることを前提として、最大の可能性を汲み尽くすという態度からきている。名前の唯一性仮説 (Unique Name Assumption) については、6 章で考察する。

5. 型付き単一化アルゴリズムの実装とタブロー法

本節では、上記単一化アルゴリズムの実装方法について述べるとともに、記述論理におけるタブロー法との類似性を指摘し、タブロー法と統合された単一化アルゴリズムについて述べる。

5.1 記述論理におけるタブロー法

Baader ら [Baader 03] によるタブロー法では、ある概念 (複合概念でよい) の充足性判定にその概念の個物を一個含む ABox を考え、TBox における記述と変換ルールにしたがって複合概念を分解した個物の定義を ABox に追加していき、ABox に含まれる記述に整合性がないことが判明したとき (それを clash と呼ぶ)、最初概念を充足不能とする。clash しない場合にはそれが TBox から伴意されないまでも充足可能である。すなわちその概念を TBox に追加することができる。ちなみに、ある TBox を充足するどのような解釈も明示的には TBox に含まれないある概念を充足するとき、その概念はその TBox に伴意されると言う。

たとえば、 $(Human \sqcap Baby)$ の充足性判定を行うには、まず最初にこの個物である $x: (Human \sqcap Baby)$ のみを含む ABox を作る。次に \sqcap 変換規則にしたがって、 $x: Human$ と $x: Baby$ を ABox に加える。 $Human$ や $Baby$ が複合概念でなければ、推論はここで終了して ABox 中に不整合はないので、 $(Human \sqcap Baby)$ は充足可能である。 $Human$ や $Baby$ が複合概念であればさらにそれぞれについて、様々な変換規則にしたがって個物の定義が ABox に追加されていく。もし ABox 中にある概念 C について $x: C$ と $x: \neg C$ という不整合が存在すれば、元の概念は充足不能である。

前節のアルゴリズムにおける Unsatisfiable? $(C \sqcap D)$ の充足性判定においてタブロー法を用いることとする。

5.2 単一化代入とタブロー法と SWCLOS の統合

SWCLOS [Koide 06] は、Common Lisp Object System (CLOS) [Kiczales 91] 上に構築された、OWL Full レベルの推論システムである。これまで SWCLOS においては、構造的包摂アルゴリズム (structural subsumption algorithm) [Baader 03] は行っていない、タブロー法による包摂関係充足

判定機能はなく、owl:unionOf と owl:someValuesFrom において推論が完全ではないという欠点を抱えていた。今回新しく、タブロー法を SWCLOS に導入し、単一化代入の実現を SWCLOS とタブロー法を用いて行うこととした。

SWCLOS ではある個物は CLOS におけるオブジェクトそのものであり、それはその個物の名前 (QName) がある場合にはその名前である Lisp シンボルの値としてバインドされている。匿名個物の場合には、Lisp のトップレベルからそれを指す手段はなく、他のオブジェクトのスロット値 (ロールのフィラー) として指されている。

タブロー法における ABox では、 $(C \sqcap D)(x)$ や $C(x)$ および $D(x)$ を実現する必要がある。Russel と Norvig の単一化アルゴリズム [Russell 95] には、Norvig による Lisp のソースコードが公開^{*1}されているが、そこでは単一化代入 $\{x/y, \dots\}$ の実現に Lisp の連想リストが用いられている。連想リストは初期の Lisp ではシンボル値のバインディング機構そのものであった。型付き変項の単一化代入 $\{x: C/y: D, \dots\}$ の実現でも、 $x: C$ すなわち $C(x)$ の実現について連想リストによる仮想的なバインディング機構を用いて、SWCLOS の名前空間と単一化操作における見かけ上の統合を行うことにした。すなわち、単一化バインディング装置 $\{x/y, \dots\}$ と同様に、SWCLOS と統合されたシンボルのバインディング装置 $\{x/C(), \dots\}$ を実現する。第 4.2 節のアルゴリズムにおいて、論理変項と定項の単一化には前者が用いられるが、その型情報には後者が用いられる。そして $(C \sqcap D)$ の充足性判定には両者が用いられる。こうすることで、単一化のバインディングには従来と同様なルーチンが用いられ、通常のタブロー法のように ABox の変換規則適応や充足性の判定も行うことができる。

6. 関連研究と考察

一階述語論理のホーン節と記述論理との関係について、Groszof ら [Groszof 03] によって議論された。しかし、RuleML やクエリは意識されていても、記述論理プログラミング (DLP) においてどのように推論を進めるかについては、何も述べられていない。一方、記述論理とは離れて、推論効率の向上、プログラム型理論への応用などを目的に、これまでも多態順序ソート論理 (polymorphic order-sorted logic) [Beierle 92, Beierle 95] において型付き単一化が定式化されてきた。Beierle は我々と同様に新しい概念 $(C \sqcap D)$ を生成するような単一化の定式化をしている。一方、Paschke [Paschke 06] はセマンティックウェブの型システムに用いられる単一化について報告したが、我々の提案したような共通概念の生成は行っていない。これら二つの異なりは、応用目的の違いから来るとしてもよい。我々は、ウェブサービス合成分解問題において、ゴールを達成するようなウェブサービス実行列を動的に生成することを目的としている。二つの異なる概念に共通に所属可能な個物を発見できれば、その実行列でゴールを達成できるという意味で、この定式化は有効である。Beierle は Vehicle のサブクラスである Car と Ship の共通概念として水陸両用車を例示している。一方、プログラム型理論への応用として、コンパイル時に静的なタイプチェックを行おうとする場合には、本報告で述べたような単一化は望ましくないであろう。

実現において Paschke は論理システムとは別に記述論理の型推論用にたとえば Pellet を用いると言った外付けのシステムを提案しているが、我々の提案はタブロー法を単一化アルゴリズムと一体化し、記述論理プログラミングまで展望し得るも

*1 <http://aima.cs.berkeley.edu/lisp/doc/overview.html>

のである．例えば，我々はすでに SWCLOS と一体化可能な Prolog 処理系を有しているが，この単一化アルゴリズムを本報告のようにすることで，OWL による論理プログラミングが可能になるものと考えている．

一般に，セマンティックウェブでは名前の唯一性仮説 (Unique Name Assumption) に立っていない．すなわち，二つの個物の名前が異なるからと言って，その名前が指す解釈も異なるとは言えない．二つの個物が異なるということを言うためには明示的に owl:differentFrom 関係を宣言しなくてはならない．

$$a \doteq b \Rightarrow a^I = b^I \quad (\text{owl} : \text{sameAs})$$

$$a \not\equiv b \Rightarrow a^I \neq b^I \quad (\text{owl} : \text{differentFrom})$$

Baader ら [Baarder 03] によるタブロー法では，at-most (owl:maxCardinality) 制約において制限回数以上の個物集合であってかつ明示的に owl:differentFrom 関係の宣言が ABox がない場合には，ABox に単一化代入 a/b を追加している．すなわち，ないことを \neg で表して，次のようにしていると言ってよい． (explicit inequality assertion)

$$a \doteq b \Rightarrow a^I = b^I$$

$$a \not\equiv b \Rightarrow a^I \neq b^I$$

$$\neg a \equiv b \Rightarrow a^I = b^I$$

第 4.2 節における単一化では， \neq のときはもちろん，二つの個物が \doteq 関係に無いときにも最終的に単一化を失敗させた．これは次のようにしていると言ってよい． (explicit equality assertion)

$$a \doteq b \Rightarrow a^I = b^I$$

$$a \not\equiv b \Rightarrow a^I \neq b^I$$

$$\neg a \equiv b \Rightarrow a^I \neq b^I$$

これはもし二つの個物の名前が違っていても単一化を成功させると，これまでの型無しの単一化と較べて大きな違いが生じると考えたからである．John と Elizabeth は明示的に同じという宣言がない限り，異なるものするという意味で，これは弱い名前の唯一性仮説に立つものである．(本来のセマンティックウェブでは，明示的に owl:sameAs 宣言と owl:differentFrom が無かったら，両者の関係は不明である．)

7. まとめ

ウェブサービスの合成分解への応用と，記述論理プログラミングへの展望を動機として，OWL のクラスを型とする型付き単一化アルゴリズムを提案し，タブロー法と融合した実装について述べた．また，名前の唯一性問題についても考察した．ここで述べたように，セマンティックウェブでは開世界仮説に立ち，名前の唯一性仮説に立たないと言っても，実際の応用の場ではこれを徹底すると何も推論が進まないということが多い．開世界仮説に関する同種の議論は次の機会 [小出 08] に報告する予定である．現在，SWCLOS にタブロー法と本単一化を実装中であるが，完了次第，型付き Prolog への応用とウェブサービス合成分解問題への応用に取り組む予定である．

参考文献

[Baarder 03] Baarder, F. and Nutt, W.: *The Description Logic Handbook*, chapter 2. Basic Description Logics, pp. 43–95, Cambridge (2003)

[Beierle 92] Beierle, C.: Logic programming with typed unification and its realization on an abstract machine, *IBM Journal of Research and Development*, Vol. 36, No. 3, pp. 375–390 (1992)

[Beierle 95] Beierle, C.: Type Inferencing for Polymorphic Order-Sorted Logic Programs, in *International Conference on Logic Programming*, pp. 765–779 (1995)

[Grosz 03] Grosz, B. N., Horrocks, I., Volz, R., and Decker, S.: Description Logic Programs: Combining Logic Programs with Description Logic, in *12th International Conference on the World Wide Web (WWW-2003)*, ACM (2003)

[Kiczales 91] Kiczales, G., Rivière, des J., and Bobrow, D. G.: *The Art of the Metaobject Protocol*, MIT Press (1991)

[Koide 06] Koide, S. and Takeda, H.: OWL-Full Reasoning from an Object Oriented Perspective, in *Asian Semantic Web Conf., ASWC2006*, pp. 263–277, Springer (2006)

[Paschke 06] Paschke, A.: A Typed Hybrid Description Logic Programming Language with Polymorphic Order-Sorted DL-Typed Unification for Semantic Web Type Systems, in *Proc. of 2nd Int. Workshop on OWL: Experiences and Directions 2006 (OWLED'06) at ISWC'06, Athens, Georgia, USA* (2006)

[Russell 95] Russell, S. and Norvig, P.: *Artificial Intelligence: A Modern Approach*, Prentice Hall (1995)

[小出 08] 小出, 武田: OWL における明示的閉世界と局所閉世界仮説, 第 22 回人工知能学会全国大会 (2008)