Semantic Web の基礎

武田 英明

国立情報学研究所* / 東京大学 人工物工学研究センター†
e-mail:takeda@nii.ac.jp

1 はじめに

World Wide Web(WWW, Web) は情報交換の手段として広く普及し，もはや Web なしの世界をイメージするのが難しくなっている．Web のおかげでありとあらゆる情報が入手可能になった．その量は日々驚くほど増大しており，いまや情報洪水への対処が真剣に必要となっている．すなわち，Web は情報そのものの共有を可能にしたが，さらにもう一歩進んで情報の持つ意味の共有が必要とされている．現在，事実上唯一の Web 検索手段である google などの検索エンジンでは単語などの情報の外形的特徴でしか検索をすることができない．もし，情報の意味で検索できれば，より的確にほしい情報を検索できるであろう．これがセマンティック Web の狙いである．セマンティック Web は Web の創始者である Tim Berners-Lee が提唱している次世代の Web のフレームワークである．彼の記事 [1] によれば，

> セマンティック Web は現在の Web の拡張であり，そこでは情報には定義された意味が用意され，人と計算機の共同作業がより容易にできるようになる．
>
> (The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.)

といっている．あるいは W3C のセマンティック Web の活動ではもう少し人と計算機の共同作業の例を挙げて説明している[1]．

> セマンティック Web はビジョンである．そこでは Web 上のデータは，単に表示目的でなく，自動化や統合，アプリケーション間でのデータ再利用などに使えるように定義されてリンクされる．(The Semantic Web is a vision: the idea of having data on the web defined and linked in a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications.)

では具体的にどのように「意味」を計算機に理解させることができるのであろうか？計算機に「意味」を理解させるという試みは古くから人工知能の分野で研究されてきたが，決定的な成果があったわけではないとても難しい問題である．しかし，Web 上での情報共有の促進という観点だけでに注目すれば，問題を簡単することができる．セマンティック Web ではこの問題をメタデータと URI という二つの仕組みを軸に解決を図っている.「意味」の同一性の維持には URI(Uniform Resource Identifier) が利用される．どんな「意味」にも URI を割り付けることで，世界中のどんな人がどんなページでも同一の「意味」を指す事ができる．一方,「意味」の定義は構造化されたメタデータとして用意される．メタデータ記述

*千代田区一ツ橋 2-1-2
†柏市柏の葉 5-1-5

[1] http://www.w3.org/2001/sw/

言語（RDFS や OWL) によってここの「意味」は相互に関係付けられる．この関係性と同一性を元に計算機は上記のような，自動処理や統合処理，データ再利用をすることが可能になる．

以下では，このメタデータ記述言語である RDF, RDFS, OWL についてその概略を入門的に説明を行う．なお，セマンティック Web に関してのより詳しい情報は包括的が特集があるので参照されたい [2].

2　セマンティック Web の技術

セマンティック Web の鍵となる技術は，メタデータ，オントロジー，信頼であろう．

メタデータは文字通り,「データに関するデータ」を指すものであり，データの内容や，量，条件等データの特徴を記述するデータのことである．セマンティック Web での情報には人間あるいは計算機のために意味に関する情報がメタデータとして付加される．そうすることで，計算機は元々の情報だけは分かりえないその情報に関する情報を知ることができる．

しかし，メタデータを付加するのは第一歩でしかない．メタデータに記述される情報をどう計算機が容易に理解可能なものとして用意できるかが次の問題である．セマンティック Web ではメタデータ同士を構造化することでその解決を図っている．すなわち，構造的に概念間の関係を記述する．これがオントロジーである．

さらには信頼性を考慮する必要がある．それぞれの情報がどれだけ信頼に足るものであるかを判定しないと，統合的利用や自動処理は危険である．しかし，Web の世界における真偽というのは，論理における真偽と違って多様なものである．このような多様な信頼性の処理についてはまだあまり研究が進んでいないの現状である（[3] 参照）.

上のような階層を示したのがセマンティック Web の技術階層と呼ばれるものである（図 1).

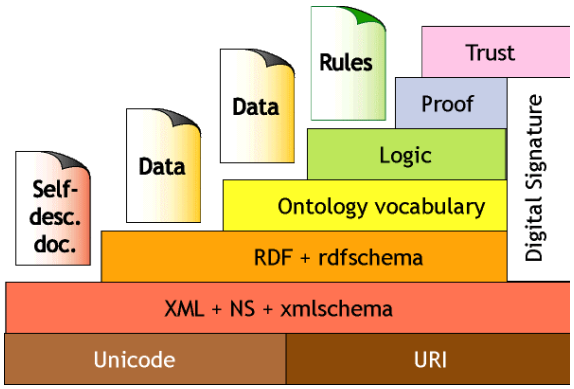


図 1: The Layer Cake for Semantic Web[4]

2.1　簡単な例

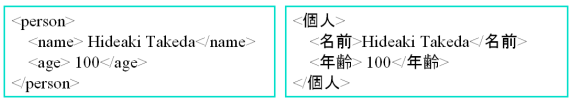
XML で書かれた簡単な例で考えてみよう（図 2(a)）．我々は簡単にこの情報が何を示しているかがわかる．名前が”Hideaki Takeda” で年齢が”100” である等々．しかし，これは我々がタグの名前（“person”, “name” 等々）をすでに解釈してしまっているからできることである．例えば，図 2(b) は日本人には (a) と同一の情報であることがわかるかも知れないが，日本語が分からない人には難しいであろう．

ここで必要なのは計算機に真の「意味」を理解させることではない．人間の解釈によって補っている曖昧性や相互利用性を上げることが出来れば十分である．例えば，

- 図 2(a) と (b) は同一であるか？
- これは “person” に関して十分な記述であるか？
- これは “person” に関して妥当な記述であるか？

などの質問に答えられることである．

しかし，これは XML では難しい．XML は本質的に構文を規定するためのものだからである．そこで意味を記述する言語が必要となる．



(a) (b)

図 2: A Simple Problem

2.2 RDF

RDF (Resource Description Framework)[2]は W3C の勧告 (recommendation) となっている Web メタデータ記述ための標準である [5]. RDF を使って，図書のメタデータのようにタイトルや著者といったメタデータを Web データに付加することができる．ただし，RDF は図書メタデータのように何をメタデータとして書くかを規定するものではなく，メタデータの書き方（記法）を規定するものである．RDF は極めて簡単なデータモデルである．すなわち，リソース，プロパティ，値の 3 つ組みである．

リソース（Resources） メタデータを付与する対象であり，URI (Uniform Resource Identifier)[6][3] またはリテラル（文字列）である．

プロパティ（Properties） リソースを記述すための特定の属性あるいは関係を示す．プロパティも URI またはリテラルである．またぷらパティもまたリソースになりうる．

値 (Value) プロパティが指し示すもので，リソースまたはリテラルである．

この 3 つ組みをステートメント (statement) と呼ぶ．ステートメントもまたリソースになりうる．

図 3 にある簡単な例を使って説明する．これは

> **http://www-kasm.nii.ac.jp/~takeda** のプロパティ **creator** の値は **Hideaki Takeda** である．

[2] http://www.w3.org/RDF/

[3] URI は resource を指すもので，URL(Uniform Resource Locator) と URN (Uniform Resource Name) からなる．

Resource (subject) http://www-kasm.nii.ac.jp/ takeda

Property (predicate) creator

Value (object) “Hideaki Takeda”

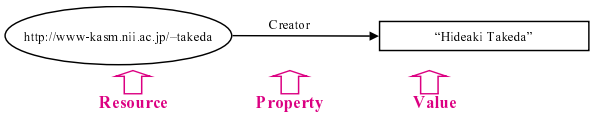


図 3: A RDF Model Example (1)

というステートメントである．このグラフが RDF の形式モデルである．とはいえ，グラフだと記述する上で不便なことが多いので，等価な文字列表現 `triple` を使うこともある．triple として表現すると，

```
<http://www-kasm.nii.ac.jp/~takeda>  creator  ‘‘Hideaki Takeda’’ .
```

となる．RDF triple は 3 つの要素と最後に “.” をつけたものである．

RDF は Web ページに限らず，下に挙げるような他のもののメタデータ記述にも使うことができる．

- ネットワーク上からアクセスできる文書やサービス
- ネットワーク上からアクセスできる実世界のオブジェクト
- “著者” といった抽象概念

唯一の制限は URI を持つことである．最後のケースからわかるように，プロパティも URI を持ちうるが，ただし必須ではない．

ステートメントを組み合わせて，もっと複雑なメタデータを書くことができる．

> **http://www-kasm.nii.ac.jp/ takeda** は プ ロ パ ティ **creator** の 値 が **http://www.nii.ac.jp/staffid/123456** で あ り， そ れ は ま た **Hideaki Takeda** をプロパティの **name** の値，

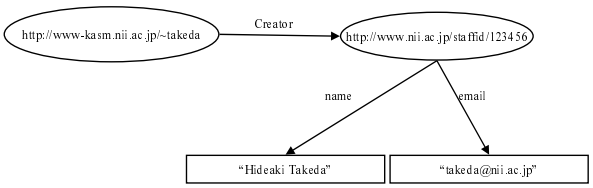


図 4: A RDF Model Example (2)

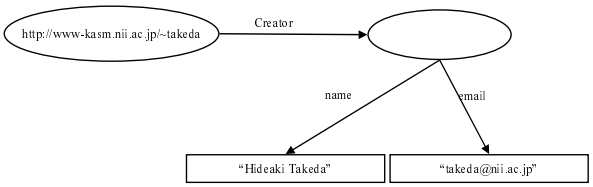


図 5: A RDF Model Example (3)

> **takeda@nii.ac.jp** をプロパティ **email** の値として持つ.

この場合，3 つのステートメントが用いられている．ここで注意が必要なのは，“http://www.nii.ac.jp/staffid/123456” は Web ページを指していないという点である．この URI の機能は人の識別のための ID としてのみ使われている．実際，具体的な情報はこの URI につけられたプロパティ “name” や “e-mail” の値として記述されている．

空白ノード (blank node) を使うともっと構造的なメタデータを書くことができる．図 5 は

> **http://www-kasm.nii.ac.jp/ takeda** はプロパティ **creator** ををもち，その値であるリソースは **Hideaki Takeda** を値とするプロパティ” name”, **takeda@nii.ac.jp** を値とするプロパティ”e-mail” を持つ.

RDF は 2 つの項しか結びつけることが出来ないので，3 つ以上を項を結び付けるには空白ノードが必要となる．

先に述べたように，RDF の形式モデルは図 3 のようなグラフである．RDF ではモデルとテキスト表現を分離している．テキスト表現では，先に見たような triple，あるいは XML を使って表現する．他には N3[7] という表現もある．

```
01 <?xml version="1.0"?>
02 <rdf:RDF
03   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
04   xmlns:dc="http://dublincore.org/2001/08/14/dces#">
05   <rdf:Description rdf:about="http://www-kasm.nii.ac.jp/~takeda">
06     <dc:creator>Hideaki Takeda</dc:creator>
07   </rdf:Description>
08 </rdf:RDF>
```

図 6: A RDF/XML Syntax Example (1)

```
01 <rdf:Description about="http://www-kasm.nii.ac.jp/~takeda">
02  <dc:creator rdf:resource="Hideaki Takeda" />
03 </rdf:Description>
```

図 7: A RDF/XML Syntax Example (1b)

図 6 は図 3 に対応する RDF/XML 構文による表現である．1 行目は XML の宣言部である．2 行目は RDF ステートメントの開始 (終了は`</rdf:RDF>`) を示している．`rdf:RDF` タグの属性として namespaces が宣言されている．プレフィックス `rdf:`で始まるタグは `http://www.w3.org/1999/02/22-rdf-syntax-ns#`namespace のタグであることを示している．同様に `dc:`は namespace`http://dublincore.org/2001/08/14/dces#` のタグを示している．5 行目から 7 行目が実際の RDF ステートメントの記述である．スタートタグ `rdf:Description` の属性 `rdf:about` がリソースを指示している．タグ `rdf:Description` で囲まれた中に，プロパティとその値が記述される．タグ `dc:creator` がプロパティを指定し，その値はタグ `dc:creator` で囲まれた部分である．

XML 構文では他の記法も許される．図 7 は同じ RDF を別の構文で書いたものである．図 6 の行 05 – 07 は図 7 の行 01 – 03 と同値である．今度は，タグ `dc:creator` は `rdf:resource` 属性を使って値を指示している．この方が短い表現である．

同様に，図 8[4]は図 4 の XML 表記である．

この例では，タグ rdf:Description がトップレベルであり，その内側 (行 04 – 07) に二つ要素があり，それぞれがプロパティとその値を指定している．

rdf:Description 要素の内側のノード http://www.nii.ac.jp/staffid/123456 を一緒にしてしまえば，図 9 のようなより短い表記にできる．この場合，タグ rdf:Description は入れ子になっている．入れ子表現はあまり理解しやすい表現ではないが，この表記には別の利点がある．この表記をつかえば，空白ノードを書くことができる．図 10 は図 5 の XML 表記である．前の例との違いは内側の rdf:Description タグの rdf:about 属性 (行 03) だけである．このような rdf:about 属性の省略は rdf:Description 要素が一意に決まるときのみ許される．

2.3 RDF Schema

RDF Schema (略して RDFS) は RDF をモデリングするために使われる．いわば RDF ステートメントの型を定義するようなものある．実際には，オブジェクト指向風なシンプルなクラス定義とクラス関係 (クラス-サブクラス関係) などが用意されている．

ただし，RDFS でのクラスはかなり原始的なもので，本格的なオブジェクト指向式のクラス概念は後に述べる OWL を使わないと実現できない．

RDFS ではまずクラスを定義する．以下の例では ex:namespace が予め用意されているとする．このとき，“Publication” がクラスであることを次のように書く．

ex:Publication	rdf:type	rdfs:Class	.

rdf:type は RDFS に予め用意されたある RDF のプロパティである．さらにクラス - サブクラス関係は次のように定義する．

[4]以降，簡略のため，タグ rdf:RDF の内側のみを表記する．

```
01 <rdf:Description about="http://www-kasm.nii.ac.jp/~takeda">
02   <dc:creator>http://www.nii.ac.jp/staffid/123456</dc:creator>
03 </rdf:Description>
04 <rdf:Description about="http://www.nii.ac.jp/staffid/123456">
05   <p:name>Hideaki Takeda</p:name>
06   <p:email>takeda@nii.ac.jp</p:email>
07 </rdf:Description>
```

図 8: A RDF/XML Syntax Example (2)

```
01 <rdf:Description about="http://www-kasm.nii.ac.jp/~takeda">
02   <dc:creator>
03     <rdf:Description resource = "http://www.nii.ac.jp/staffid/123456">
04       <p:name>Hideaki Takeda</p:name>
05       <p:email>takeda@nii.ac.jp</p:email>
06     </rdf:Description>
07   </dc:creator>
08 </rdf:Description>
```

図 9: A RDF/XML Syntax Example (2b)

```
01 <rdf:Description about="http://www-kasm.nii.ac.jp/~takeda">
02   <dc:creator>
03     <rdf:Description>
04       <p:name>Hideaki Takeda</p:name>
05       <p:email>takeda@nii.ac.jp</p:email>
06     </rdf:Description>
07   </dc:creator>
08 </rdf:Description>
```

図 10: A RDF/XML Syntax Example (3)

ex:Book	rdf:type	rdfs:Class	.
ex:Book	rdfs:subClassOf	ex:Publication	.

この場合，ex:Book というリソースのインスタンスは常に ex:Publication というリソースのインスタンスであることを示す．このようにクラスの階層関係をつくっていくことができる．

また，rdfs:subPropertyOf を使って，プロパティの階層関係を定義することができる．

次にクラスのインスタンスは次のように定義する．

ex:theTeXbook	rdf:type	ex:Book	.

クラスはよく属性やスロットと一緒に定義される．例えば,「本は人を値とする属性著者をもつ」は以下のように定義される．

ex:Person	rdf:type	rdfs:Class	.
ex:author	rdf:type	rdf:Property	.
ex:author	rdfs:range	ex:Person	.
ex:author	rdfs:domain	ex:Book	.

1 行目はクラス Person を宣言している．残りの行でプロパティauthor を定義している．2 行目で author がプロパティであることを宣言している．3 行目はこのプロパティの値がクラス Person であることを宣言している．最後の行はこのプロパティのリソースがクラス Book であることを宣言している．SVO モデルで言い換えれば，述語 author は Book を主語に，Person を対象に持つということである．結果として，クラス Book はプロパティauthor を属性として持ちうることになる．

RDFS も当然，RDF/XML 構文で書くことができる．図 11 はクラス定義の例である．RDF/XML 構文でかくとき，あるリソースが rdf:type プロパティをもつことを簡単に書くことができる．図 12 はその例である．

ここで用いた例の全体を図 13 に示す．Person と Book のインスタンスの宣言は rdf:type プロパティ

```
01 <rdf:Description rdf:ID="Publication">
02   <rdf:type rdf:resource="rdfs:Class"/>
03 </rdf:Description>
```

図 11: A RDF/XML Syntax Example (4)

```
01 <rdfs:Class rdf:ID="Publication"/>
```

図 12: A RDF/XML Syntax Example (4b)

の略記である．

知識表現として RDFS には注意が必要である．RDFS をオブジェクト指向言語あるいはフレーム言語としてみると，プロパティの部分で，随分異なっている．通常クラスの属性はクラスに従属するもの，すなわちクラスにローカルになるのに対して，RDFS のプロパティは常にグローバルである．例えば，先の例でも，author は一見クラス Book に従属しているようにみえるが，実は常にそうではない．author プロパティはその定義域がクラス Book，値域 Person に束縛されたグローバルなものである．この違いは，別のクラスで同じ author プロパティを使おうとするときにはっきりする．たとえば，クラス Software も author プロパティ使いたいとする．このときは同じ author プロパティrdf:range,rdf:domain を拡張することになる[5]．

2.4 OWL

RDFS はオブジェクト指向風の機能を提供しているが，オブジェクト指向言語としては決して十分ではない．OWL (Web Ontology Language)[8] はよりオブジェクト指向言語に近い語彙及び機能を提供している．クラス間の関係（排他関係等），個数制限（ ”1 個だけ ”等），等価関係，型宣言，プロパティのメタ属性（ ”対象 ”関係など），数え上げクラスなどが用意されている．OWL も W3C の勧告になっている．

[5] rdf:range や rdf:domain は 2 度以上用いてよい．

```
01 <?xml version="1.0"?>
02 <rdf:RDF
03   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
04   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
05
06 <rdfs:Class rdf:ID="Person"/>
07
08 <rdfs:Class rdf:ID="Publication"/>
09
10 <rdfs:Class rdf:ID="Book">
11   <rdfs:subClassOf rdf:resource="#Publication"/>
12 </rdfs:Class>
13
14 <rdf:Property rdf:ID="author">
15   <rdfs:domain rdf:resource="#Book"/>
16   <rdfs:range rdf:resource="#Person"/>
17 </rdf:Property>
18
19 <Person rdf:ID="donaldEKnuth"/>
20
21 <Book rdf:ID="theTeXBook">
22   <author rdf:resource="#donaldEKnith">
23 </Book>
24
25 </rdf:RDF>
```

図 13: A RDF/XML Syntax Example (5)

歴史的には OWL は DAML+OIL[9][10] の後継であり，これはさらに，米国 DARPA Agent Markup Language (DAML) プロジェクト[6]で開発された DAML-ONT と，主にヨーロッパで開発された OIL[11][7]を統合されたものである．OWL は記述論理（Description Logics）に基づいている．記述論理はフレームベースシステム，意味ネットワーク，KL-ONE 式の言語の形式化として利用されてきたものである．記述論理の詳細については [12] を参照されたい．

OWL は豊富な記述要素をもっているので，ここでは RDFS との比較で重要な点を中心に取り上げる．OWL については全体像は [13] を参照されたい．

まず，クラスを定義しよう．クラス定義 (class definition) はクラス記述から構成する．クラス定義は `rdfs:subClassOf`, `owl:equivalentClass`,`owl:dijointWith` を使う．`rdfs:subClassOf` を用いると，そのクラスのサブクラスとして新しいクラスを定義する．`owl:equivalentClass` は他のクラスと同値のクラスを定義する．`owl:dijointWith` は他のクラスの和として新しいクラスを定義する．1 番目は必要条件，2 番目は必要十分条件を定義している．

クラス記述 (class description) は次のいずれかである．

1. クラス識別子,
2. 個体の数え上げ,
3. プロパティ制約,
4. クラス記述の積 (intersection),
5. クラス記述の和 (union),
6. クラス記述の補集合 (complement)

重要なのは 3 番目で，これによってプロパティをクラスに従属させることができる．プロパティ制約は値制約，個数制約の 2 種類ある．前者は，`owl:allValuesFrom`, `owl:someValuesFrom`,`owl:hasValue` などを使い，それぞれ全ての値に対する制約，ある値に対する制約，特定の値であるという制約をつける．後者は，`owl:minCardinality`, `owl:maxCardinality`, `owl:cardinality` を使い，値の個数の最小値，最大値，特定の数に制約する．これらのプロパティ制約は `owl:Restriction` タグを使い，`rdf:resource` 属性で制約するリソースを指定する．

図 14 にその例を示す．クラス `Person` は 3 つのクラスのサブクラスの積として定義されている．つまり，このクラスは 3 つのどのクラスの多重継承である．一つ目のクラスはクラス `Animal` で，残りプロパティ制約によって作られた名前のないクラスである．一つは，`name` プロパティを少なくとも 1 つ持つというクラスであり，もう一つはプロパティ `eat` があり，その値がクラス `LivingThing` のインスタンスであるようなクラスである．この結果，このクラス `Person` はクラス `Animal` のサブクラスで，少なくとも名前を一つ持ち，生物を食べるものである，と定義される．この場合，プロパティ `eat` は別のクラス

[6] http://www.daml.org/
[7] http://www.ontoknowledge.org/oil/

```
01 <owl:Class rdf:ID="Person">
02   <rdfs:subClassOf rdf:resource="#Animal"/>
03   <rdfs:subClassOf>
04    <owl:Restriction>
05      <owl:onProperty>
06        <owl:DatatypeProperty rdf:about="#name"/>
07      </owl:onProperty>
08        <owl:minCardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
09          1
10        </owl:minCardinality>
11    </owl:Restriction>
12  </rdfs:subClassOf>
13  <rdfs:subClassOf>
14    <owl:Restriction>
15      <owl:onProperty>
16        <owl:ObjectProperty rdf:about="#eat"/>
17      </owl:onProperty>
18      <owl:allValuesFrom rdf:resource="#LivingThing" />
19    </owl:Restriction>
20  </rdfs:subClassOf>
21 </owl:Class>
```

図 14: An OWL Example (1)

記述で自由に使うことができる．この点が RDFS と違う点である．

OWL のプロパティは DatatypeProperty と ObjectProperty に 2 つに大別される．前者は RDF リテラルあるいは XML Schema type を値にとるものであり，後者はクラスのインスタンスを値にとるものである．プロパティは RDFS 同様，rdfs:range, rdfs:domain, rdfs:subPropertyOf で関係付けることができる．さらに，推移性や対称性といった関係の性質を指定することができる．同様なもので，FunctionalProperty は関数関係を宣言していて，値がユニークであることを指示している．

図 15 はプロパティ定義の例を示している．最初のプロパティ eat はクラス Thing が値域であるので，ObjectProperty である．2 番目のものは，整数が値域であるので，DatatypeProperty である．また，その値はユニークであることを要求している．

これらの定義を用いて，図 16 に示すようにクラス Person のインスタンスを定義できる．クラス定義より，クラス Person のインスタンスは一つ以上の name プロパティを持ち，eat プロパティの値が LivingThing クラスのインスタンスであるようなインスタンスである．さらにプロパティ age の定義より，その値は正の整数でユニークである．図 16 に示

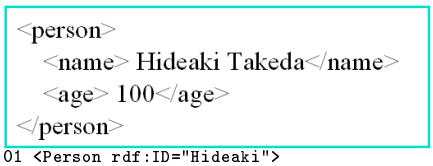
```
01 <owl:ObjectProperty rdf:ID="eat">
02   <rdfs:domain rdf:resource="#LivingThing"/>
03  <rdfs:range rdf:resource="&owl;Thing"/>
04 </owl:ObjectProperty>
05 <owl:DatatypeProperty rdf:ID="age">
06   <rdf:type rdf:resource="&owl;FunctionalProperty" />
07  <rdfs:domain rdf:resource="#Animal" />
08  <rdfs:range rdf:resource="&xsd;positiveInteger" />
09 </owl:DatatypeProperty>
```

図 15: An OWL Example (2)

すインスタンスは，上記の条件を満たしているので，妥当なものである．

さて，最初の問題（図 2) に戻って考えてみよう．図 16 は図 2 に相当するものである．この記述においては，クラス Peson に基づいて，この記述がクラス Person のインスタンスとして妥当なものであるかどうかをチェックできる．さらにある種の推論もすることができる．例えば，クラス Person はクラス Animal のサブクラスであるので，クラス Animal を定義域であるものはすべてクラス Person のインスタンスにも適用可能である．例えば，医者を探しているエージェントを考えてみよう．もし医者が見つからないときに，クラス階層を一つ遡って動物の医者を代わりに推薦することができる．

OWL には 3 つの言語，OWL Lite, OWL DL,




```
01 <Person rdf:ID="Hideaki">
02   <rdfs:label>Hideaki</rdfs:label>
03   <rdfs:comment>
04     Hideaki is a person.
05     His name is Hideaki Takeda.
06     His age is 100.
07   </rdfs:comment>
08  <name>Hideaki Takeda</name>
09  <age rdf:datatype="&xsd;positiveInteger">100</age>
10 </Person>
```

図 16: An OWL Example (3)

OWL Full に分けられている．ここまでの説明は基本的には OWL DL に基づいている．OWL DL は記述論理の表現力を基本的に持っている．ただし，このためは計算の複雑性が増大するため，より表現力を制限した ”軽い ”バージョンとして OWL Lite がある．OWL Lite はクラスの積のみであり，個数制限は 0 か 1 のみなどの制限がある．OWL Full は普通の RDF ステートメントを許容するものである．より表現力があるものの，形式意味論的には問題があり，現時点では実用的な利用には適さない．

3 今後の展開

今回紹介した RDF,RDFS,OWL といったセマンティック Web の記述言語開発は一段落しており，標準として徐々に普及を始めている．代わって，現在は 3 つの方面で研究が行われている．まず，これらの言語を実際に使うアプリケーションを作ろうというもので，様々なシステムが提案されている．顕著な方向としては，Semantic Desktop というものがある．ISWC2005 では同名のワークショップ[8]が開かれ 100 人近く集まり盛況であった．これは自分が使う PC の上で使えるアプリケーションにセマンティック Web 技術を適用しようというものである．セマンティック Web なので当然ネットワークアプリケーションであるわけで，セマンティック Web によるリッチクライアントという見方もできる．また，ルール言語の開発もホットトピックスの一つである．OWL は基本的に宣言的な記述したできないのに対して，ルール言語では動的な推論を可能とするものである．これも 2005 年に初の国際会議が開かれている[9]．Web Serice との統合も期待されている分野である．Semantic Web Services という名前で各方面で盛んに研究されている．

4 まとめ

本稿ではセマンティック Web の基礎技術である RDF, RDFS, OWL について入門的に紹介した．最後に述べたように，セマンティック Web は基本的な言語開発の段階を終え，第 2 段階に入っている．言語開発においては米国とヨーロッパの研究グループで開発が進んできた．言語開発などではなかなか後発のグループが関与することは難しいが，今後は応用面が注目されるので，日本の研究グループも多いに活躍可能だと思っているし，期待している．

参考文献

[1] Berners-Lee, T., Hendler, J. and Lassila, O.: The Semantic Web, *Scientific American* (2001).

[2] 徳田雄洋, 栗原聡：特集「セマンティック Web と計算機科学」, コンピュータソフトウエア, Vol. 22, No. 4, pp. 2–64 (2005).

[3] 武田英明：信頼の Web, コンピュータソフトウエア, Vol. 22, No. 4, pp. 19–25 (2005).

[4] Berners-Lee, T.: Semantic Web, Keynote Speech, XML 2000 (2000), http://www.w3.org/2000/Talks/1206-xml2k-tbl/.

[8] http://www.semanticdesktop.org/

[9] http://2005.ruleml.org/

[5] Lassila, O.: Web Metadata: A Matter of Semantics, *IEEE Internet Computing*, Vol. 2, No. 4, pp. 30–37 (1998).

[6] Berners-Lee, T., Fielding, R. and Masinter, L.: RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax, Technical report, IETF (1998).

[7] Berners-Lee, T. and Connolly, D.: Primer: Getting into RDF & Semantic Web using N3: http://www.w3.org/2000/10/swap/Primer.html.

[8] McGuinness, D. L. and Harmelen, van F.: OWL Web Ontology Language Overview, W3C Recommendation 10 February 2004: http://www.w3.org/TR/owl-features/.

[9] Connolly, D., Harmelen, van F., Horrocks, I., McGuinness, D. L., Patel-Schneider, P. F. and Stein, L. A.: DAML+OIL (March 2001) Reference Description, W3C Note 18 December 2001: http://www.w3.org/TR/daml+oil-reference.

[10] Horrocks, I.: DAML+OIL: a Description Logic for the Semantic Web, *IEEE Data Engineering Bulletin*, Vol. 25, No. 1, pp. 4–9 (2002).

[11] Fensel, D., Harmelen, van F., Horrocks, I., McGuinness, D. and Patel-Schneider, P.: OIL: An ontology infrastructure for the semantic web, *Intelligent Systems*, Vol. 16, No. 2, pp. 38–44 (2001).

[12] 兼岩憲:OWLの推論とその計算量, コンピュータソフトウエア, Vol. 22, No. 4, pp. 26–34 (2005).

[13] Dean, M. and Schreiber, G.: OWL Web Ontology Language Reference, W3C Recommendation 10 February 2004: http://www.w3.org/TR/owl-ref/.