TM-1142

知識ベース・知識表現言語 Quixote

横田　一正、安川　秀樹、高橋　千恵、
西岡　利博、平井　千秋、森田　幸伯（沖）

December, 1991

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191 ~5
Telex ICOT J32964

Institute for New Generation Computer Technology

知識ベース・知識表現言語 QUIXOTE*

横田一正、安川秀樹 ((財) 新世代コンピュータ技術開発機構)
高橋千恵 ((財) 日本情報処理開発協会)、西岡利博 ((株) 三菱総合研究所)
平井千秋 ((株) 日立製作所)、森田幸伯 (沖電気工業 (株))

1991 年 12 月 2 日

目次

*本論文は 1990 年度成果報告書を元に加筆修正を行ったものである。

1 概要

Quixote は、知識ベース管理基本プログラム Kappa の上位に位置づけられ、知識ベース管理システムの中核の役割を果たす知識ベース言語である。これはまた、遺伝子情報処理、法的推論、あるいは自然言語処理など、さまざまな知識情報処理の知識を表現するための知識表現言語でもある。

Quixote の設計方針としては以下の 2 点を考慮した:

- 知識情報処理で必要とされる演繹、制約解消などの推論をサポートするために、(制約) 論理型言語の枠組を基礎に置く。
- さまざまな知識情報処理が対象とする、複合構造や部分情報などの複雑なデータや知識を効率的に表現する能力を持つ。

これらを実現するために、オブジェクト指向概念を持った演繹データベース (DDB)、すなわち演繹・オブジェクト指向データベース (DOOD) の枠組 [Yokota and Nishio 89] を基礎に置いた。DOOD は、データモデルの観点からは、オブジェクト識別性の導入、複合オブジェクトの導入、およびカプセル化が考えられ、計算モデルの観点からは、オブジェクト指向パラダイムや制約論理型パラダイムが重要である。これらの拡張の方向はまた、自然言語の意味表現における素性構造、パラメータ、状況などの概念とも対応し、より包括的な知識表現の可能性を持っている。さらに DDB で研究開発されている問合せ処理は、*Quixote* で記述された大量の知識を格納した知識ベースに対する効率的な問合せ処理の可能性も与えてくれる。

まず知識オブジェクトを特定するためオブジェクト識別性として、複合オブジェクトを基礎に置いた。これは、ルールによって動的に生成されるオブジェクトを特定するために、従来のオブジェクト識別性概念を大幅に拡張したものである [Morita, et al 90]。そして意味論としては、P. Aczel によって提案された集合論 ZFC$^-$/AFA [Aczel 88] のラベル付きグラフとした [Yasukawa and Yokota 90]。

知識オブジェクトのもつさまざまなプロパティは、オブジェクト識別性に付属する属性として表現されており、属性名 (ラベル) はそれらをラベル付きグラフの領域上の制約に対応させている。この制約解消系は、J. Barwize [Barwise 89] と向井 [Mukai 90(1), Mukai 90(2)] の結果に基づいている。またこの属性は、オブジェクト識別子間の順序関係に従って下位オブジェクトに継承される。この継承については、多重継承、例外継承を扱うこともできる。

このオブジェクトに基づいて、ルール、プログラム、およびデータベースが定義される。この定義の中には、モジュール概念が導入されている。これは、知識のモジュール化、ルール継承、などを可能とするもので、大規模知識ベースや仮説的推論を必要とする知識情報処理に適している。

Quixote は、以上のような特徴を持った知識ベース言語・知識表現言語として研究開発されている。以下 2 節で、*Quixote* 第 1 版の言語仕様を述べ、3 節でその意味論、4 節で今後の課題を述べる。

2 言語仕様

本節では *Quixote* の言語仕様について述べる。

Quixote は、オブジェクト指向概念、演繹データベース、制約論理型言語を取り入れた知識ベース／知識表現言語である。

Quixote では、データや知識などを "オブジェクト" という概念の単位で記述する。*Quixote* では、いわゆるクラスもオブジェクトとして扱う。したがって、概念を表現する場合もそれをオブジェクトとして表現することができる。各オブジェクトを特定するために、オブジェクト項が用いられる。オブジェクト項で表現された各オブジェクトに対して、その性質を表現するものとして属性 (プロパティ) 情報が与えられる。属性情報は、属性項の形で記述される。また、各オブジェクトに対して汎化関係が与えられる。たとえば人間は動物であるといったいわゆる IS_A 関係が表現できる。汎化階層はオブジェクト項の包摂関係と対応つけられる。属性情報は、汎化階層に従って継承される。

さらに、属性項を要素として規則の形で、知識を表現する。これらの知識は、モジュールとよばれる単位ごとに記述される。モジュールは、状況、視点、時間、世界などを表現するのに用いることができる。モジュールが異なれば、同じオブジェクト項で表現されるオブジェクトの属性情報が異なっても良い。さらにモジュール間に順序を与え

ることにより、モジュール間での規則の継承を行なう。これによりひとまりの知識をモジュールの形でまとめておき、他のモジュールからそのモジュールを引用することができる。

これらのデータベースに対して、問合せは基本的には変数を含む属性項の形で行なわれるが、モジュール間の順序や、一時的に利用する規則などを同時に記述することができる。

2.1 *Quixote* のオブジェクト

Quixote では知識表現言語の基本要素として拡張項を採用している。これは、Prolog などでいう項を拡張したものである。*Quixote* では、オブジェクトを識別する識別子として、オブジェクト項と呼ばれる拡張項を用いる。以下、この節ではオブジェクト項について説明する。

原始オブジェクト (**primitive objects**) の有限集合 BO と束 $BO^* = (BO, \preceq, \top, \bot)$ を仮定する。任意の $a, b \in BO$ について、$a \preceq b$ は、b が a を包摂する、すなわち、a は b であることを意味する。$\top$ および $\bot$ は、それぞれ、BO^* の上限および下限である。また、$a \sqcap b$ および $a \sqcup b$ を、それぞれ、$\{a, b\}$ の下限および上限とする。BO^* の要素を**基本オブジェクト** (**basic objects**) という。

$BO^* = (\{animal, mammal, human, dog\}, \preceq, \top, \bot)$ は束の例で、下記の包摂関係が成り立つ。

$$mammal \preceq animal$$
$$human \preceq mammal$$
$$dog \preceq mammal$$

また、(原子) **ラベル** ((atomic) **labels**) の有限集合 L を仮定する。*Quixote* のオブジェクト (**基礎オブジェクト項** (ground **object terms**) と呼ばれる) を次のように定義する。

o を基本オブジェクト、$l_1, l_2, \cdots$ を互いに異なったラベル、$o_1, o_2, \cdots$ をオブジェクト項とする。

- どの基本オブジェクトもオブジェクト項である。
- 項 $o[l_1 = o_1, l_2 = o_2, \cdots]$ は、それはオブジェクト項である。
- 項が上の定義によりオブジェクト項であると示されるときに限って、その項はオブジェクト項である。

オブジェクト項 $o[l_1 = o_1, l_2 = o_2, \cdots]$ において、o は頭部、o_i はそのオブジェクト項の l_i-値である、という。頭部は、$\top$ であるときに限って省略することができる。

$BO = \{human, 20, 30, int, male, female\}$, $20 \preceq int$, $30 \preceq int$, $L = \{age, sex\}$ とする。そのとき、下記の項は *Quixote* におけるオブジェクト項である。

$$human.$$
$$human[age = 20, sex = male],$$
$$[age = 20].$$

最初のオブジェクト項は *human* という概念またはクラスを表し、2 番目のオブジェクト項は *20-aged male human* という概念を表わしている。どちらも *human* の種類を表わしているが、互いに異なる *human* の種類を表わしている。オブジェクト項は概念を表わし、オブジェクト項の中のラベル値の記述は、ある一定の概念を示すために十分な 性質 を表わしているということを認識すべきである。別の言い方をすると、オブジェクト項の中のラベル値の記述は、他の概念から、今表現しようとしている概念を区別するために使うキーの性質を表わす。オブジェクト項は、その中で、ラベル値の記述以外にもっと別な性質を表わしたいかもしれない。たとえば、$human[age = 20, sex = male]$ について、$[weight = 60kg]$ のような性質を持たせたいかも知れない。このような種類の性質をオブジェクト項の 属性 (*attributes*) という。属性については、さらに詳しい議論を 2.2.2 節で行う。

下記のような項は、オブジェクト項ではない。

$$human[age = 20][sex = male],$$
$$human[age = 20, age = 30].$$

オブジェクト項は、ノードの値が基本オブジェクトで、アークがラベルで表わされるようなグラフによって一意的に示すことができる。たとえば、オブジェクト項 $human[age = 20]$ は、ノードの集合 $\{n_1, n_2\}$ とアークの集合 $\{n_1 \xrightarrow{age} n_2\}$ からなるグラフによって表現される。ここで、ノード n_1 および n_2 の値 $n_1\downarrow$ および $n_2\downarrow$ は、それぞれ $human$ および 20 である。3.1節では、グラフのモデルを超集合の上で定義する。

下記のように、基礎オブジェクト項を値域とする変数を含むオブジェクト項も記述可能である。

$$human[age = X, sex = Y].$$

$X[age = 20]$ は、変数が基礎オブジェクト項を値域としないため、オブジェクト項ではない。

χ を変数の全体とし、T を基礎オブジェクト項の全体、$T[\chi]$ を変数を含むことが可能なオブジェクト項の全体であるとする。

包摂関係は、$T \times T$ 上の 2 項関係である ($\sqsubseteq$ と書く)。包摂関係の厳密な定義は、3.2節で与えるが、$\sqsubseteq$-関係の直観的な理解はここの説明で十分である。直観的に、$o_1 \sqsubseteq o_2$ (o_2 が o_1 を包摂する) は、o_2 より o_1 の方がアークが多く、$\preceq$-順序に関して、o_2 のノードの値が、そのノードに対応する o_1 のノードの値より大きければ、成り立つ。

たとえば、オブジェクト項

$$animal と human[age = 20, sex = male]$$

を比較すると、後者が前者よりアークを多く持ち、また、前者のルートノードの値 ($animal$) の方が後者のルートノードの値 ($human$) より大きい。すなわち、$human \preceq animal$ が成り立つので、

$$human[age = 20, sex = male] \sqsubseteq animal$$

となる。同様に、

$$human[age = 20] \sqsubseteq animal[age = int]$$

が成り立つ。しかし、

$$human[age = 20] と human[sex = male]$$

については、$\sqsubseteq$ に関して比較することができない。さらに、オブジェクト項 $\top$ は、すべてのオブジェクト項の中で最大である。

合同関係 (congruence relation) ($\cong$ と書く) を以下のように定義する。

$$o_1 \cong o_2 \stackrel{\text{def}}{=} o_1 \sqsubseteq o_2 \wedge o_2 \sqsubseteq o_1$$

オブジェクト項の全体が $\sqsubseteq$ によって半順序で関係づけられたとき、$\cong$ は相等 (equality) ということを表わす。

u, v をオブジェクト項とする。**原子制約 (atomic constraint)** とは、下記の形式のいずれかであるリテラルである。

$$u \sqsubseteq v$$
$$u \cong v.$$

一般性を失うことなく、少なくとも u あるいは v は基本オブジェクトあるいは変数をとると仮定できる。制約とは原子制約の集合である。制約は、その要素の連言、すなわち、原子制約の連言と考えられる。したがって、ここでの制約言語は、連言だけが論理結合子として許される限量子なしの一階言語の部分言語である。

制約を使うことによって、T の部分集合を値域とする変数を含むようにオブジェクトを拡張することができる。たとえば、下記は、*QUIXOTE* のオブジェクト項である[1]

$$human[affiliation = X] \mid \{X \sqsubseteq company\},$$
$$human[hobby = X] \mid \{tennis \sqsubseteq X\},$$
$$human[age = X] \mid \{X \cong 20\}.$$

[1] 2 番目のオブジェクトの $hobby$-値には集合が使えるべきである。実際、*QUIXOTE* では属性値として集合を導入しており、2.2.1節で説明される。

最初のオブジェクトは、勤務先 ($affiliation$) が会社 ($company$) の種類であるような人 ($human$) という概念を表わしている。最後のオブジェクトは、$human[age = 20]$ と同じである。"ある人の使用人がその人自身であるような人" のように自己参照のオブジェクトを表わしたい場合がある。このようなオブジェクトは、下記のような制約を用いることによって定義することができる。

$$X \mid \{X \cong person[employee = X]\}.$$

下記は、$QUIXOTE$ における syntax-sugaring の一覧である。

$$\begin{aligned}
o[l \doteq o'[\cdots]] &\Leftrightarrow o[l = X] \mid \{X \cong o'[\cdots]\},\\
o[l \rightarrow o'[\cdots]] &\Leftrightarrow o[l = X] \mid \{X \sqsubseteq o'[\cdots]\},\\
o[l \twoheadrightarrow o'[\cdots]] &\Leftrightarrow o[l = X] \mid \{X \sqsupset o'[\cdots]\},\\
o[l = X@o'[\cdots]] &\Leftrightarrow o[l = Y] \mid \{Y \cong X, X \cong o'[\cdots]\},\\
o[l \rightarrow X@o'[\cdots]] &\Leftrightarrow o[l = Y] \mid \{Y \sqsubseteq X, X \cong o'[\cdots]\},\\
o[l \twoheadrightarrow X@o'[\cdots]] &\Leftrightarrow o[l = Y] \mid \{Y \sqsupset X, X \cong o'[\cdots]\}.
\end{aligned}$$

また、"ある人の使用人がその人自身であるような人" のように循環構造を持ったオブジェクト項は、$QUIXOTE$ では、$X@person[employee = X]$ のように表わすことができる。この変数を、**注釈変数 (annotated variable)** という。自己参照のオブジェクト項があるために、循環性を扱わねばならない。これが、$QUIXOTE$ における意味論の定義域として、Aczel の超集合論を採用した理由である。超集合論は、循環事象を扱うための通常の集合論の技術をすべて提供してくれる。

ここでの、オブジェクトの概念化の方法は、オブジェクト項をオブジェクトを表わすための識別子 (**オブジェクト識別子**) としてみることである。例えば、$human[age = 20, sex = male]$ は、"20-$aged\ male\ human$" という概念を一意に示す項であって、$human$ というものが、20 $aged$ で、かつ $male$ であるということを意味しているわけではない。$human[age = 20, sex = male]$ という項 (すなわちオブジェクト) は、age と sex 以外に、$name$ や $occupation$ のようなラベルを持つことはできるが、そのようなラベルは、オブジェクトの概念化のために行なう定義に対しては本質的なものではない。たとえば、下記の記述 (これは属性項と呼ばれる) は、$QUIXOTE$ では、"20-$aged\ male\ human$" は、(デフォルトとして) 結婚しているという事実を表わすために使う。

$$human[age = 20, sex = male]/[married = yes].$$

これを属性項と呼ぶが、これについては 2.2節で述べる。

ここでは変数を含むオブジェクト項の定義域 $T[\setminus]$ およびその上での制約に関して考察を行っているので、オブジェクトの識別は $T[\setminus]$ 上の同値関係によって定義される。したがって必要なことは、制約を解くということ、または、少なくともその可解性を調べるということである。Barwise[Barwise 89] は、**シミュレーション・ペア (simulation pair)** の存在、すなわち超集合 (hypersets) 間において同値および包摂の集合が存在することを用いて、変数が具象化される場合の超集合の上での制約の可解性に関して重要な結果を示した。Mukai[Mukai 90(2)][Mukai 90(1)] は、Barwise の結果を選言と否定を含む制約の場合に拡張し、Aczel の超集合論のサブクラスが、Jaffar と Lassez [Jaffar and Lassez 87] の提案による CLP-schema の基準を満たすことを示した。3.3節では、制約言語に関する可解性の条件について、超集合の上での制約言語に関する Mukai の結果を基礎に議論する。

オブジェクト項の構文は図 1のように規定される。ここで、表内で (⋯) 内には、計算機上の記法を表現している。たとえば、"—" は、'-', '>' の 2 文字で表現する。また、"ATOM" は、文字列で表現されるアトムを表している。たとえば、変数 (<var>) は、大文字から始まる文字列で表現される。図中、属性名を表すラベル (<lab>) は実装上の理由から、文字列 "l_" で始まるアトムもしくは、文字列 "l_" で始まるアトムをヘッドにもつオブジェクト項で表現するものとしている。また、同様にモジュール名 (<m-id>) についても文字列 "m_" で始まるオブジェクト項で表現するものとする[2]。表現 (expression) (<exp>) は、利用者の便宜のために、複雑なオブジェクト項の略記を定義するためのものである。利用者は、文字列 ''e_'' で始まる任意のアトムに、適当なオブジェクト項を与えることができる。

[2]ただし、この制限は実装上の便宜のためであるので、本文中ではこれに従わない。

```
<o-term>          ::=  <comp-o-term>
                     | <g-var>
                     | <exp-name>
                     | <dot-term>
<g-var>           ::=  <u-string> | "_"<string>
<exp-name>        ::=  "e_"<string>
<dot-term>        ::=  <o-term>"!"<lab>
<self>            ::=  "&self"

<comp-o-term>     ::=  <o-head> ["["<o-attr>-list(",")"]" ["|""{"<o-cnstr>-list(",")"}"]]
                     | <a-o-term>
<o-head>          ::=  <basic-obj>
<o-attr>          ::=  <lab> <attr-op> <o-term>        % 1 版は<o-term>に<dot-term>は許さない
                     | <lab> <attr-op> <m-id>
<lab>             ::=  "1_"<comp-o-term>               % 1 版は"1_"<basic-obj> のみ
<attr-op>         ::=  "⟶" ("->") | "⟵" ("<-") | "=" ("=")
<a-o-term>        ::=  <l-var>"@" <comp-o-term>
<m-id>            ::=  "m_"<comp-o-term>
                     | <g-var>
                     | <self>

<cnstr-o-term>    ::=  <o-head> ["["<o-attr>-list(",")"]"]
                     | <l-var>
                     | <exp-name>
<o-cnstr>         ::=  <l-var> <sub-rel> <cnstr-o-term>

<sub-rel>         ::=  "⊑" ("=<") | "⊒" (">=") | "≅" ("==")
<l-var>           ::=  <u-string> | "_"<string>
```

$[\cdots]$	$\cdots$ は省略可能
$\{\cdots\}$	$\cdots$ が 0 回以上
<$\cdots$>-*list*	<$\cdots$> $\{$<$\cdots$>$\}$; $\cdots$ が 1 回以上
<$\cdots$>-*list*("a")	<$\cdots$> $\{$"a" <$\cdots$>$\}$; $\cdots$ がデリミタ "a" 付きで 1 回以上
a (a')	a は論文用、a' は実際の記法

図 1: オブジェクト項の構文

2.2 属性と継承

2.2.1 属性項

オブジェクト項で表現されるオブジェクトの性質の記述のために属性項が用いられる。$o, o_1, \cdots, o_n$ をオブジェクト項、$l_1, \cdots, l_n$ をラベルとする。属性項は下記で定義される。

$$o/[l_1\ op_1\ o_1, \cdots, l_n\ op_n\ o_n].$$

ここで op_i $(1 \leq i < n)$ は、$\rightarrow, \leftarrow$ または $=$ である。各々の $l_i\ op_i\ o_i$ を o の**属性 (記述)** という。たとえば、以下のような属性項が記述できる。

$$taro \sqsubseteq person/[name = \text{"}Taro\text{"}, friend^{+} \twoheadrightarrow \{hanako, jiro\}, father \rightarrow person]$$

これは、$taro$ は、$person$ のインスタンスであり、名前は "$Taro$"、友達 ($friend$) としては、$\{hanako, jiro\}$ が含まれ、父親は $person$ のインスタンスであることが知られているという意味である。"/" の左側はオブジェクト項であり、右側はオブジェクト項の属性を指定するための**属性指定 (attribution)** である。

各属性は、原子ラベルに関しては 2.1節で説明したオブジェクト項のように制約の形式で書くことができ、さらにオブジェクト項の制約は、変数名を適当に変えることによって属性項の制約の集合の中に一緒に集めて記述することができる。たとえば、下記の例を考える。

$$o[l = X] | \{X \sqsubseteq o'\}/[l_1 = X, l_2 = Y] | \{X \cong o_1, Y \sqsupseteq o_2\}.$$

属性指定の中の X を Z に変えることによって、下記のように書き直すことができる。

$$o[l = X]/[l_1 = Z, l_2 = Y] \mid \{X \sqsubset o', Z \cong o_1, Y \sqsupseteq o_2\}.$$

このほかに、ラベルの扱いが、オブジェクト項の中と属性指定の中とでは異なることに注意が必要である。たとえば

$$taro \sqsubseteq person/[name = \text{"}Taro\text{"}, friend^{+} \twoheadrightarrow \{hanako, jiro\}, father \rightarrow person]$$
$$taro/[father = saburo, age = 10]$$

という記述があれば、これらは、同じオブジェクトに対する記述であることからマージされ以下が導かれる。

$$taro \sqsubset person/[name = \text{"}Taro\text{"}, friend^{+} \twoheadrightarrow \{hanako, jiro\}, father = saburo, age = 10]$$

属性項の構文を図 2に示す。

属性名はオブジェクト項を用いることができる。オブジェクト項を属性名に使用した場合、属性名に引数が与えられることになる。これは、引数に依存した属性値の計算規則を与えることによりオブジェクトに対するメソッドを表現できる。

属性の値をみるために、たとえば、上記で l_1 の値をみるために $o[l = x].l_1$ という項を使う。これを**ドット項 (dotted term)** という。ドット項 $o[l = x].l_1$ の解釈は、$l_1((o, \{(l, x)\}))$ である。

属性の値としては、個体オブジェクト項のほかに、個体オブジェクト項の集合がある。たとえば、太郎の趣味がスキーとテニスであることを、オブジェクト "$taro$" とその属性 "$hobby$" を以下のように定義することで記述できる。

$$taro/[hobby^{+} = \{ski, tennis\}]$$

特に個体オブジェクト項の集合と限定している意味は、(少なくとも現時点では)$QUIXOTE$ では集合の集合を除外しているということである。

実際には、$QUIXOTE$ における集合は、単なる個体オブジェクト項の集合ではなく、ある特殊な性質を持つが、その説明の前に、個体オブジェクト項の集合上に定義される順序関係について述べることにする。

```
<a-term>        ::=  <o-term> [<sub-rel> <objs>]
                          ["/"["["<a-attr>-list(",")"]"] ["|"{"<a-cnstr>-list(",")"}"]]
<objs>          ::=  <o-term>
                   | "{"<o-term>-list(",")"}"
<a-attr>        ::=  <lab> <attr-op> <ind-term>
                   | <set-lab> <attr-op> <set-terms>
                   | <lab> "→" ("->") <set-terms>
                   | <set-lab> "←" ("<-") <ind-term>
<ind-term>      ::=  <o-term>
                   | <m-id>
                   | <sort>
<a-cnstr>       ::=  [<m-id>":"] <ind-term> <sub-rel> [<m-id>":"] <ind-term>
                   | [<m-id>":"] <set-lhs> <set-rel> [<m-id>":"] <set-terms>
<sort>          ::=  <sort-name>["/"["["<a-attr>-list(",")"]"]
                     % 第1版は、単に <sort-name> しか許されない。
<sort-name>     ::=  "s_"<comp-o-term>

<set-lab>       ::=  <lab>"+" (<lab>"+")
<set-terms>     ::=  "{"<o-term>-list(",")"}"     % 1 版は <g-var>,<set-var> は含まない
                   | "{"<m-id>-list(",")"}"       % 1 版は <g-var>,<set-var> は含まない
                   | <g-var>
                   | <o-term>"!"<set-lab>
                   | <set-var>
<set-lhs>       ::=  <o-term>"!"<set-lab>
                   | <set-var>
<set-var>       ::=  <g-var>"+" (<g-var>"+")

<rule-a-term>   ::=  <o-term> ["/"["["<a-attr>-list(",")"]"] ["|"{"<a-cnstr>-list(",")"}"]]

<set-rel>       ::=  "⊑H" ("+<") | "⊒H" (">+") | "≅H" ("=+=")
```

図 2: 属性項の構文

Quixote では、個体オブジェクト項間の順序関係 $\sqsubseteq$ により、集合間の順序を Hoare 順序 (Hoare pre-order) として定義している。すなわち、集合間の順序を $\sqsubseteq_H$ で表すことにすると、個体オブジェクト項の集合 S_1 と S_2 が $S_1 \sqsubseteq_H S_2$ という関係にあるならば、

$$\forall x \in S_1\ \exists y \in S_2 \quad x \sqsubseteq y$$

が成り立つ。なお、$S_1 \cong_H S_2$ は、$S_1 \sqsubseteq S_2$ かつ $S_1 \sqsupseteq S_2$ のときである。たとえば、$1 \sqsubseteq odd, 2 \sqsubseteq even, 3 \sqsubseteq odd. odd \sqsubseteq int, even \sqsubseteq int$ とすると、

$$\begin{aligned}
\{1,2\} &\sqsubseteq_H \{1,2,3\} \\
\{1,2\} &\sqsubseteq_H \{odd, even\} \\
\{1,2,3\} &\sqsubseteq_H \{int\} \\
\{1, int\} &\sqsubseteq_H \{2, int\} \\
\{2, int\} &\sqsubseteq_H \{1, int\}
\end{aligned}$$

が成り立つ。関係 $\sqsubseteq_H$ が前順序であることは定義から明らかだが、最後の二つの例から分かるように、反対称律は成り立たない。すなわち、関係 $\sqsubseteq_H$ は半順序ではない。

Quixote では、主に計算量的な考慮から、扱う集合の領域を、個体オブジェクト項の集合の領域を $\sqsubseteq_H$ で割った商集合としている。この制限のもとでは、$\sqsubseteq_H$ が半順序となるのは明らかである。このとき問題となるのは、$\sqsubseteq_H$ の同値類の代表元をいかに決めるかということになる。*Quixote* では、以下の規則を用いて、集合の正規形を決定することにしている。

$$\sigma \sqsubseteq \tau \wedge \{\cdots, \sigma, \cdots, \tau, \cdots\} \Rightarrow \{\cdots, \tau, \cdots\}$$

たとえば、先に示した例では、

$$\begin{aligned}
\{1, int\} &\Rightarrow \{int\} \\
\{2, int\} &\Rightarrow \{int\}
\end{aligned}$$

となり、関係 $\sqsubseteq_H$ が半順序であることが確かめられる。

また、集合値ラベルの有限集合 L^+ を仮定する。L^+ と、原子ラベルの有限集合 L との共通部分は空である。*Quixote* では、集合値ラベルの名前は常に $^+$ で終ると規定し、両者の区別をしており、集合値ラベルは、その名前の $^+$ までの部分と等しい名前を持つ原子ラベルとは、まったく無関係である。したがって、

$$smith/[name = \text{“Smith”}, name^+ = \{\text{“William”}, \text{“Bill”}, \text{“Billy”}\}]$$

という定義があるときに、オブジェクト $smith$ の$name$-値は “Smith” であり、$name^+$ 値は

$$\{\text{“William”}, \text{“Bill”}, \text{“Billy”}\}$$

であり、両者は矛盾しない。

下記のように、集合を値域とする変数を含む属性項も記述可能である。

$$human/[hobby^+ \leftarrow X^+, children^+ = Y^+]$$

Quixote では、集合変数についても名前は常に $^+$ で終ると規定し、両者の区別をしている。

下記は、集合に関する syntax-sugaring の一覧である。

$$\begin{aligned}
o/[l^+ = \{o_1, \ldots, o_n\}] &\Leftrightarrow o/[l^+ = X^+] \mid \{X^+ \cong_H \{o_1, \ldots, o_n\}\}, \\
o/[l^+ \sqsubseteq \{o_1, \ldots, o_n\}] &\Leftrightarrow o/[l^+ = X^+] \mid \{X^+ \sqsubseteq_H \{o_1, \ldots, o_n\}\}, \\
o/[l^+ \sqsupseteq \{o_1, \ldots, o_n\}] &\Leftrightarrow o/[l^+ = X^+] \mid \{X^+ \sqsupseteq_H \{o_1, \ldots, o_n\}\}, \\
o/[l = \{o_1, \ldots, o_n\}] &\Leftrightarrow o/[l = X] \mid \{\{X\} \sqsubseteq_H \{o_1, \ldots, o_n\}\}, \\
o/[l^+ \leftarrow o'[\cdots]] &\Leftrightarrow o/[l^+ = X^+] \mid \{X^+ \sqsupseteq_H \{o'[\cdots]\}\},
\end{aligned}$$

これ以外の組合せ、すなわち、

$$o/[l = \{o_1, \ldots, o_n\}]$$
$$o/[l \leftarrow \{o_1, \ldots, o_n\}]$$
$$o/[l^+ = o'[\cdots]]$$
$$o/[l^+ \rightarrow o'[\cdots]]$$

は型エラーとなる。

また、これとは別に、各属性 "l" もしくは "l^+" に対して、あるオブジェクトのインスタンス・オブジェクトでの属性値を集めてくる特殊なラベル "$\{l\}$" もしくは "$\{l^+\}$" を持つ。たとえば、

$$taro \sqsubseteq person/[my\_label = a].$$
$$jiro \sqsubseteq person/[my\_label = b].$$

このとき、$person$ オブジェクトのラベル "$\{my\_label\}$" は、$person$ のインスタンスでのオブジェクトの my_label の値をすべて集めたものとなる。すなわち、

$$person/[\{my\_label\} \leftarrow \{a, b\}]$$

ただし、利用者がこのラベルを指定できるのは、問合せのときだけである。

2.2.2 属性の継承

本節では、オブジェクト項間における属性の継承について、継承と制約のための一般規則を用いて説明する。

属性指定は、$\sqsubseteq$-順序で関連づけられたオブジェクト項の間で継承されると考えるのが自然である。たとえば、下記の例を考える。

$$swallow \sqsubseteq bird.$$
$$bird/[canfly \rightarrow yes].$$

$swallow$ は $bird$ であり、$bird$ は $[canfly \rightarrow yes]$ という属性指定をもつので、$swallow$ は、指定がなくても、同じ属性指定を持つべきである。

オブジェクト間における属性の継承に関する一般的な規則は下記である。

Definition 1 (継承の規則)

$$o_1 \sqsubseteq o_2 \quad \Rightarrow \quad o_1.l \sqsubseteq o_2.l \wedge o_1.l^+ \sqsubseteq_H o_2.l^+.$$

□

この規則によって、下記が成り立つ。

もし、$o1 \sqsubseteq o2$ ならば、

- もし o_2 が属性指定 $[l \rightarrow o']$ をもつなら、o_1 も同じ属性指定をもち、
- もし o_1 が属性指定 $[l \rightarrow o']$ をもつなら、o_2 も同じ属性指定を持つ。

上の例では、$bird$ が $[canfly \rightarrow yes]$ という属性を持っているので、$swallow$ もそれを継承し、

$$swallow/[canfly \rightarrow yes]$$

という属性指定がされていることになる。

$=$ は $\rightarrow$ と $\leftarrow$ が同時に成り立つことであったので、属性指定 $[l = o']$ については両者が成り立つことに注意が必要である。たとえば、オブジェクト項 $fruits$ と $banana$ について、

$$banana \sqsubseteq fruits,$$
$$banana/[color = yellow]$$

であるとすると、$[color \leftarrow yellow]$ という属性指定が $banana$ から $fruits$ に継承され、

$$fruits/[color \leftarrow yellow]$$

という属性指定がされていることになる。

さらに、継承について追加規則を考えることにより、属性指定の継承に関する例外 (exception) の概念を導入することができる。例外の規則は下記のように書くことができる。

Definition 2 (例外の規則)
オブジェクト項の中のラベルについての記述は、そのオブジェクト項の属性指定に優先する。 □

たとえば、$bird[canfly \rightarrow no]$ というオブジェクト項の$canfly$- 値について、下記の定義に関連させて考えてる。

$$bird/[canfly \leftarrow yes].$$

定義により、$bird[canfly \rightarrow no] \sqsubseteq bird$ であるので、継承規則によって、$bird[canfly \rightarrow no]$ は属性指定 $[canfly \leftarrow yes]$ を継承する。ところが、$bird[canfly \rightarrow no]$ は、オブジェクト項の中に $[canfly \rightarrow no]$ という属性指定を持っているので、$canfly$- 値としてはこれが優先され、

$$bird[canfly \rightarrow no]/[canfly \rightarrow no]$$

が得られる。

例外の規則によって、下記が成り立つ。

$$o[l_1\ op_1\ x_1, \cdots, l_n\ op_n\ x_n]/[l_1\ op_1\ x_1, \cdots, l_n\ op_n\ x_n].$$

ここで、$op_i \in \{\rightarrow, \leftarrow, =\} (1 \leq i \leq n)$ である。

継承と例外を扱うことによって、ラベル $l_1 = x_1, \cdots, l_n = x_n$ の記述を持つオブジェクト項 o の属性指定は、下記のように定義される。

$$((C_o \cup C_{\uparrow}^{o} \cup C_{\downarrow}^{o}) \setminus \{l_1, \cdots, l_n\}) \cup \{o.l_1\ op_1\ x_1, \cdots, o.l_n\ op_n\ x_n\}$$

ここで、C_o はオブジェクト項 o 自身がもつオブジェクト項の属性指定に関する制約の集合、$C_{\uparrow}^{o}$ は、オブジェクト項 o より大きいオブジェクト項から継承された o の属性指定に関する制約の集合、$C_{\downarrow}^{o}$ はオブジェクト項 o より小さいオブジェクト項から継承された o の属性指定に関する制約の集合である。また $C \setminus \{l_1, \cdots, l_n\}$ は C から $l_1, \cdots, l_n$ の属性に関する原子制約を除いた制約である。

オブジェクト項の属性指定が解を持たない場合もありうる。その場合には、オブジェクト項の定義は矛盾 (inconsistent) であるという。たとえば、下記を考える。

$$bird/[canfly - yes]$$
$$penguin/[canfly = no]$$
$$penguin \sqsubseteq bird.$$

ここで、$penguin$ は $bird$ から、$[canfly \leftarrow yes]$ という属性を継承するので、$penguin$ は、

$$penguin/[canfly = X] | \{X \cong no, X \sqsubset yes\}$$

という制約を持つことになるが、$\cong$ が $\sqsubseteq$ と $\sqsupseteq$ とが同時に成り立っていることに注意すると、これは、

$$penguin/[canfly = X]|\{X \sqsubseteq no, X \sqsupseteq no, X \sqsubseteq yes\}$$
$$\Downarrow$$
$$penguin/[canfly = X]|\{X \sqsubseteq (no \mid yes), X \sqsupseteq no\}$$

となり、仮に $yes \mid no = \bot$ とすれば、

$$penguin/[canfly = X]|\{X \sqsubseteq \bot, X \sqsupseteq no\}$$

で、変数 X の解はない。しかもこの場合、$bird$ についても、$penguin$ から属性指定 $[canfly \leftarrow no]$ を継承するので、

$$bird/[canfly = X]|\{X \cong yes, X \sqsupseteq no\}$$
$$\Downarrow$$
$$bird/[canfly = X]|\{X \sqsubseteq yes, X \sqsupseteq yes, X \sqsupseteq no\}$$
$$\Downarrow$$
$$bird/[canfly = X]|\{X \sqsubseteq yes, X \sqsupseteq (yes \mid no)\}$$

となり、仮に $yes \mid no = \top$ とすれば、

$$bird/[canfly = X]|\{X \sqsubseteq yes, X \sqsupseteq \top\}$$

で、変数 X の解はない。

例外継承に関しても同様な問題が起こり得る。オブジェクト項 $bird$ と $bird[canfly = no]$ があるときに、

$$bird/[canfly = yes]$$

という属性指定がされている場合を考える。例外の規則により、$bird[canfly = no]$ については、

$$bird[canfly = no]/[canfly = no]$$

であり、矛盾は起こらないが、$bird$ に関しては、上の例から、

$$bird/[canfly = X]|\{X \cong yes, X \sqsupseteq no\}$$
$$\Downarrow$$
$$bird/[canfly = X]|\{X \sqsupseteq (yes \mid no), X \sqsubseteq yes\}$$
$$\Downarrow$$
$$bird/[canfly = X]|\{X \sqsupseteq \top, X \sqsubseteq yes\}$$

と矛盾してしまう。

2.3 モジュールとルール

Quixote では、オブジェクト項の集合をモジュール化するために、モジュール識別子と呼ぶ特殊なオブジェクト項を導入する。

その形式は下記である。

$$m :: \sigma|C$$

これを (単位) ルール (**(unit) rule**) といい、m がモジュール識別子、$\sigma|C$ を表わす。この単位ルールは $\sigma|C$ が m 中にあると言っている。モジュール識別子 m が省略され、単に

$$\sigma|C$$

と書かれたルールは、$\sigma|C$ がすべてのモジュールにあると言っている。

単位ルール $s :: \sigma$ に対応して、命題 (proposition) $m : \sigma$ が下記のように定義される。

$$(m : \sigma|C) \text{ is true} \quad \text{iff} \quad m :: \sigma|C.$$

Quixote では、(非単位) ルールも下記のように記述可能である。

$$m :: \sigma \Leftarrow m_1 : \tau_1, m_2 : \tau_2, \cdots, m_n : \tau_n||C.$$

ここで、$m, m_1, \cdots, m_n$ はモジュール識別子、$\sigma, \tau_1, \cdots, \tau_n$ は属性項、C は制約である。このとき、σ をルールのヘッド、$m_1 : \tau_1, m_2 : \tau_2, \cdots, m_n : \tau_n$ をルールのボディという。このルールは、$m_i : \tau_i$ $(1 \leq i \leq n)$ がすべてが真で、C が充足可能ならば、σ が m の中にある ($m : \sigma$ が真) ということを表わしている。また、このルールは、このルールが m の中にある (このルールは m だけからアクセス可能である) ということも表わしている。ボディのモジュール識別子 m_i が省略され、単に τ_i と書かれた場合、"ヘッドのモジュール識別子 (この場合は m) で表されるモジュールに τ_i がある" という命題とみなされる。ヘッドのモジュール識別子も省略されていた場合には、"すべてのモジュールに τ_i がある" という命題とみなされる。たとえば、

$$m_1 :: o_1,$$
$$m_2 :: o_2 \Leftarrow m_1 : o_1,$$
$$m_2 :: o_3 \Leftarrow o_1,$$
$$m_2 :: o_4 \Leftarrow o_2$$

とすると、$m_1 : o_1, m_2 : o_2, m_2 :: o_4$ は真であるが、$m_2 : o_3$ は偽である。

ルールの構文は、図3の通りである。

Quixote は、データベースの更新のためにトランザクション (transaction) と言う概念を持っており、それをネストすることが出来る。トランザクションの終了には、トランザクション中での更新を有効とする正常終了 (end_transaction) と、更新を無効とする異常終了 (abort_transaction) の 2 種類がある。最初のものは、

"begin_transaction" <props> "end_transaction"

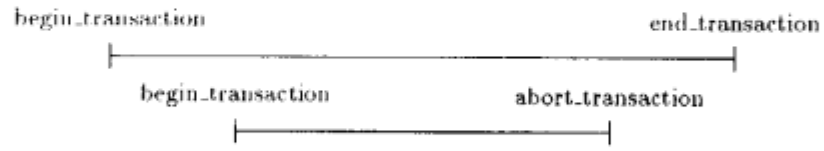
で実行され、あとのものは、

"begin_transaction" <props> "abort_transaction"

で実行される。

トランザクションがネストしている場合には、以下のように働く。

- 正常終了するトランザクション中に、異常終了するトランザクションがネストしている場合、つまり、



の時は、ネストしているトランザクションは異常終了し、全体のトランザクションが正常終了する。

- 異常終了するトランザクション中に、正常終了するトランザクションがネストしている場合、つまり、

```
<rules>          ::=  <rule>
                    | <u-rule>
                    | <rules> ";;" <rules>

<rule>           ::=  [<m-lab> "::"] <clause>
                    | [<m-lab> "::"] "{{"<clause>-list(";;") "}}"
<clause>         ::=  <rule-a-term> ["⇐" ("<=") <body>]
<body>           ::=  <props>["||" "{"<r-cnstr>-list(",")"}"]
<props>          ::=  <prop>
                    | <props> "," <props>
<prop>           ::=  [<m-id>":"] <a-term>
                    | <a-cnstr>
<m-lab>          ::=  <m-id>
                    | "{"<m-id>-list(",")"}"
<r-cnstr>        ::=  <a-cnstr>

<u-rule>         ::=  [<m-lab> "::"] <u-clause>
                    | [<m-lab> "::"] "{{"<u-clause>-list(";;")"}}"
<u-clause>       ::=  <u-rule-a-term> ["⇐" ("<=") <u-body>]
<u-rule-a-term>  ::=  <rule-a-term>
<u-body>         ::=  <u-props> ["||" "{"<r-cnstr>-list(",")"}"]
<u-props>        ::=  <u-prop>
                    | <u-rule-a-term>
                    | <u-props> ";" <prop>
                    | <prop> ";" <u-props>
                    | <u-props> ";" <u-props>
<u-prop>         ::=  [<m-id>":"] <u-flag> <prop>
                    | <trans>
```

図 3. ルールの構文

の時は、ネストしているトランザクションは正常終了せず、すべてのトランザクションが異常終了する。

Quixote では、更新を実現している。更新の assert は "+" で、retract は "−" で表す。更新を実現するために、項の評価の順番を規定するものと、規定しないものを明示できるようにしている。前者が ";" で、後者が "," である。ファクトのオブジェクト o を更新し、問合せを行う例は以下の通りである。

$$?- \ -o/[l=a],\ +o/[l=b],\ o'/[l'=X].$$

Quixote では、ルールで、オブジェクト指向におけるメソッドを実現している。それは、ルールを以下のように解釈することによって、可能にしている。

$$o/[l=a] \Leftarrow implement$$

つまり、ルールのヘッドのオブジェクト項に対して、ラベルをメソッド、ルールのボディをそのメソッドのインプリメントが記述されていると思うのである。上記の記述は、オブジェクト o の、メソッド l が記述されていると解釈される。たとえば、ルール

$$person/[legal\_name = X] \Leftarrow person/[last\_name = X]$$

は、オブジェクト $person$ のメソッド $legal\_name$ を記述されていると解釈できる。

複数のルールが同じオブジェクト項の属性値の決定に関係する場合は複雑である。たとえば、今、番号が 2 の倍数だと航空券が、3 の倍数だと宿泊券が当たる籤があるとすると、そのルールは下記のように書ける。

$$o/[hotel\text{-}ticket = yes] \Leftarrow o/[number = X]||\{X \sqsubseteq 2n\}, \quad (1)$$

$$o/[air\text{-}ticket = yes] \Leftarrow o/[number = X]||\{X \sqsubseteq 3n\} \quad (2)$$

ここで $2n, 3n$ はそれぞれ、2,3 の倍数の全体を包摂する基本オブジェクトである。単位ルール $o/[number = 5]$ を与えると、どちらのルールも成り立たない。$o/[number = 2]$ を与えると、ルール (1) だけが成り立つので、

$$o/[number = 2, hotel\text{-}tichet = yes]$$

が得られる。$o/[number = 6]$ が与えられると、両方のルールが成り立ち、

$$o/[number = 6, hotel\text{-}ticket = yes, air\text{-}ticket = yes]$$

が得られる。

モジュール間におけるルールの継承についての一般的な規則は下記である。r をルールとすると、

$$m \sqsupseteq_S m' \ \wedge \ m' :: r \Rightarrow m : r.$$

$\sqsupseteq_S$ はモジュール識別子の集合上の 2 項関係で**部分モジュール関係**という。オブジェクト項間の包摂関係 $\sqsubseteq$ が、基本オブジェクト間でしか定義されないのに対し、部分モジュール関係 $\sqsubseteq_S$ は、任意のモジュール識別子の間で定義できる。たとえば、

$$m_1[l_1 = o] \sqsupseteq_S m_2[l_2 = o']$$

という指定が可能である。さらに継承されるモジュール間には、一種の集合演算を定義できる。部分モジュール関係の構文は図 4の通りである。ただし、$\sqsupseteq_S$ は半順序関係ではないので、

```
<m-sub>    ::=  <m-id> "⊒S" ("⊃-") <m-desc>
<m-desc>   ::=  <m-id>
             |  "(" <m-desc> ")"
             |  <m-desc> "∪" ("+") <m-desc>
             |  <m-desc> "\" ("-") <m-desc>
```

図 4: 部分モジュール関係の構文

$$\cong_S \ = \ \sqsupseteq_S \ \wedge \ \sqsubseteq_S$$

で定義される同値関係を考え、それによって定義される半順序関係

$$\sqsupseteq_S \ / \cong_S$$

を以下 (単に $\sqsupseteq_S$ として) 考える。

部分モジュール関係の例を考えよう。

$$m_1 :: john \cong human[name = \text{"John"}].$$
$$m_2 :: john/[age \rightarrow 20].$$
$$m_3 :: john/[age \rightarrow 30].$$
$$m_2 \sqsupseteq_S m_1.$$
$$m_3 \sqsupseteq_S m_1.$$

$john$ は、それが m_2 にあるか m_3 にあるかによって、20 歳か 30 歳であり、その名前については m_2 と m_3 のどちらにおいても "$John$" である。ここで、オブジェクト項の存在は、モジュール全体についてグローバルであるが、オブジェクト項の属性指定はローカルであり、かつモジュール識別子間の包摂関係によって継承されるということを記憶することが必要である。もし、$m_2 \sqsubseteq_S m_4$ かつ $m_3 \sqsubseteq_S m_4$, という指定を行った場合には、m_4 は $[age \rightarrow 20 \mid 30] = [age \rightarrow \bot]$ という制約、すなわち、$john$ は矛盾の定義を持つことになる。

ルールの先頭のモジュール識別子が省略されていた場合は、そのルールはすべてのモジュールにあるという意味だが、ボディのモジュール識別子が省略されていた場合、ルールの先頭のモジュール識別子が省略されていることを意味している。たとえば、

$$m_2 \sqsupseteq_S m_1,$$
$$m_1 :: o_1 \Leftarrow o_2,$$
$$m_2 :: o_2$$

のとき、m_2 は m_1 を継承するので、

$$m_2 :: o_1 \Leftarrow m_1 : o_2$$

が生ずる。すなわちルールの条件部は部分モジュール関係によって変化しない。そこで、条件部を継承されたモジュールで評価するために "&self" を導入する。たとえば

$$m_2 \sqsupseteq_S m_1,$$
$$m_1 :: o_1 \Leftarrow \&self : o_2,$$
$$m_2 :: o_2$$

のとき、m_2 が m_1 を継承した時点で "&self" が評価され、

$$m_2 :: o_1 \Leftarrow m_2 : o_2$$

となる。$m_2 : o_2$ なので、o_1 は m_2 で真となる。

モジュール識別子はオブジェクト項として定義されるので、モジュールはパラメタライズすることができる、すなわち抽象化されることが可能である。たとえば、下記を考える。

$$m[l = X] :: john[age \rightarrow X].$$

変数 X は、処理の途中で、ある整数に具体化されることになる。

また、先に述べたように、$\sqsupseteq_S$ はオブジェクト項の包摂関係と異なり、任意のモジュール間に定義できる。このとき、モジュール識別子中に使われる変数 (パラメータ) は、以下のようになる。

$$m[l_1 = X, l_2 = Y, l_3 = Z] \sqsupseteq_S m_1[l_4 = X, l_5 = W] \bowtie m_2[l_6 = X, l_7 = Y, l_8 = W]$$

Z はダミー、W の制約は外れる。(ただし、$\bowtie$ は、2 つモジュール間のオペレータ、"$\cup$" 、あるいは "$\backslash$" を示す。)

2.4 プログラム (データベース)

QUIXOTE のプログラムは

1) 環境セクション
2) モジュール・セクション
3) オブジェクト・セクション
4) ルール・セクション

の 4 つのセクションから構成されている。その構文は、図 5の通りである。

環境セクションの構文は、図 6の通りである。

その他のセクションについては図 7の通りである。

```
<program>       ::=  <program-def> "."
<program-def>   ::=  <b-pgm> ";;" <section>-list(";;") ";;" <e-pgm>
                   | <section>-list(";;")
<section>       ::=  <env-sect>
                   | <obj-sect>
                   | <mod-sect>
                   | <rule-sect>
<b-pgm>         ::=  "&begin_program" | "&b_pgm"
<e-pgm>         ::=  "&end_program"   | "&e_pgm"
```

図 5: プログラムの構文

2.5 問合せ

前節までに定義した *Quixote* データベースあるいは *Quixote* プログラムに対する問合せの構文は、図 8の通りである。

問合せに <program-def> を許すことによって、問合せ時に、オブジェクト項間の包摂関係や部分モジュール関係を変更したり、モジュール毎にルールやファクトを追加することができる。また、<q-mode> によって、問合せ処理に制限を付加できるようにもなっている。

3 意味論

Quixote の他の言語と大きく異なる特徴の一つは、意味論が Aczel[Aczel 88] の提案した超集合論 ZFC$^-$/AFA に基づくことである。*Quixote* はオブジェクトの循環構造を扱うため、意味論における定義域は ZFC$^-$/AFA におけるラベル付きグラフの集合である。3.1節ではラベル付きグラフを、3.2節ではラベル付きグラフ上の包摂関係を定義する。3.3節では、これらに関する制約の可解性を議論する。3.4節では、ラベル付きグラフ上での属性項の解釈を与える。本節中の、ラベル付きグラフの上での制約に関する部分は、Mukai[Mukai 90(1), Mukai 90(2)] に負うところが大きい。

3.1 ラベル付きグラフ

この節では、オブジェクト項を解釈するための定義域を与えるために、超集合の定義域のサブクラスを定義しよう。

準備として、原始オブジェクトの有限集合 BO と束 $BO^* = (BO, \preceq, \top, \bot)$ を仮定する。任意の $a, b \in BO$ について、$a \preceq b$ は、b が a を包摂する、すなわち、a は b であることを意味する。$\top$ および $\bot$ は、それぞれ、BO^* の上限および下限である。また、$a \sqcap b$ および $a \sqcup b$ を、それぞれ、$\{a, b\}$ の下限および上限とする。BO^* の要素が *Quixote* における基本オブジェクトに対応する。$\backslash$ を変数の全体とし、T を基礎オブジェクト項の全体、$T[\backslash]$ を変数を含むことが可能なオブジェクト項の全体であるとする。また、単数値ラベルの有限集合 L を仮定する。

次に、Aczel による超集合の定義域のサブクラス $\mathcal{G}$ を与える。$\mathcal{G}$ の要素は、アークをラベルと関連づけたグラフについての集合論的符号化であるため、**ラベル付きグラフ (labeled graph)** と呼ばれる[3]。(基礎) ラベル付きグラフの全体は、すべての $G \in \mathcal{G}$ が対 (a, b) であるような最大の集合である。ここで、a は $a \in BO^*$ 、b は空集合か、または、定義域として L の部分集合、余域として $\mathcal{G}$ の部分集合をもつ (有限) 関数である。

基礎オブジェクト項から基礎ラベル付きグラフへの写像が全単射、すなわち T と $\mathcal{G}$ との間の対応が 1 対 1 であることを示すことは容易である。たとえば、$human$ 、$human[age = 20, sex = male]$ 、および $[age = 20]$ の各オブ

[3] Aczel[Aczel 88] は、"ラベル" という用語を別な意味で使っている。Aczel は、"ラベル" をグラフの各ノードに関連づけており、これは本稿におけるノードの値に対応する。

```
<env-sect>    ::= <b-env> ";;" <env>-list(";;") ";;" <e-env>
<env>         ::= "&name[&pgm-name=" <pgm-name>"]"
                | "&author[&aut-name=" <aut-name>"]"
                | "&date[&date="<date>"]"
                | <b-exp> ";;" <exp-def>-list(";;") ";;" <e-exp>
                | "&include""["<def-lib> list(",")"]"
<exp-def>     ::= <exp-name> "=" <o-term>
                | <exp-name> "=" <m-id>          % 9/19 修正
                | <exp-name> "="
                  "&del" "("<lab>-list(",")")" <exp-name>
                | <exp-name> "="
                  <exp-name>["["<o-attr>-list(",")"]"["|" "{"<o-cnstr>-list(",")"}"]]
                | <exp-name> "="
                  "&abs" "(" "["<o-attr>"]" ["|" "{"<o-cnstr>-list(",")"}"] ")"<exp-name>
<def-lib>     ::= <lib-label> "=" <lib-name>
                | <lib-label>+ (<lib-label>"+") "=" "{"<lib-name>-list(",")"}"
                | "&sort_lib=" <sort-mod>
<b-env>       ::= "&begin_environment_section" | "&b_env"
<e-env>       ::= "&end environment_section"   | "&e_env"
<b-exp>       ::= "&begin_expression" | "&b_exp"
<e-exp>       ::= "&end_expression"   | "&e_exp"
<pgm-name>    ::= <string>
<aut-name>    ::= <string>
<date>        ::= <string>
<exp-name>    ::= "e_"<string>
<lib-name>    ::= <string>-list("/")
<lib-label>   ::= "&exp_lib" | "&pgm_lib"
<sort-mod>    ::= <l-string>
```

図 6: 環境セクションの構文

ジェクト項は、それぞれ下記のラベル付きグラフに対応する。

$$(human, \emptyset)$$
$$(human, \{(age, \{(20, \emptyset)\}), (sex, \{(male, \emptyset)\})\})$$
$$(\top, \{(age, \{(20, \emptyset)\})\}).$$

簡単のために、$\mathcal{I}$ をオブジェクト項を入力して対応する基礎ラベル付きグラフを戻す解釈関数とする。

$\mathcal{G}$ は、アトムをもつ Aczel の超集合の定義域 V_A のサブクラスである。オブジェクト項の場合には、変数を含むラベル付きグラフの定義域 $\mathcal{G}[\chi_{\mathcal{G}}]$ を持つことが可能である。ラベル付きグラフは、Aczel の超集合の意味で集合論的構成であり、連立方程式の解として一意的に定義される。これは、Aczel の Solution Lemma が述べていることである。たとえば、

$$(human, \{(age, \{(20, \emptyset)\}), (sex, \{(male, \emptyset)\})\})$$

というラベル付きグラフは、下記の連立方程式について変数 x の解を求めたものである。

$$\begin{aligned} x &= (human, \{x_1, x_2\}), \\ x_1 &= (age, \{x_3\}), \\ x_2 &= (sex, \{x_4\}), \end{aligned}$$

```
<obj-sect>    ::=  <b-obj> ";;" <obj-map>-list(";;") ";;" <e-obj>
<obj-map>     ::=  <obj-map> ";;" <subsum> ";;" <obj-sub>-list(";;")
<obj-sub>     ::=  <basic-obj> <sub-rel> <basic-obj>
                 | <basic-obj> <sub-rel> "{"<basic-obj>-list(",")"}"
<b-obj>       ::=  "&begin_object_section" | "&b_obj"
<e-obj>       ::=  "&end_object_section"   | "&e_obj"
<subsum>      ::=  "&subsumption" | "&subsum"
<obj-map>     ::=  "&object_map" | "&obj_map"

<mod-sect>    ::=  <b-mod> ";;" <mod-map>-list(";;") ";;" <e-mod>
<mod-map>     ::=  <modl-map> ";;" <submod> ";;" <m-sub>-list(";;")

<rule-sect>   ::=  <b-rule> ";;" <rules> ";;" <e-rule>
<b-rule>      ::=  "&begin_rule_section" | "&b_rule"
<e rule>      ::=  "&end_rule_section"   | "&e_rule"

<rules>       ::=  <rule>
                 | <u-rule>
                 | <rules> ";;" <rules>
```

図 7: 他セクションの構文

$$x_3 = (20, \emptyset).$$
$$x_4 = (male, \emptyset).$$

さらに厳密にするために、$\gamma_{\mathcal{G}}$ の部分集合から $\mathcal{G}$ の部分集合への関数として割当て (assignment) f を定義する。定義域として変数の集合 X と $\gamma_{\mathcal{G}}$ をもつ割当て f は、すべての $x_i \in X$ について、

$$f(x_i) = f(a_{x_i})$$

が成り立つとき、そしてそのときだけ、連立方程式 $x_i = a_{x_i}$, $(x_i \in X)$ の解である。

Solution Lemma の 2 つの重要な点は下記である。

(1) 解の存在

(2) 解の一意性

たとえば、$X \& human[employee = X]$ というオブジェクト項の解釈 (すなわちラベル付きグラフ) を考える。それは、下記の連立方程式で与えられる。

$$x = (human, \{x_1\}),$$
$$x_1 = (employee, \{x\}).$$

Aczel の補題によって、X の解は一意的に存在するので、それを Ω_1 と呼ぶことにする。Ω_1 は、下記の条件を満たす超集合である。

$$\Omega_1 = (human, \{(employee, \{\Omega_1\})\}).$$

Aczel の超集合論では、下記の連立方程式も同じ解 (Ω_1) を持つ。

$$x = (human, \{x_1\}),$$
$$x_1 = (employee, \{x_2\}),$$
$$x_2 = (human, \{x_3\}),$$
$$x_3 = (employer, \{x_2\}).$$

```
<query>          ::=  "?-" <query-body> [";;" "&q_mode" "[" <q-mode>-list(",") "]"]
                                        [";;" <program-def>] "."
                    | "?-" "&open/" "[" <op-attr>-list(",") "]" "."
                    | "?-" "&close/" "[" <cl-attr>-list(",") "]" "."

<query-body>     ::=  <query-props>["||" "{" <r-cnstr>-list(",") "}"]
                    | <u-props>["||" "{" <r-cnstr>-list(",") "}"]
<query-props>    ::=  <query-prop>
                    | <query-props> "," <query-props>
<query-prop>     ::=  [<n-id>":"] <query-a-term>
                    | <a-cnstr>

<op-attr>        ::=  "&pgm_name=" <pgm-name>
                    | "&lib_name=" <lib-name>
                    | "&lib_name+=" "[" <lib-name>-list(",") "]"
                    | "&mode=" <op-mode>
                    | "&ver=" <version>
<cl-attr>        ::=  "&pgm_name=" <pgm-name>
                    | "&lib_name=" <lib-name>
                    | "&mode=" <cl-mode>
                    | "&exit=" <cl-exit>
<q-mode>         ::=  "&proc_mode=" <proc-mode>
                    | "&ans_mode=" <ans-mode>
                    | "&inheritance=" <inheritance>
<op-mode>        ::=  "&readonly" | "&exclusive"
<cl-mode>        ::=  "&end" | "&abort"
<proc-mode>      ::=  "&single" | "&multi"
<ans-mode>       ::=  "&normal" | "&minimal"
<inheritance>    ::=  "&all" | "&down" | "&up" | "&no" | [<u-flag>] <integer>
<cl exit>        ::=  "&yes" | "&no"
<version>        ::=  <string>
```

図 8: 問合せの構文

すなわち、そうなれば、循環構造を広げる (*unfold* する) ことが可能になるということである。たとえば、Aït-Kaci の ψ- 項モデルでは、循環構造を広げることはできない。すなわち、2 つの連立方程式は同じ解を持つことはできない。これは、循環構造に関する本稿の扱いと、無限木に基づく通常の循環構造の扱いとの間のきわめて重要な相違点である。

3.2 ラベル付きグラフの上での包摂関係

最初に、次の条件を満たす最大の関係として、2 項関係 $\sqsubseteq_{\mathcal{G}}$ を定義する。

a_1, a_2 を基本オブジェクト、b_1, b_2 を原子ラベルの部分集合から $\mathcal{G}$ の部分集合への関数としたとき、$(a_1, b_1) \sqsubseteq_{\mathcal{G}} (a_2, b_2)$ ならば、下記が成り立つ。

- $a_1 \preceq a_2$,
- 任意の対 $(l, g_2) \in b_2$ に対して、$g_1 \sqsubseteq_{\mathcal{G}} g_2$ であるような対 $(l, g_1) \in b_1$ が存在する。

たとえば、下記が成り立つ。

$$(human, \{(age, \{(20, \emptyset)\}), (sex, \{(male, \emptyset)\})\}) \sqsubseteq_{\mathcal{G}} (human, \emptyset).$$

$$(human, \{(age, \{(20, \emptyset)\}), (sex, \{(male, \emptyset)\})\}) \sqsubseteq_{\mathcal{G}} (\top, \{(age, \{(20, \emptyset)\})\}).$$

さらに、前述 Ω_1 の場合のように、$X@human[employee = X, name = \text{“John”}]$ というオブジェクト項の解釈 Ω_2 は下記のようになる。

$$\Omega_2 = (human, \{(employee, \{\Omega_2\}), (name, \{(\text{“John”}, \emptyset)\})\}).$$

通常の集合論で、$\Omega_2 \sqsubseteq_{\mathcal{G}} \Omega_1$ が成り立つのをみるのは容易ではない。しかし、Aczel の超集合論では容易である。直観的に述べると、Ω_1 と Ω_2 を含む集合は、Aczel の超集合論で coinductive に定義され、それは、Ω_1 と Ω_2 をもし比較したとすると、一度だけ広げ、Ω_1 と Ω_2 を使わずにそれらを比較することで十分であるということを意味している。このようにして、$\Omega_2 \sqsubseteq_{\mathcal{G}} \Omega_1$ は、下記の 2 つのラベル付きグラフを比較することによって成り立つことが結論づけられる。

$$(human, \{(employee, \_)\}),$$
$$(human, \{(employee, \_), (name, \{(\text{“John”}, \emptyset)\})\}).$$

ここで、“_” は特殊なアトムと仮定している。

$\sqsubseteq_{\mathcal{G}}$ の定義で、b_2 は関数であるから、g_2 が 2 番目の条件の中に存在するとしたら g_2 は一意的であることに、注意する必要がある。$\sqsubseteq_{\mathcal{G}}$ が前順序であることを調べるのは容易である。さらに、$\sqsubseteq_{\mathcal{G}}$ は半順序になりうる。すなわち、$g_1 \sqsubseteq_{\mathcal{G}} g_2$ かつ $g_2 \sqsubseteq_{\mathcal{G}} g_1$ が成り立つときには、$g_1 = g_2$ は、$\mathcal{G}$ に関する等式理論を付加しなくても、成り立つ。[4]本稿の目的では、$\sqsubseteq_{\mathcal{G}}$ を半順序と仮定する。このような理由で、下記に定義した $\cong_{\mathcal{G}}$ は $\mathcal{G}$ 上の等号である。

$$g_1 \cong_{\mathcal{G}} g_2 \stackrel{\text{def}}{=} g_1 \sqsubseteq_{\mathcal{G}} g_2 \wedge g_2 \sqsubseteq_{\mathcal{G}} g_1.$$

オブジェクト項の上での包摂関係 $\sqsubseteq$ は、$\sqsubseteq_{\mathcal{G}}$ を用いて下記のように定義される

$$o_1 \sqsubseteq o_2 \stackrel{\text{def}}{=} \mathcal{I}(o_1) \sqsubseteq_{\mathcal{G}} \mathcal{I}(o_2).$$

一般に、オブジェクト項の定義は、2.2.1節で示したように制約と関連づけられる。オブジェクト項間の $\sqsubseteq$-関係は制約の解の間の$\sqsubseteq_{\mathcal{G}}$-関係として解釈される。たとえば、$human[age = Y] \mid \{Y \sqsubseteq int\}$ というオブジェクト項は同値関係と包摂関係からなる下記の集合における X の解として定義される。

$$\begin{aligned} X &\cong_{\mathcal{G}} (human, \{X_1\}), \\ X_1 &\cong_{\mathcal{G}} (age, \{Y\}), \\ X_2 &\cong_{\mathcal{G}} (int, \emptyset), \\ Y &\sqsubseteq_{\mathcal{G}} X_2. \end{aligned}$$

問題は、上記のような制約の可解性に関して何らかの条件が必要かということである。3.3 節でこの条件について説明する。

次に、$\mathcal{G}$ の上での**交わり (meet)** と**結び (join)** の操作を導入する。任意の 2 つのラベル付きグラフ (G_1,G_2 とする) について、下記のような記述が可能である。

$$\begin{aligned} G_1 &= (o_1, \{(l_1, \{x_1\}), \cdots, (l_i, \{x_i\}), (l'_1, \{x'_1\}), \cdots, (l'_j, \{x'_j\})\}), \\ G_2 &= (o_2, \{(l_1, \{y_1\}), \cdots, (l_i, \{y_i\}), (l''_1, \{y'_1\}), \cdots, (l''_k, \{y'_k\})\}). \end{aligned}$$

ここで、$0 \leq i, j, k$、o_1, o_2 は基本オブジェクト、l_i, l'_j, l''_k はラベル、x_i, x'_j, y_k, y'_i はラベル付きグラフである。2 つのラベル付きグラフ G_1 と G_2 の交わりと結びは、下記のように再帰的に定義される。

交わり ($G_1 \mid G_2$ と書く)

$$\begin{aligned} G_1 \mid G_2 \stackrel{\text{def}}{=} (o_1 \sqcap o_2, \ & \{(l_1, \{x_1 \mid y_1\}), \cdots, (l_i, \{x_i \mid y_i\}), \\ & (l'_1, \{x'_1\}), \cdots, (l'_j, \{x'_j\}), \\ & (l''_1, \{y'_1\}), \cdots, (l''_k, \{y'_k\})\}) \end{aligned}$$

[4]このことは、$woman \cong human[sex = female]$ のような等式公理を追加してもよいことを意味している。しかし、Aczel の超集合論の基本原理ではすべての超集合は一意的構成 (unique picture) を持つとしているため、超集合の定義域の上で、この種の等式理論を持つことができるかどうかは明確ではない。この原理は、等式理論を付け加えると、破れてしまうようである。

結び ($G_1 \mid G_2$ と書く)

$$G_1 \uparrow G_2 \stackrel{\text{def}}{=} (o_1 \sqcup o_2, \{(l_1, \{x_1 \uparrow y_1\}), \cdots, (l_i, \{x_i \uparrow y_i\})\})$$

この定義を厳密に考えるためには、前述した Ω_1 のような自己参照型の構造が存在するので、1つのオブジェクトに対して複数の定義があってもよいことに注意すべきである。こういう場合には、2つのラベル付きグラフの交わりは、1つの(定義のような)等式によって定義することはできない。下記の2つのラベル付きグラフを考える。

$$X \cong_{\mathcal{G}} (human, \{(employee, \{X\})\}),$$
$$Y \cong_{\mathcal{G}} (john, \emptyset)$$

交わり $X \downarrow Y$ は、下記の連立方程式として与えられる。

$$X \downarrow Y \cong_{\mathcal{G}} (human, \{(employee, \{X \mid Z\})\}),$$
$$X \downarrow Z \cong_{\mathcal{G}} (john, \{(employee, \{X\})\}).$$

もし、$U \cong_{\mathcal{G}} (john, \{(employee, \{U\})\})$ ならば、下記を得る。

$$X \mid U \cong_{\mathcal{G}} ((human \sqcap john), \{(employee, \{X \downarrow U\})\}).$$

この方程式は一意の解を持つ。

下記は成り立つ。

$$\begin{aligned} G \downarrow G &\cong_{\mathcal{G}} G \\ G_1 \downarrow G_2 &\cong_{\mathcal{G}} G_2 \downarrow G_1 \\ G_1 \downarrow G_2 &\sqsubseteq_{\mathcal{G}} G_1 \\ G_1 \uparrow G_1 &\cong_{\mathcal{G}} G_1 \\ G_1 \uparrow G_2 &\cong_{\mathcal{G}} G_2 \uparrow G_1 \\ G_1 &\sqsubseteq_{\mathcal{G}} G_1 \mid G_2 \end{aligned}$$

上の命題について、$\sqsubseteq_{\mathcal{G}}$ は $\mathcal{G}$ に関する半順序であるので、下記の結論は容易である。

Proposition 1

$$G_1 \downarrow G_2 \Leftrightarrow inf\{G_1, G_2\},$$
$$G_1 \uparrow G_2 \Leftrightarrow sup\{G_1, G_2\},$$

3.3 ラベル付きグラフの上での制約の可解性

Barwise[Barwise 89] は、パラメータ付き超集合 (変数を含む超集合) の上に、超集合方程式と超集合包摂関係の集合に関する可解性の条件を示した。その論文の中で、同じ体をもつバイシミュレーション関係とシミュレーション関係の対からなるシミュレーションペアの概念を定義することによって、パラメトリック超集合上における超集合方程式と超集合包摂関係の集合が解を持つということと、明確な条件を満たすシミュレーションペアが同じ集合に対して存在するということは同値であることを示した。

このことは、シミュレーションペアに関する条件を満たす2種類の関係を持てば、その2種類の関係によって制約を定義することができ、さらに、超集合の定義域上でその制約の可解性を調べることができるということを意味している。

本稿における制約の可解性に関する条件を得るために、本稿の制約関係である $\cong_{\mathcal{G}}$ と $\sqsubseteq_{\mathcal{G}}$ は、バイシミュレーション関係とシミュレーション関係の定義に関する条件を満たすことを示さねばならない。

まず最初に、ラベル付きグラフの定義域 $\mathcal{G}$ は、下記の2つの点で、[Barwise 89] のものとは違うことに注意する必要がある。

(1) 本稿の定義域は、超集合の定義域のサブクラス、すなわち、ラベル付きグラフの定義域である。

(2) 本稿におけるアトム (基本オブジェクト) は半順序である。

第1点目は問題ない。第2点目は、困難な問題を引き起こすようにみえるかもしれない。しかし、本稿では、基本オブジェクトに関しては変数を持たない。すなわち、変数はラベル付きグラフを値域としている。そして、基本オブジェクトに関しては、固定の代数構造を想定している。このような理由で、本稿では、基本オブジェクトの固定定義域をみることによって、決定的に考察することが可能な2つの基本オブジェクトに関する比較だけを必要としている。

さらに、Barwise の結果は、制約中のどの変数も具体化される、すなわち、各変数 x に対して、制約中にはただ1つの方程式 $x = u$ が存在することを前提としている。ここで、u は超集合 (ラベル付きグラフ) あるいはアトムである。一般に、ある変数は、具体化されないかも知れない。そういう場合に対して、本稿では、制約の解空間は変えずに、制約を拡張する方法を確実に行えるようにしなければならない。

Mukai[Mukai 90(1), Mukai 90(2)] は Barwise の結果を拡張して、遺伝的な (hereditary) 有限集合の上の制約は satisfaction-complete かつ solution-compact であることを示した。このことは、AFA-宇宙のサブクラスが、CLP(X) のための定義域として使うことができることを示唆している。本稿のラベル付きグラフの定義域についても同様である。

次に、ラベル付きグラフの上における制約の可解性に関して2つの結果を示す。1つは制約の可解性条件に関することで、これが [Barwise 89] におけるものと同様であることを示す。その方法は、$\cong_{\mathcal{G}}$ と $\sqsubseteq_{\mathcal{G}}$ に関する明確な制約規制を与えること、それがバイシミュレーションおよびシミュレーション関係を定義するための条件をも含んでいることを示すことである。もう1つは、与えられた制約の拡張可能性に関することであり、これによって、各変数は具体化されることになる。

これら2つの結果は、Mukai[Mukai 90(2)] によって直接与えられた。ここでは、いくつかのコメントを付けてその結果だけを示す。詳細については [Yasukawa and Mukai 90] に述べる。Mukai [Mukai 90(2)] では、超集合上の制約が解を持つための条件を、無矛盾で制約規則に関して閉じている原始制約の集合の存在条件として特徴づけるという方針が示されている。以下の議論は、そのの方針をそのまま適用している。違いは制約の対象領域がラベル付きグラフの領域に制限されている点だけであり、基本的な結果は Mukai[Mukai 90(2)] から直接得られるものである。

まず最初に、本稿で扱う制約とその解を定義する。これは [Mukai 90(2)] のものと同様である。

原子制約は、方程式 $u \cong_{\mathcal{G}} v$ または包摂 $u \sqsubseteq_{\mathcal{G}} v$ である。ここで、u, v は $\mathcal{G}[\chi] \cup \chi$ である。制約は、原子制約の集合であり、原子制約の連言である。充足関係 ($\models$) は通常のように、すなわち、割当て f と明白な節をもつ制約 c の間の2項関係 ($f \models c$ と書く) として定義される。$f \models c$ ならば、割当て f は制約 c の解である。

与えられた制約を拡張するために、制約に関する規則 (これを制約規制という) を使う。下記は、制約規制の一覧である ([Mukai 90(2)] を参照)。

(1) $x = x$,

(2) $x = y$ ならば $y = x$,

(3) $x = y$ かつ $y = z$ ならば $x = z$,

(4) $(a_1, b_1) = (a_2, b_2)$ ならば、$a_1 = a_2$ かつ、b_1 の任意の元 (l, g_1) に対して $g_1 = g_2$ となるような b_2 の元 (l, g_2) が存在する。

(5) $x \sqsubseteq_{\mathcal{G}} x$,

(6) $x \sqsubseteq_{\mathcal{G}} y$ かつ $y \sqsubseteq_{\mathcal{G}} x$ ならば $x \cong_{\mathcal{G}} y$,

(7) $x \sqsubseteq_{\mathcal{G}} y$ かつ $y \sqsubseteq_{\mathcal{G}} z$ ならば $x \sqsubseteq_{\mathcal{G}} z$,

(8) $x \cong_{\mathcal{G}} y$ かつ $x \sqsubseteq_{\mathcal{G}} z$ ならば $y \sqsubseteq_{\mathcal{G}} z$,

(9) $x \cong_{\mathcal{G}} y$ かつ $z \sqsubseteq_{\mathcal{G}} x$ ならば $z \sqsubseteq_{\mathcal{G}} y$,

(10) $(a_1, b_1) \sqsubseteq_{\mathcal{G}} (a_2, b_2)$ ならば、$a_1 \preceq a_2$ かつ、b_2 の任意の元 (l, g_2) に対して $g_1 \sqsubseteq_{\mathcal{G}} g_2$ となるような b_1 の元 (l, g_1) が存在する。

(11) $x \sqsubseteq_{\mathcal{G}} y$ かつ $x \sqsubseteq_{\mathcal{G}} z$ ならば $x \sqsubseteq_{\mathcal{G}} (y \downarrow z)$,

(12) $y \sqsubseteq_{\mathcal{G}} x$ かつ $z \sqsubseteq_{\mathcal{G}} x$ ならば $(y \uparrow z) \sqsubseteq_{\mathcal{G}} x$.

規則 (4) では、$\cong_{\mathcal{G}}$ の性質 (定義するための条件) について述べ、規則 (10) では、$\sqsubseteq_{\mathcal{G}}$ の性質について述べ、規制 (11) および (12) では、制約の中に 2 つのラベル付きグラフの下限および上限を導入している。$\downarrow$ は、2 つのラベル付きグラフの下限であるから、$(x \downarrow y)$ は、x か y が変数の場合について $z \sqsubseteq_{\mathcal{G}} x, z \sqsubseteq_{\mathcal{G}} y$ であるような変数 z を意味する。$(l, g_1), (l, g_2)$ の間の対応は、存在するとすれば一意であることに注意する必要がある。

与えられた制約に制約規則を適用することによって、制約規則の下で閉じた制約 (原子制約の場合) が得られる。これを、与えられた制約の閉包という。

前述の制約規則の定義から、$\cong_{\mathcal{G}}$ はバイシミュレーション関係についての定義を行うための条件を満たし、$\sqsubseteq_{\mathcal{G}}$ の逆がシミュレーション関係についての定義を行うための条件を満たし、両者でシミュレーションペアを構成する。ただし、アトム (基本オブジェクト) の扱いは異なっている。

前述したように、[Barwise 89] ではアトムの全体は離散的であるのに対して、本稿では基本オブジェクトが定義域のアトムであり、かつ半順序をもっている。しかし本稿では、アトム (基本オブジェクト) を値域とする変数は扱わない。よって、アトムの扱いに関する違いは、$(\sqsubseteq_{\mathcal{G}}, \cong_{\mathcal{G}})$ をシミュレーションペアとして考えるとき何ら問題を引き起こすことはない[5]。

下記は、Mukai[Mukai 90(2)] の単一化補題を本稿の制約用に言い直したものである。

Proposition 2 *(Mukai[Mukai 90(2)]参照)* 下記の 2 つの条項は同値である。

(1) 制約 c は正規閉包を持つ。

(2) 制約 c は $\mathcal{G}$ において解を持つ。

この命題は、本稿では $\mathcal{G}$ 上における制約は決定可能であることを示す。

一般的には与えられた制約の閉包は正規ではないかも知れない。しかし任意の閉包は、閉包を拡張する手続きを用いて拡張したとき、正規閉包を持つことが保証されている (たとえば [Mukai 90(2)]、[Yasukawa and Mukai 90] を参照)。

オブジェクト項の上の単一化は、$\mathcal{G}$ の上での制約の可解性を調べる問題として考えられる。2 つのオブジェクト項の単一化を定義する目的のため、解に下記の制限が課される。

Definition 1 *(単一化の条件)*
単一化は、制約に関する正規閉包が、$g \sqsubseteq (\bot, \emptyset)$ のようなラベル付きグラフ g を含まない場合のみ成功する。

制約の例を考えよう。

$$X \sqsubseteq human[name = N_1], Y \sqsubseteq animal[name = N_2, age = 20] \mid \{N_2 \sqsubseteq string\}$$

これは、$human \preceq animal$ のような 2 つのオブジェクト項 X と Y の単一化に対応する制約の例である。

$$\begin{aligned}
X &\cong_{\mathcal{G}} Y, \\
X &\sqsubseteq_{\mathcal{G}} (human, \{(name, \{N_1\})\}), \\
N_1 &\sqsubseteq_{\mathcal{G}} (\top, \emptyset), \\
Y &\sqsubseteq_{\mathcal{G}} (animal, \{(name, \{N_2\}), (age, \{Z\})\}), \\
N_2 &\sqsubseteq_{\mathcal{G}} (string, \emptyset), \\
Z &\cong_{\mathcal{G}} (20, \emptyset).
\end{aligned}$$

[5] アトムの扱いが問題を引き起こさないことを示すためには、Aczel のラベル付き系 (labeled system) を使用するのが便利なようである。ラベル付き系では、(連立方程式で変数に対応する) 各ノードは、ラベルと呼ばれる一意集合を持つ。"ラベル" という用語は、本稿での使用とは違った用い方をしているので注意が必要である。それは、基本オブジェクトの集合からラベルの集合への 1 対 1 の写像と考えることができる。

この制約から始めると、制約に関する規則と閉包の拡張手続きを適用することによって、下記の方程式と包摂関係が得られる。

$$\begin{aligned}
X &\sqsubseteq_{\mathcal{C}} (human, \{(name, \{N\}), (age, \{Z\})\}),\\
N &\sqsubseteq_{\mathcal{C}} N_1,\\
N &\sqsubseteq_{\mathcal{C}} N_2,\\
X &\cong_{\mathcal{C}} (human, \{(name, \{N\}), (age, \{Z\})\}),\\
N_1 &\cong_{\mathcal{C}} (\top, \emptyset),\\
N_2 &\cong_{\mathcal{C}} (string, \emptyset),\\
N &\cong_{\mathcal{C}} (string, \emptyset).
\end{aligned}$$

この結果は、初期の制約の中のすべての変数が具体化されるので初期の制約が可解であることを示している。これは、また、下記に示す割り当て f が制約の可能な解であることも示している。

$$\begin{aligned}
f(X) &= (human, \{(name, \{(string, \emptyset)\}), (age, \{(20, \emptyset)\})\}),\\
f(Y) &= (human, \{(name, \{(string, \emptyset)\}), (age, \{(20, \emptyset)\})\}),\\
f(Z) &= (20, \emptyset),\\
f(N_1) &= (string, \emptyset),\\
f(N_2) &= (string, \emptyset).
\end{aligned}$$

制約の可解性は、本稿におけるオブジェクト識別性および属性の継承の概念に本質的なものである。

3.4 ラベル付きグラフの上での属性項の解釈

本節では、属性項のラベル付きグラフ上での意味論について説明する。

ここで、オブジェクト項の中と属性指定の中とでは、ラベルの扱いが異なることに注意が必要である。

既に述べたように、オブジェクト項はラベル付きグラフとして解釈される。よって、オブジェクト項中のラベルはグラフ中のアークの名前として使用される。これに対し、属性指定の中では、ラベルは $\mathcal{G}[\chi]$ 上の一意関数と解釈される。たとえば、属性項

$$o[l = x]/[l_1 \rightarrow x_1, l_2 \leftarrow x_2, l_3 = x_3]$$

に中の属性指定は、

$$\begin{aligned}
l_1((o, \{(l, x)\})) &\sqsubseteq_{\mathcal{G}} x_1\\
x_2 &\sqsubseteq_{\mathcal{G}} l_2((o, \{(l, x)\}))\\
l_3((o, \{(l, x)\})) &\cong_{\mathcal{G}} x_3
\end{aligned}$$

と解釈される。これから明らかなように、ドット項 $o.l$ の解釈は、関数 l に引数として o に対応するラベル付きグラフを与えた値である。たとえば、$o[l = x].l_1$ の解釈は $l_1((o, \{(l, x)\})$ である。

具体的に記述のないラベルについては、その値は存在するが、制約はないと考える。たとえば上の例で、l_4 をラベルとすると、その値は下記のような制約となる。

$$l_4((o, \{(l, x)\})) \sqsubseteq_{\mathcal{G}} (\top, \emptyset).$$

以上から、*Quixote* では、すべてのラベルはオブジェクト項に対して定義した関数と解釈される。

4 今後の課題

本稿では、知識ベース / 知識表現言語 *Quixote* の言語仕様と意味論について概説を行なった。

Quixote の特徴は、拡張された属性構造 (オブジェクト項) をオブジェクト識別子として用い、オブジェクト (識別子) 間の順序 (包摂関係) の定義と属性の継承を統一的な扱いと、モジュール識別子の導入による知識ベースの分割・統合ということができる。

また、*Quixote* のオブジェクト項の体系は宣言的な意味論を持ち、オブジェクト項間の演繹的な関係を記述するルールと併せて考えると、*Quixote* は、演繹的パラダイムとオブジェクト指向的パラダイムの融合の実現に対する1つの解として見ることもできる。

また、オブジェクト項の集合も属性値として許しており、Hoare 順序により、その意味を与えている。

現在、*Quixote* の第1版のの処理系の試作をほぼ完了している。以下、現段階での *Quixote* の問題点と今後の課題について、言語仕様、実装技術、応用の3つの観点から述べる。

まず、言語仕様に関しては、

- 包摂関係の部分的 / 動的な定義

  基本オブジェクト間の順序をモジュールごとに変更したり、動的に追加したりといったことが必要になる場合が考えられる。

- 関数型の導入

  現在、基本オブジェクトと基本オブジェクト間の包摂関係は宣言的に定義される (データとして与えられるもの) としているが、現実には、判定のための関数 (手続き) を用意する必要がある。

- ルールの追加 (assert)

  現時点では、追加はファクトに限られているが、一般にはルールの追加も必要と考えられる。

などの点が課題として残されている。

また、現在の *Quixote* では、制約理論としてオブジェクト項間の包摂関係のみを扱っているが、

- 制約理論 / 領域の拡大 (disjunction, negation を含む。有理数系、 temporal logic、 etc.)

も応用問題に応じて考えていく必要がある。

言語処理系の実装技術に関しては、第1版処理系に関して

- ルールのコンパイル手法の拡充

  Quixote では、属性値の一意性、属性値の継承に関連して、複数のルールの組合せを考慮する必要があるが、第1版では、単純なルールの組合せによる前処理を行なっているだけである。今後、具体的なルールの解釈処理のアルゴリズムとともに、処理系の実行効率も考慮したコンパイル方式を検討する必要がある。

- 属性継承の計算のための効率的アルゴリズム

  オブジェクト項間の包摂関係にしたがって、属性値に関する制約が継承されるが、基本的にはオブジェクト項のなす束構造を走査するような処理が必要とされる。現在は、素朴なアルゴリズムを設定しているが、部分的な制約を束構造上に伝播するような効率的アルゴリズムが必要とされる。

- モジュール間のルール継承の効率的アルゴリズム

  現在は、ルールの追加は許されていないため、ルールの継承は事前評価可能であるが、実際には、モジュール識別子が変数を含む場合が考えられ、ルールの継承を動的に行なう必要があり、効率的なアルゴリズムが必要とされる。

- データベース管理システム Kappa とのインタフェース

Kappa のデータモデルである非正規関係を *Quixote* のオブジェクト項に変換することで対応することとなるが、*Quixote* と Kappa における集合の扱いが異なるため、集合に関する変換に1対1対応が付かないという問題が残されている。

応用面では、分子生物学データベース、判例データベースと法的推論、文生成プランニングを題材として、第1版言語仕様に従った記述実験を進めてきたが、

- 例題の実行・テスト
- 言語仕様および処理系の feasibility の確認
- 言語仕様の修正・拡大

を行なうことが今後の課題である。

参考文献

[Aczel 88] P. Aczel: *Non-Well-Founded Set Theory*, CSLI Lecture Notes No. 14, 1988.

[Barwise 89] J. Barwise: "AFA and the Unification of Information", in *The Situation in Logic*, CSLI Lecture Notes No. 17, 1989.

[Jaffar and Lassez 87] J. Jaffar and J.-L. Lassez: "Constraint Logic Programming", In *Proceedings of the 14th ACM Symposium on Principle of Programming Languages*, 1987.

[Morita, et al 90] Y. Morita, H. Haniuda, and K. Yokota, "Object Odentity in *Quixote*", in *Proc. of SIGDB and SIGAI of IPSJ*, Oct., 1990.

[Mukai 90(1)] K. Mukai: "Coinductive Semantics of Horn Clauses with Compact Constraint", *ICOT Technical Report*, TR-562, 1990.

[Mukai 90(2)] K. Mukai: "CLP(AFA): Coinductive Semantics of Horn Clauses with Compact Constraint", *The Second Conference on Situation Theory and Its Applications*, Kinloch Rannoch Scotland, Sep., 1990.

[Yokota and Nishio 89] K. Yokota and S. Nishio: "Towards Integration of Deductive Databases and Object-Oriented Databases: A Limited Survey", *Advanced Database System Symposium*, Kyoto, Dec.7-8, 1989.

[Yasukawa and Mukai 90] H. Yasukawa and K. Mukai: "Constraints over Complex Objects", *in preparation*.

[Yasukawa and Yokota 90] H. Yasukawa and K. Yokota: "An Overview of a Knowledge Representation Language *Quixote*", *draft*, 1990.