1A4-03

OWL 意味論とオブジェクト指向の視点

The Object Perspective for OWL Semantics

小出 誠二[*1]
Seiji Koide

[*1] 株式会社ギャラクシーエクスプレス
Galaxy Express Corporation

We have realized OWL processor SWCLOS on top of CLOS, in which each assertion involves the entailments as well as the checking of domain and range constraints with one by one input in the manner of interactive lisp. Whereas RDF entailment rules are clearly described thus far, OWL entailment rules are not clear. In this paper we address some of entailment rules in OWL in order to implement them into SWCLOS, and we demonstrate the results in dealing with Wine Ontology and a family ontology.

1. はじめに

CLOS(Common Lisp Object System)上で RDF や OWL オントロジー記述を処理できるプロセッサ SWCLOS[Koide2004]を開発している. SWCLOS では, リスプの対話機能を生かすために主張(assertion)を受け付けるたびに領域(domain)制約や値域(range)制約のチェック, RDF 伴意(entailment)ルールや OWL 伴意ルールによる推論結果の追加を行っている. たとえば, 三つ組みの入力が行われたとき, プレディケイトが未定義であれば即座にそれを rdf:Property の個体と定義する(伴意ルール **rdfs1**).

RDF の伴意ルールが RDF の意味論[Hayes2004]で詳細に記述されているのに対して, OWL における伴意ルールには不明な点が多い. Horst[Horst2004]は OWL の完全な伴意セットは知られていないと述べている. RDF 意味論と OWL 意味論[Patel-Schneider2004]が矛盾することはないが, SWCLOS 実装上 RDF 伴意ルールと同様に OWL における伴意ルールを可能なかぎり明確にしておく必要がある. 本論文ではワインオントロジーや家族オントロジーなどの例題を用いて OWL 意味論に従ったいくつかの伴意ルールについて述べ, SWCLOS におけるオブジェクト指向との関連について報告する.

2. OWL 意味論

RDF および OWL におけるクラスとは, ある個体集合(外延)のことであり, クラスの包摂関係は個体集合の包含関係にほかならない. すなわちクラス y がクラス x を包摂する($x \subseteq y$)とは, クラス x の個体はすべてクラス y の個体であることと同義である.

OWL における推論とは, クラス間の包摂関係を判定することと, 個体があるクラスに属するか否かを判定することにほかならない[兼岩 2005]. SWCLOS では OWL のクラスを CLOS のクラスに OWL 個体を CLOS インスタンスに写像した. こうすることにより, rdfs:subClassOf の推移律(transitivity)や rdf:type の包摂律は rdfs:Class のメタ巡回(rdfs:Class rdf:type rdfs:Class)をのぞいて CLOS により自動的に計算される. すなわち, クラス間の包摂関係の判定には Common Lisp 関数 cl:subtypep, 個体のクラス帰属判定には cl:typep を用いればよい. そして見かけ上 rdfs:Class メタ巡回を回避するために, SWCLOS では RDF の意味論に合わせた gx:subtypep と gx:typep を用意した.

連絡先:小出誠二, ㈱ギャラクシーエクスプレス, 港区浜松町1－18－16, Fax03-5733-7190, koide@galaxy-express.co.jp

OWL におけるクラス関係は, rdfs:subClassOf 以外でも規定される. SWCLOS では owl:intersectionOf や owl:unionOf が現れたとき, 以下に述べるように出現したクラス(無名クラスや制約クラスも含め)について CLOS におけるクラス上下関係を定義する.

2.1 上位／下位クラス関係

(1) owl:intersectionOf ($x = y1 \cap y2 \ldots \cap yn$)

もし x が y1, …, yn の intersectionOf ならば, x はクラス(領域制約から owl:Class)かつ y1, …, yn もクラス(owl:Class か owl:Restriction とされる, cf. OWL Web Ontology Language, Boolean class expressions)で x の個体集合は y1, …, yn の個体集合の共通集合である. 逆も真. また x の個体はすべて y1, …, yn の個体なので x は y1, …, yn の下位クラスとなる.

(2) owl:unionOf ($x = y1 \cup y2 \ldots \cup yn$)

もし x が y1, …, yn の unionOf ならば, x はクラス(領域制約から owl:Class)かつ y1, …, yn もクラス((1)と同様)で x の個体集合は y1, …, yn の合併集合である. 逆も真. 各 y の個体は必ず x の個体なので, 各 y は x の下位クラスとなる.

(3) owl:equivalentClass ($x = y$)

もし x と y が equivalentClass であれば, x はクラス(領域制約から owl:Class)かつ y もクラス(値域制約から owl:Class)で x のすべての個体は同時に y の個体であり, y のすべての個体は同時に x の個体でもある. 逆も真. x と y は互いに下位クラスの関係にある.

(4) owl:complementOf ($x = \neg y$)

もし x が y の complementOf であれば x と y はクラス(領域制約と値域制約から共に owl:Class)で x の個体集合は owl:Thing の個体集合から y の個体集合を除いたものになる. 逆も真. x と y は個体を共有しないから owl:disjointWith の関係にある.

定義式の右辺に後述のプロパティ制約を用いることでタキソノミー(外延の階層構造すなわちクラス階層構造)だけではない内包的な概念定義を行うことができる. たとえばワインオントロジー[1]では次のような記述がある.

[1] http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine.rdf

```
(owl:Class Wine
  (rdfs:subClassOf food:PotableLiquid)
  (rdfs:subClassOf
    (owl:Restriction
      (owl:onProperty hasMaker)
      (owl:cardinality 1)))
  (rdfs:subClassOf
    (owl:Restriction
      (owl:onProperty hasMaker)
      (owl:allValuesFrom Winery)))
  …
  )
```

RDF における大域的制約と同様に, このプロパティの局所的値域制約を用いることで伴意を実行できる. たとえば上記入力時にもし hasMaker が定義されていなければ, owl:onProperty の値域(大域的)は rdf:Property であるから hasMaker を rdf:Property の個体と伴意できるし, 上記知識入力後に以下の入力があったとき, たとえ Elyse が定義されていなくても, ElyseZinfandel は Zinfandel の個体であり Zinfandel が Wine であれば上記局所的値域制約により hasMaker の値はどのワインにおいても Winery なのでそれを Winery の個体と定義してよい.

```
(Zinfandel ElyseZinfandel
  (hasMaker Elyse))
```

どのような場合にどのように伴意できるか前述(1)～(4)において逆も真(if and only if, iff)に留意して, 伴意ルールを明確にする必要がある. ワインオントロジーでは白ワインを以下のように記述している.

```
(owl:Class WhiteWine
  (owl:intersectionOf
    (owl:Class Wine)
    (owl:Restriction
      (owl:onProperty hasColor)
      (owl:hasValue White))))
```

ここで owl:intersectionOf が使われていることで, 別の主張においてもし主題がワインの下位クラスで hasColor に上記と同様な記述があれば, それは WhiteWine の下位クラスと伴意できるし, hasColor に White を持つワインの個体定義であればそれは WhiteWine の個体であると伴意できる. もしこれが以下のように記述してあったら, ワインでかつ hasColor が White だからと言って WhiteWine とすることはできない(ワインでかつ hasColor に White を持つほかのワインがあるかも知れない).

```
(owl:Class WhiteWine
  (rdfs:subClassOf (owl:Class Wine))
  (rdfs:subClassOf
    (owl:Restriction
      (owl:onProperty hasColor)
      (owl:hasValue White))))
```

owl:intersectionOf に比べて owl:unionOf が出現することはまれである. owl:unionOf に関連する伴意については後述の owl:disjointWith を参照されたい.

owl:equivalentClass は主に二つの目的で使用される. 一つは別々のオントロジー記述された概念をあとから等価と定義するため(たとえば food:Wine は vin:Wine と等価)で, もう一つは概念の内包的な定義の使用法である. OWL ガイド[Smith2004]にある例では, TexasThings はプロパティ locatedIn に TexasRegion を持つものとしているが, owl:equivalentClass を使うことで逆に locatedIn に TexasRegion を持つものは(owl:Thing の個体)は無条件に TexasThings の個体であることを伴意できる.

```
(owl:Class TexasThings
  (owl:equivalentClass
    (owl:Restriction
      (owl:onProperty locatedIn)
      (owl:someValuesFrom TexasRegion))))
```

2.2 クラス／個体関係

(1) owl:disjointWith

値にクラスをとって, 主題となるクラスがこのクラスとは互いに素であること, すなわち互いに個体を共有することがないことを述べる. 逆も真. 部分的に個体を共有する二つのクラスは下位クラス関係でも互いに素でもない.

(2) owl:oneOf (x = {a1, …, an})

a1, …, an は個体であり, クラス x がかっきりこれらの個体をとることを述べている. 逆も真. すなわち, これ以外に個体がなければそれらを列挙できる. 一般的に RDF および OWL では複数の主張が単調増加的に加算可能であるが, owl:oneOf についてそうすると「逆も真」が意味をなさない. owl:oneOf の出現はある主題について一回だけとすべきであろう.

開世界を前提にしても unionOf と一緒に用いることで, 閉世界と同様な有用な計算が得られる. 次のように, もし x がフルーツと分かっていて, SweetFruit ではないことが知れれば, それは NonSweetFruit としてよい. 何故なら unionOf により, フルーツであるのは SweetFruit か NonSweetFruit かどちらかと定義されていて, 両者は互いに素と主張されているから.

```
(owl:Class NonSweetFruit)
  (rdfs:subClassOf EdibleThing)
  (owl:disjointWith SeetFruit))
(owl:Class Fruit
  (owl:unionOf
    (owl:Class SweetFruit)
    (owl:Class NonSweetFruit)))
```

owl:oneOf について OWL ガイドでは以下の例を示している. この表記では White, Rose, Red は直接的には WineColor の個体であるとは述べていないが, この表現だけでそのように伴意すべきである. また別の個体追加において, もしそれが WineColor の個体であって White でも Rose でもないことが分かればそれを Red と伴意できる. もしそれらが owl:AllDifferent とされていなければそのようには伴意できない.

```
(owl:Class WineColor
  (rdfs:subClassOf WineDescriptor)
  (owl:oneOf
    (owl:Thing White)
    (owl:Thing Rose)
    (owl:Thing Red)))
(owl:AllDifferent
  (owl:distinctMembers
    (vin:WineColor Red)
    (vin:WineColor White)
    (vin:WineColor Rose)))
```

2.3 個体／個体関係

OWL では, 名前(URI)が異なるから個体として異なるとは言えない. 異なる名前の個体について同一物とするための記述 owl:sameAs, あるいは同一物ではないとするための記述 owl:differentFrom がある. 宵の明星と明けの明星を同一物とするかどうか, またスーパーマンとクラークケントは同一物であるかどうかは OWL 記述の問題ではなく, オントロジー工学上の問題である. オントロジストはもし適当であると考えれば, いかようにも記述する.

2.4 局所プロパティ制約

OWL ではクラスに局所的なプロパティ制約を設けることができる. OWL 意味論[Patel-Schneider2004]では継承(inherit)という

言葉は出てこないが，上位クラスにつけられたプロパティ制約は当然その下位クラスにも適応される（ある下位クラスに所属するすべての個体はその上位クラスにも所属する，包摂律）ので，上位クラスに定義された局所的プロパティ制約は，オブジェクト指向と同様に，その下位クラス関係に継承されると言える．局所プロパティの役割は，タキソノミーに対して内包概念を付け加えることである．

OWL ではプロパティにカージナリティ制約や全称限量子(∀)や存在限量子(∃)つきの制約を課すことができる．一般に RDF では同一主題についてテキスト上に分散していくつでも書くことができるし，一つの主題についての RDF 表現中にいくつでも同一プロパティについて書くことができる．したがって，同一主題の同一プロパティについて複数の記述が現れたときは，知識単調増加の原則に従い，常に制限を厳しくする方向で制約の集約を行うこととする．

(1) カージナリティ制約 (>=nR, <=nR, nR)

プロパティ値の生起する数について以下のような制約がある．

- owl:cardinality:プロパティ値の生起の数を指定
- owl:minCardinality: プロパティ値の最小生起数を指定
- owl:maxCardinality: プロパティ値の最大生起数を指定

owl:minCardinality では最大値，owl:maxCardinality では最小値を取るように集約する．最小カージナリティ > 最大カージナリティとなった場合には充足不可能となる．

(2) owl:allValuesFrom (∀R.C)

プロパティ値のとるべきクラスを制約する．プロパティが存在しなくてもよいが，もし存在すればその値はすべて指定のクラスの個体でなければならない．rdfs:range 制約も含めて複数の値域制約を集積する場合には，MSCs（most specific concepts）すなわち最も特殊な概念セットを計算して結果とする．計算された MSCs 中に互いに素なペアが一つでもあれば充足不可能である．MSCs 中に複数のクラスがあればプロパティ値はそれらのすべての個体でなければならない（RDF では個体は複数クラスに所属できる）．この制約を受ける個体のクラスが不明であれば，それをその時点で MSCs の個体として定義してよい（あとから異なるクラスに所属する定義が出てくれば，その時点で MSCs を計算し，必要であればクラスに追加あるいは修正する）．

(3) owl:someValuesFrom (∃R.C)

プロパティ値のとるべきクラスを制約する．主題となる個体にそのプロパティは必ず存在しなければならず，プロパティ値の中には必ず最低1個は指定のクラス個体がなければならない．クラスを指定したくないときは owl:Thing を指定する．カージナリティ1と組み合わせて使われる場合には問題がない．それはある個体には必ずそのプロパティがあっていつでもどこでもその値が制約されることを意味する．したがって，プロパティ値に与えられた個体が未定義であれば，それをそのクラスの個体として定義してよい．

ところが，カージナリティ指定がない場合や1個以上の場合に問題が生じる．すなわち，プロパティ値に未定義な個体を発見しても，それをその制約クラスの個体とすることができないし，逆に制約を満足しないプロパティ値を見つけてもそれが制約に違反しているとも言えない．極端に言えばすべてのプロパティ値が制約を満足しなくても違反しているとは言えない．何故ならば，開世界を前提にして，どこかにその制約を満足するオントロジー記述があるかも知れないからである．もし，カージナリティ最大数と比較してこれ以上プロパティ値が有り得ない場合（これを close された状態と言うことにする）にのみ，それをその制約クラスの個体として定義することができる．

(4) owl:hasValue (R.{a1, …, an})

先の二つのプロパティ制約がクラスを取るのに対して，これは個体をとる．これはこの制約を持つ主題(個体)はこのプロパティに対してかっきりこの列挙のどれかを持つことを意味する．列挙の個数が1個の場合は，通常のオブジェクト指向におけるスロットのデフォールト値定義と似ているが，プロパティ値をあとから変更できないところが異なる．

3. OWL 処理系の実装

すべての OWL クラスは CLOS クラスとして OWL 個体は CLOS インスタンスとして実装され，owl:Class は rdfs:Class の下位クラスとして，owl:Thing は rdfs:Resource の下位クラスとして実装された．SWCLOS では RDF グラフノードは CLOS オブジェクトに相当し，RDF グラフエッジとその指し示す値がスロットに相当する．

3.1 充足可能性と複数クラス

概念の追加を行ったとき，開世界と知識の単調増加を前提として，充足可能性(Satisfiability)を調べなければならない．SWCLOS では同一クラスについて追加定義が可能である．システムは常に定義済みのクラスの上位クラスと新しく定義追加された上位クラスの集合について MSCs を導き，クラスの追加・修正が行われる．CLOS におけるクラスはインスタンスを生成するための鋳型であり，複数のクラス概念を許してはいない．そこで複数クラスの場合にはそれらを上位クラスとする特殊なクラス(shadow-class というメタクラスのインスタンス)を作り，実装上はそのインスタンスとしている．

3.2 局所プロパティ制約と継承

RDF では主題と対象の二項関係の集合をプロパティの外延と解釈する．rdfs:range による値域制約はあるプロパティの外延について主題を問わず適応されるが，OWL における局所プロパティの値域制約は，主題がそのクラスの個体であるような二項関係にのみ適応されるものとなる．これを CLOS に引き当てて考えると，CLOS 個体のクラスが所有するスロット定義がそのクラスにおける局所プロパティ制約に相当することがわかる．そこで CLOS の標準スロット定義用オブジェクトである standard-direct-slot-definition と standard-effective-slot-definition の下位クラスに Property-direct-slot-definition と Property-effective-slot-definition（ともに RDF 用）を定義し，さらにその下位クラスに OwlProperty-direct-slot-definition と OwlProperty-effective-slot-definition（ともに OWL 用）を定義して，OWL 用スロット定義には maxcardinality，mincardinality という二つのファセットを新たに設け，カージナリティ制約情報をこの二つのファセットに保存するようにした．一方，allValuesFrom や someValuesFrom の制約情報は標準スロット定義に存在する type ファセットに保存するようにした．(setf slot-value)などでスロット値を追加するときこのタイプ制約やカージナリティ制約に違反しないかチェックされる．

すべての OWL クラスは直接定義されたそのインスタンス情報を持っていて，ユーティリティとして準備された collect-direct-instances や collect-all-instances でそのクラスの直接インスタンスや下位クラスまで考慮したすべてのインスタンスを知ることができる．owl:oneOf による個体の列挙は，直接主題クラスのインスタンス情報として記録されたり，列挙を有する無名クラスが生成されて制約クラスとして記録されたりされる．hasValue 制約は

一個の場合はスロットデフォールト値に似ているのでスロット定義の initform として記述しておく. こうすることで, そのクラスの個体を生成するときには CLOS により自動的に制約値が個体のスロット値となる.

スロット定義を利用しての制約の継承は CLOS 標準の機構によって自動的に行われる. すなわち, 個体定義時にはそのクラスを包摂するすべてのクラスから direct-slot-definition が集められ, それらの集積が effective-slot-definition となり, 個体生成に使用される. したがって行うべき事は, 単調増加原則に従った制約の集約と充足可能性をチェックすることだけであり, CLOS のメタオブジェクトプロトコルがこれを可能にした. 個体生成時には effective-slot-definition に記述された制約に従って, 制約チェックや伴意の推論を実行するようにした.

3.3 各種伴意ルールの実現

CLOS メタオブジェクトプロトコルを用いて, 前述の伴意ルールをエンティティの定義時, 追加定義時, スロット値の追加時に実行するようにした. SWCLOS では RDF グラフノードが一つの CLOS オブジェクトとなるが, OWL 表記における (owl:Restriction …)や(owl:Class (owl:oneOf …))のような無名ノードも, オブジェクトとして生成される. 加えて, 保守用や推論用に設けられたいくつかのスロットにも必要な情報が格納され, 各種の推論計算に用いられる.

4. OWL 処理例

4.1 家族オントロジー

家族オントロジーを整然と上位クラスから下位クラスやインスタンスに順に定義することもできるが, ここではそうではなく伴意ルール実例を示すことに焦点を当てて述べる.

```
(defIndividual Female (rdf:type Gender)
  (owl:differentFrom Male))
```

- owl:differentFrom の値域制約により Male は owl:Thing 個体と伴意
- owl:differentFrom の領域制約により Female は owl:Thing 個体であるが, Gender が未定義であるためこれを owl:Thing と同じように owl:Class のインスタンスと伴意

```
(defResource Person (rdf:type owl:Class)
  (owl:intersectionOf
   Human
   (owl:Restriction (owl:onProperty hasGender)
                    (owl:cardinality 1))))
```

- owl:onProperty の値域制約により hasGender は rdf:Property のインスタンスと伴意
- owl:intersectionOf により Human は owl:Class のインスタンスと伴意

```
(defResource Woman (rdf:type owl:Class)
  (owl:intersectionOf
   Person
   (owl:Restriction (owl:onProperty hasGender)
                    (owl:hasValue Female))))
(defIndividual 皇后美智子 (rdf:type Person)
  (hasGender Female))
```

- 皇后美智子は owl:intersectionOf の各要素を満たすので, すなわち Person かつ hasGender に Female を持つので (gx:typep 皇后美智子 Woman)となり, Woman のインスタンスと伴意

```
(defResource Parent (rdf:type owl:Class)
  (owl:equivalentClass
   (owl:Class
    (owl:intersectionOf
     Person
     (owl:Restriction (owl:onProperty hasChild)
                      (owl:someValuesFrom Person))
     (owl:Restriction (owl:onProperty hasChild)
                      (owl:allValuesFrom Person))))))
```

ここで Parent 定義における下線部を注意しておく. Parent の通常の定義では下線部がない[Baader2003]. しかし, ある親(個体)の hasChild プロパティの値が未知だったとき, 下線部なしではその値を Person であると伴意できない. 伴意機能を有効に活用するためには下線部が必要である. 一方, hasGender プロパティ制約は Woman→Person の包摂関係から継承されるが, ここにカージナリティ1が記述されているため, Woman の持つ Gender は無条件に Female と伴意できる.

5. おわりに

セマンティックウェブプロセッサ SWCLOS の OWL 処理機能についてその一部を報告した. SWCLOS の実装には Allegro Common Lisp を用いているが, Allegro Prolog は ACL に統合され CLOS オブジェクトスロット値を Prolog 変数とすることができるため, Allegro Prolog を用いてクエリ機能を容易にプログラムすることができる.

謝辞

本報告は, 文科省ITプログラム「ITを活用した大規模システムの運用支援システムの構築」の一部として実施されたものである. 本プロジェクト実施では大須賀節雄東京大学名誉教授に技術評価委員長をお願いし, オントロジー構築に関して大阪大学溝口理一郎教授にご協力戴いている. 家族オントロジーについて溝口教授に議論いただいた. 国立情報学研究所兼岩博士からは[兼岩 2005]とともに OWL 意味論について議論を戴いた. 記して感謝の意を表する.

参考文献

[Baader2003] Baader, F. and W. Nutt, Basic Description Logics, The Description Logic Handbook (ed. Baader et al.), 43-95, Cambridge Univ. Press, 2003.

[Hayes2004] Hayes, Patric: RDF Semantics , W3C Recommendation , http://www.w3.org/TR/rdf-mt/, W3C, 2004.

[Haarslev2004] Haarslev, Volker and Ralf Moeller, RACER User's Guide and Reference Manual, Univ. Hamburg, 2004.

[Horst2004] Horst , Herman J. ter, "Extending the RDFS Entailment Lemma", The Semantic Web – ISWC2004, 79-91, 2004.

[兼岩 2005]兼岩憲, OWL の推論とその計算量, コンピュータソフトウェア, 2005 年 10 月発行予定, 2005.

[小出 2004] 小出ほか, リスプによる RDFS プロセッサ, 第 18 回人工知能学会全国大会, 3H3-03, 2004.

[Koide 2004] Koide, S. and M. Kawamura, "SWCLOS: A Semantic Web Processor on Common Lisp Object System", http://iswc2004.semanticweb.org/demos/32/, 2004.

[Patel-Schneider2004] Patel-Schneider, P. F., Patrick Hayes, and Ian Horrocks: OWL Web Ontology Language Semantics and Abstract Syntax, http://www.w3.org/TR/owl-semantics/, W3C, 2004.

[Smith2004] Smith, Michael K., Chris Welty, and Deborah L. McGuinness, OWL Web Ontology Language Guide, http://www.w3.org/TR/owl-guide/, W3C, 2004.