このドキュメントは
Design Issues - Ideas about Web Architecture
http://www.w3.org/DesignIssues/InterpretationProperties.html
の和訳です。
このドキュメントには和訳上の誤りがありえます。
内容の保証はいたしかねますので、必ず W3C Web サイトの正式版ドキュメントを参照して下さい。

Tim Berners-Lee

Date: 1998, last change: $Date: 2001/01/19 03:30:18 $
Status: personal view only. Editing status: first draft.
Up to Design Issues

Web デザイン論

このページでは、play: と表記した架空の名前空間を使っている。これは文中で例を挙げるために使用しているだけで、読者はその仕様を想像できるものと考えている。

解釈プロパティ

概要：自然言語やエンコーディング、それにある抽象概念と他との間の類似関係については、RDF 上のプロパティとして最も良くモデル化されている。私はこれらを、ある値とその値を特定の方法で解釈した（または考え得る範囲で処理した）結果との関係を表す上で、「解釈プロパティ」と呼んでいる。

自然言語の注釈付けの問題

これまで（2000/02 まで）、RDF のコミュニティでは文字列として自然言語をどのように表記するかについて、常に混乱していた。XML ではこの問題は簡単だ。なぜなら、その文字列の意味について厳密に説明する必要はないからだ。一連のテキストをマークアップし、それはフランス語だと宣言すればいい。

> 彼の名前は <html:span xml:lang="fr"> Jean-Fran&ccedilla;ois</html:span> だが、
> 私は彼を Dan と呼んでいる。

標準規格になるべしと XML コミュニティからプレッシャーを受ける中で、RDF の仕様には「ある文字列がある言語で書かれている」ということを示す公式の手段として、このような属性を含めた。しかしこれは間違いだった。そのような属性についてはシンタックスに任せるべきであって、RDF の仕様が定義しているモデルに組み込むべきではなかったのだ。
識別子について述べたセクションで取り上げた例を考えてみよう。

```
<rdf:description>
    <rdf:type>http://www.people.org/types#person</a>
    <play:name>Ora Yrjo Uolevi Lassila</play:name>
    <play:mailbox resource="mailto:ora.lassila@research.nokia.com"/>
```

```
    <play:homePage resource="http://www.w3.org/People/Lassila"/>
</rdf:description>
```

この例は、RDF グラフにおいて５つのノードを表している：Ora という人自身（彼は Web アドレスを持たない）を示す無名ノードと、このリソースのタイプが人（person）であり、一般名、Ｅメールアドレス、所定のホームページを持つということを示す４つのアークだ。
言語のプロパティはどこに追加すべきだろうか？もちろん、言語属性をその XML 記述に追加できるだろうが、それは RDF モデルに翻訳すると失われてしまう：どんな３項関係で書いても無理だろう。

試み１：それは「人（person)」のプロパティなのか？

iCalendar（リンクにある私のメモを参照のこと）のような多くの仕様では、RDF の「人（person)」の定義に別のプロパティを追加している。

```
<rdf:description>
    <rdf:type>http://www.people.org/types#person</a>
    <play:name>Ora Yrjo Uolevi Lassila</play:name>
    <play:namelang>fi</play:namelang>
    <play:mailbox>ora.lassila@research.nokia.com</play:mailbox>
    <play:homePage>http://www.w3.org/People/Lassila/</play:homepage>
</rdf:description>
```

ここで、プロパティ play:namelang は「Aが自然言語Bで表記する名前を持っている」ことを意味するものと定義されている。iCalendar の仕様は定義がさらに複雑で、lang プロパティは、上記と同様の場合では名前の表記言語であり、別の場合ではオブジェクトの記述言語となる。これはモデリング上の混乱だ。このような方法を採る利点としては、構造をフラットに保つことができ、（email などの）RFC822 ヘッダのような XML 以前のシステムでも一応対処可能なシンタックスを持つことができる。
上記の例では、このような混乱を招く多くの欠点がある。Ora はそれぞれフィンランド語と英語の２つの名前を持っているかも知れず、そうすると iCalendar のモデルでは表現できないことになる。play:namelang の属性は明らかに「人」に結びつけられていて「名前」に付随してはいないため、そのような（２種類の言語による名前を持つという解釈を）自動的に行なうことは到底できない。この場合、構造が事実を全く反映していない。

試み２：それは「文字列」のプロパティなのか？

２つめの試みとして、言語を文字列自体のプロパティとして表現するグラフを作ってみる。"Ora Yrjo Uolevi Lassila" がフィンランド人だというのは間違いない？そう、Ora はフィンランド人だ。しかしそれとは話が違う。ここで言わなければならないのは、その文字列がフィンランド語だということだ。すると、問題は、RDF が文字列そのものを記述の対象にできないというところになる。心配しなくていい。実際、RDF はあるノードが正に文字列であることを示し、それ以外のことを表していないと言うことができる rdf:value プロパティを創り出している。これは中間ノードを導入することで可能になった。

```
<rdf:description>
    <rdf:type resource="http://www.people.org/types#person" />
    <play:name rdf:parseType="Resource">
```

```
        <rdf:value>Ora Yrjo Uolevi Lassila</rdf:value>
        <play:lang>fi</play:lang>
      </play:name>
    <play:mailbox resource="mailto:ora.lassila@research.nokia.com"/>
    <play:homePage resource="http://www.w3.org/People/Lassila">
</rdf:description>
```

上記はその例だ。RDF グラフの中で少なくとも見た目にはいい感じだ。実際、別のモデリング上のエラーが生じたという点を置いておけば、この案を使うことはできた。では言語がテキスト文字列のプロパティなのかと言うと、そうではない。つまり、文字列 "Tim" - は英語（Timothy の略称）なのか？それともフランス語（"Timothe" の略称）なのか？私は、ある言語か別の言語かを判断できるテキスト文字列の長いリストを加える必要はない。文字列自体は基本的に英語であるという仮定を持つシステムならば、単にその事実を表に出さないだけだろう。

試み３：記述対象と文字列との関係

実際、状況としては Ora の名前が自然言語のオブジェクトであり、それはフィンランド語の文字列 "Ora Yrjo Uolevi Lassila" に基づいた解釈だというところだ。言い換えれば、フィンランドの言語は Ora の名前とそれを表記した文字列との間の関係であり、２要素の関係をプロパティでモデル化しているわけだ。

```
<rdf:description>
    <rdf:type>http://www.people.org/types#person</a>
    <play:name>
        <lang:fi>Ora Yrjo Uolevi Lassila</lang:fi>
      </play:name>
    <play:mailbox>ora.lassila@research.nokia.com</play:mailbox>
    <play:homePage>http://www.w3.org/People/Lassila/</play:homepage>
</rdf:description>
```

これはかなり良さそうだ。Ora はフィンランド語の "Ora" という名前を持っている。これならば、RDF システムがその文字列に対するノードと、「Ora という人物」の概念から来る "フィンランド語" リンクを作ることができる。恐らくは「流通（currency）」の概念から来るデンマーク語リンクや、「重さ（1/15 ポンド）」の概念から来る古典英語リンクや、言うまでも無く「岸（shore）」の概念から来るラテン語リンクも作れるだろう。
ここで気になる問題は、そのような概念全てに対する ISO の仕様を参照できるように言語は文字列であってほしいのだが、もちろん lang: 空間の仕様がそれと同じ仕様を参照してはいけないということではない。
もう一つ気になる問題は、play:name の属性が必ず文字列になるということが理にかなっているのかという点だ。大抵の場合そこには文字列が来るだろう：貧弱なシステムが、文字列の代わりに自然言語が来る場合に備えて何を準備しておくべきか？私の考えでは、全ての文字列と全ての自然言語オブジェクトを含む１つのクラスを作ることが１番の方法だろう。そのような自然言語オブジェクトも認めないような文字列をどう使った場合にも、多言語対応ソフトウェアでははるかに難しくなる - これは深刻な問題だ。
[[これは RDF のパッケージングに関する別の興味深い疑問を想起させる。XML のパッケージングや email のパッケージングにおける要望で、恐らく RDF においても殆ど同じ様に望まれていることがある。それは、あなたが私に何らかのタイプ X について尋ねた場合、私はあなたに、尋ねられた X やその解釈に役立つ情報が含まれた何らかのタイプのパッケージを必ず提供できる、ということだ。しかしそれはまた別の話、@@@ プロパティやシンタックスをじっくり考えて定義するときの話だ。@@@]]

真に重要なことは、関連するリソースを使って人々を識別する場合であっても、全ての人が決まった URI を持っているという無理な仮定をせずに済んでいるように、我々は抽象的な概念について語るために RDF の能力を利用しているということだ。

解釈プロパティとしてのデータタイプ*

ここで言うデータタイプとは、プログラミング言語の原子タイプ（atomic types）の意味であり、例を挙げれば XML データタイプ（XML Schema part 2）のことである。データタイプの定義は、入力文字列における制約の定義（例えば有効な日付は正規表現で表すものと定義する）と、タイプのインスタンスが表す数学的な抽象固体の定義とを含む。その１つは表現と抽象値とプロパティを用いる文字列との間の関係をモデル化することができる。

```
<rdf:Description about="#myshoe">
    <shoe:size>10</shoe:size>
</rdf:Description>                                    <#myshoe> shoe:size "10".
```

この例は、"10" という値について何も説明していない。我々はこのようなタイプのモデルについていちいち考えずに毎日を過ごすことができる：靴のサイズはインチ単位の十進数であると分かっている。しかしそこには、設計者が作ったデータタイプについて多くの問題やトレードオフが存在する（例えば、

・全てに場合において、文字列表現から値の型を説明できるか？（例：厳密には 1.4e4？1.4d4？）
・異なるデータタイプの値を区別できるか？（例：1 = 1.0 か？）
・データタイプの集合は拡張可能か？（例：複素数や素数を追加できるか？）
・表現の特性が値のそれを決めるのか？
・値の特性が表現のそれを決めるのか？（利用可能な表現のみが正規の表現？）

セマンティック Web においてこれらの一般的な問題をモデル化できれば、任意のシステムにおけるデータのプロパティを記述する上で素晴らしいだろう。解釈プロパティを導入することによって、１つの文字列を十進数としての解釈や、単位を含む長さという解釈に導くことができる。問題は、内輪で大抵使っている RDF グラフが上に書いたようなものであるということだ。shoe:size が指す対象は "10" なのだ。
前述の「試み１」と全く同様にシステムを単純化するならば、shoe:size は整数のクラスだと宣言する。これは、あらゆる値が十進数の文字列であることを暗に示して（ここでは明言して）いる。そのような文字列とタイプが与えられれば、上記の例のような抽象的な値を整数の 10 であると結論づけることができる。これはうまくいく。その仕組みは、上記の問いの答えが、私が考える様に［No, Yes, Yes, Yes, No］となる XML データタイプで利用される。潜在的な問題としては、データタイプが分かるまでは２つの値を比較することが出来ないという点がある。
RDF でこのような表現を明確にモデル化するには、新たなノードやアークを導入する必要があるだろう。面倒なことだが。

```
<rdf:Description about="#myshoe">
    <shoe:size>
        <rdf:value>10</rdf:value>
    </shoe:size>                              <#myshoe> shoe:size [ rdf:value "10" ].
</rdf:Description>
```

こうすると、靴のサイズを "10" に結びつけるデータタイプの関係が存在することを表す rdf:value を定義できる。全てのデータタイプの関係はこの仕組みを用いた rdf:value のサブプロパティだ。これを形式化すれば、

データタイプの情報を RDF グラフに追加できる。ここで、そのオブジェクトが既定のクラスに属すのか、データタイプの関係が必ずある特定の関係に帰着するものと考えるかといった仮定を選択することができる。解釈プロパティを入れ子状にすることもできる - ある文字列を十進数と解釈し、それからフィーと単位の長さと解釈するように。しかしこれは、特別なノードを加えない限り不可能だ。全てのリテラルに対して新たな抽象ノードを導入するために、全ての RDF を３項関係に落とし込む方法を大幅に変更してしまうのか -- それは大変だ。"10" を一般的なリソース、即ち幾つかのデータタイプの関係において "10" が表すもの全ての集合に関する抽象概念だと宣言するのか。これもまた恐ろしい話で、もはやこれまで使ってきた意味で「等しい（equals）」という語は使えない。
元のアークに並べて余計なアークを加える代わりに、shoe:size のような全てのプロパティを、その shoe といくつかの文字列表現とのやや曖昧な関係として残しておき、機能プロパティ（rdf:actual など）を使って、shoe:size をタイプした抽象的な値を指す（より実用的な）プロパティに関連付けることはできる。

```
{ <#myshoe> shoe:size "10" } log:implies
{ <#myshoe> [is rdf:actual of shoe:size] [rdf:value "10"] } .
```

@@@ RDF/DAML(2001/1 の時点）では、データタイプの記述についての明らかな前進は全く見られない @@

更なる例

解釈プロパティという名前は、この種の用途に対して私が独断で選んだものだった。それが良い言葉かどうかは自身が無い。しかし、私はこれを使うよう勧めたい。Base64 エンコーディングはもう一つの例だ。どこにでも出てくるが、XML デジタル署名は１箇所にしかない。

```
<rdf:description>
    <play:name parseType="Resource">
        <lang:fi    parseType="Resource">
          <enc:base64>jksdfhher78f8e47fy87eysady87f7sea</enc:base64>
        </lang:fi>
     </play:name>
</rdf:description>
```

さらに別の例として、タイプの強制がある。何らかの日時の記述を以ってそれを date: として使いたい場合を考える。

```
<rdf:description>
    <play:event parseType="Resource">
         <play:start parseType="Resource">
             <play:date>2000-01-31 12:00ET</play:date>
         </play:start>
         <play:sumary>The Bryn Poeth Uchaf Folk festival</play:summary>
    </play:event>
</rdf:description>
```

このようなプロパティは、特殊で且つ／または曖昧性の無いプロパティであることが多い。例えば enc:base64

は明らかに可逆な変換だ。それには２つの文字列、一方は印字可能なものでもう一方は他に制約のないバイト文字列が関係する。バイト文字列は一般に XML ドキュメント内では記述できなかった。enc:base64 の定義は、A は base64 でエンコードすると A を生成するものである。この処理はプロセッサに依存せずに与えられた B から A を生成することができる。エンコーディングの名前空間（ここでは接頭辞 enc: が示している）の仕様上は、どこであろうと適切なプロセッサと使用可能な文字列であれば、その文字列の base64 エンコーディングを扱えることになっているかもしれない。
解釈プロパティは、そのあたりがどうなっているのかを明確にする。例えば、

```
<rdf:description about="http://www.w3.org/">
    <play:xml-cannonicalized parseType="Resource">
        <enc:hash-sha-1 parseType="Resource">
            <enc:base64>jd8734djr08347jyd4</enc:base64>
        </enc:hash-sha-1>
    </play:xml-cannonicalized>
</rdf:description>
```

上記は、様々な処理に対して全く個別に定義したプロパティを使い、W3C ホームページを正規化して SHA-1 ハッシュ値を取り base64 エンコーディングした値が jd8734djr08347jyd4 である、と明確に記述している。HTTP の場合のように、ヘッダは階層化できず、ボディはエンコーディングや圧縮その他が注釈として無秩序に書き込まれる状況と比較すれば、仕様として、どのようなことがどの順序で起きるかについて正しい結果を得る方法を与えなければならない。

結論

抽象的なものとして１つの文字列の解釈を表現することは、RDF プロパティを使って容易に為しえる。このことは、簡潔で精緻なモデルを作るために役立つ。しかし、RDF にデータタイプの概念を持ち込むと、今日我々が知っているような RDF とは互換性がなくなってしまう。