TM-0431

レコードと演繹

横田一正

January, 1988

ICOT

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

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

レコードと演繹

Deduction Based on Record Structures

横田 一正
(財) 新世代コンピュータ技術開発機構

1. はじめに

構造データの表現とそれに基づいた言語あるいは論理体系の研究が盛んになってきている[Ait-Kaci 84,Mukai87, Maier 86, Bancilhon 86, Rounds 86, Abiteboul86]. 構造データとは, レコード, オブジェクト, 複合オブジェクト, エンティティ(実体), 状況, 分類データなどである. この応用領域は, データベース, 知識ベース, 自然言語のほか, 種々の知識処理の領域に及んでいる. 本発表では, 現在ICOTの知識ベース・プロジェクトの1つ(Kappa [横田87])で, 非正規関係の演繹機能の実現のための表現言語として研究しているCRL[Yokota 87]を中心にして, それらの一般的な形式的表現を議論したい.

2. オブジェクトとレコード

オブジェクトの概念はプログラム言語や知識表現などでひじょうに有用な'もの'として利用されているが, ここでは, 実世界のさまざまな構造をもったデータという視点からオブジェクトを考える(この意味だけからはでは'エンティティ'といったほうがよいかもしれない). そしてそれを名前づけと制約の集合として, つまりレコード構造の1つとして表現することを考えよう. そのために制約の識別子として属性名 (ラベルあるいはスロット)を考える. いくつかの表現が考えられる.

(1) 単純なレコード構造

'*'を結合律(A), 交換律(C), べキ等律(I) を満たす演算子とし, 集合記法の代りに'*'の結合として表現すると

〈オブジェクト〉 ::= 〈制約〉{'*'〈制約〉}*
〈制約〉 ::= 〈属性名〉'/'〈属性値〉

はオブジェクトを平坦なレコード構造として表現している (ただしオブジェクトが制約の組合せとして定義されるとき同一の属性名は使用できないとする). たとえば人というオブジェクトを特定するために, 名前, 年齢, 住所という制約の組合せで表現するとすれば,

名前/"中浜"*年齢/"28"*住所/"京都"

の制約集合(オブジェクト)はある人(オブジェクト)を表わしている. ここで属性値は内部構造を持たないオブジェクト(個体)に対応した名前であると考えることができる.

(2) 階層構造の導入

(1) は平坦な構造しか表現しえないので, '構造'の表現としてはあまりに制限されている. そこでオブジェクト間に階層の導入を行う:

〈オブジェクト〉 ::= 〈属性値〉 | 〈制約〉{'*'〈制約〉}*

〈制約〉　　::= 〈属性名〉'/' '('〈オブジェクト〉')'

これに従えば，

名前/(姓/"中浜"*名/"鉄")*年齢/"28"*住所/"京都"

のように部分構造を表現したり，

名前/"鉄"*長男/(名前/"明"*長男/(名前/…))

のようなオブジェクトの階層構造を表現できる．

(3) 集合の導入

(2) の例で複数の子供を持った人を表現する場合，各子供を異なった属性として扱うこともできるが，それらを同種の属性として扱う場合集合の導入が必要となる．そこで

〈オブジェクト〉 ::= 〈属性値集合〉 | 〈制約〉{'*'〈制約〉}*
　　| '{'{〈オブジェクト〉{','〈オブジェクト〉}*}'}'

〈制約〉　　::= 〈属性名〉'/' 〈オブジェクト〉

と定義する．すると

名前/{"鉄"}*子供/{名前/{"明"}, 名前/{"薫"}}

あるいは

名前/{"鉄"}*趣味/{"読書","散歩","昼寝"}

のように表現できる．ただし記法上単一要素の集合の括弧は省略ことにする．

オブジェクトの内部構造の操作のために以下の記法を導入する：

0=a1/01*…*an/0nのとき0//〈ai〉=0i,

0={01,…,0n}のとき0//〈a〉={01//〈a〉,…,0n//〈a〉},

0//〈a1,a2,…,an〉=(0//〈a1〉)//〈a2,…,an〉.

次にオブジェクト間に部分関係≧'を定義する：

i) ある属性名列pがあって01//p=02なら01≧'02,

ii) 01⊇02なら01≧02,

iii) 01={011,…,01n},02={021,…,02m}なら任意の02jに対して01iが存在し，
01i≧'02jなら01≧'02.

01≧'02 のとき02を01の部分オブジェクトという．属性名列が空の場合0//〈〉=0 であるので，この部分オブジェクト関係は明らかに半順序になっている．

3. オブジェクトのセマンティクス

上のオブジェクトの意味領域として部分タグ木 (PTT) の集合を考える．つまりCIL [Mukai 87]同様PTTは，

T: 属性名列の木領域
　(属性間の接合演算子を'.'とする),

f: Tから属性値集合への部分関数

の対(T,f) で定義する．部分木の操作のために属性名列によるPTT t=(T,f)の部分木を

t//〈a〉=(a.T,a.f)
　　={x | ∃z,y∈T x.z=a.y}, {(a.l,v) | (l,v)∈f})

t//〈a1,a2,…,an〉=(t//〈a1〉)//〈a2,…,an〉

と定義する．

PTT (T1, f1), (T2, f2)の合併演算'+'はf1∪f2が関数であるときにのみ(T1, f1)+(T2, f2)=(T1∪T2, f1∪f2) と定義する．PTTの集合間の演算は

$$\{t1, \cdots, tm\} + \{u1, \cdots, un\} = \{v \mid v = ti + uj \text{が定義可能}\}$$

と定義する．そしてt1+…+tnをΣ tiと書くことにする．

オブジェクトの意味としてPTT の集合を割り当てる．つまりオブジェクトOのPTTの集合への割当てをI[O]と簡単に記し以下のように定義する('<>'は空属性とする)：

$$\begin{aligned}
I(O) &= \{(\{<>\}, \phi)\} && \text{if } O = \{\}, \\
&= \{(\{<>\}, \{(<>, c)\})\} && \text{if } O = c, \\
&= \sum_{i=1}^{n} ai.I[Oi] && \text{if } O = a1/O1 * \cdots * an/On, \\
&= \bigcup_{i=1}^{n} I[Oi] && \text{if } O = \{O1, \cdots, On\}.
\end{aligned}$$

オブジェクトO1, O2に対して，O1$\supseteq$'O2, I[O1]={t1, …, tn}とすると，ある属性名列p1, …, pnが存在し，

$$I[O2] = \{ti//pi \mid ti//pi \text{が定義可能}\}$$

となる．

そしてPTT間の包含関係$\supseteq$'を

$$(T1, f1) \supseteq' (T2, f2) \Leftrightarrow T1 \supseteq T2 \text{かつ} f1 \supseteq f2.$$

PTTの集合間の包含関係を

$$\{t1, \cdots, tn\} \supseteq' \{u1, \cdots, um\} \Leftrightarrow \forall uj, \exists ti \ ti \geq' uj$$

と定義する．(3) のオブジェクトの集合は，上の割当てによってPTT の集合に対応する．したがってオブジェクト間のequalityと包含関係が自然に導入される：

$$a1/O1 * a2/O2 * a3/O3 = a3/O3 * a2/O2 * a1/O1,$$

$$a1/\{a11/O11, a12/O12\} * a2/O2 = a2/O2 * a1/\{a12/O12, a11/O11\},$$

$$a1/\{O11, O12, O13\} * a2/O2 \supseteq a1/\{O11, O13\} * a2/O2.$$

(1), (2) についても(3)の特殊な場合(singleton)と考えることができる．(3) で定義される空集合{}は関数が未定義であることに対応する．

4. オブジェクトとデータベース

データベースは，実世界のオブジェクトに対応する表現の集合であり，ここでは上の'オブジェクト'の集合，つまりデータベース自身1つのオブジェクトと考える．これを特徴づけるために2つの概念を導入する．

まずデータベースはスキーマとそれに対応する実体から構成されるが，ここではスキーマをクラスに，実体の要素をインスタンスに対応させる．2節の(1), (2), (3)はそれぞれ以下のように再定義する：

(1') 〈オブジェクト〉 ::= 〈クラス〉[':'〈インスタンス〉]
〈インスタンス〉 ::= 〈制約〉{'*'〈制約〉}*

〈制約〉　　::= 〈属性名〉'/' 〈属性値〉

(2') 〈オブジェクト〉 ::= 〈クラス〉[':'〈インスタンス〉]
〈インスタンス〉 ::= 〈属性値〉 | 〈制約〉{'*'〈制約〉}*
〈制約〉　　::= 〈属性名〉'/' '('〈オブジェクト〉')'

(3') 〈オブジェクト〉 ::= 〈クラス〉[':'〈インスタンス〉]
〈インスタンス〉 ::= 〈属性値集合〉 | 〈制約〉{'*'〈制約〉}*
| '{'〈オブジェクト〉{','〈オブジェクト〉}*'}'
〈制約〉　　::= 〈属性名〉'/' 〈オブジェクト〉

たとえば

人:(名前/文字列:"中浜"*年齢/整数:"28")

はオブジェクトである．記法の容易さから"クラス:" は明記の必要がないときは省略する．

クラスの役割としては，インスタンスの制限，その継承関係の利用，および同一クラスのインスタンスに対する共通の操作の提供を意図している(ここでは第3の役割については触れない)．クラス間には半順序≧があり，特殊なクラス⊤(top)，⊥(bottom) を導入し，クラス全体はそれによって完備束であることを仮定しよう．クラスの意味としてPTT の集合を割り当てる．そして拡張されたオブジェクトC:Oの意味を

I[C:O]=I[C]∩I[O]

と定義する．

このオブジェクトに対しても属性名列による内部構造への参照を上と同様に定義できる:

O=C:(a1/O1*…*an/On)のとき O//<ai>=Oi,

O=C:{O1,…,On}のとき O//<a>=C:{O1//<a>,…,On//<a>},

O//<a1,a2,…,an>=(O//<a1>)//<a2,…,an>.

そしてオブジェクト間には，クラスを持たないオブジェクトと同様に２種類の半順序を定義することができる:

① 包含関係⊇'

これはPTTの集合間の半順序⊇'に対応した関係で

C1:O1⊇'C2:O2 ⇔ I[C1:O1]⊇'I[C2:O2]

と定義する．

② 部分関係≧'

これは部分オブジェクトを定義する:

i) ある属性名列pがあってO1//p=C2:O2ならば

O1≧'C2:O2,

ii) C1≧C2かつO1⊇O2ならC1:O1≧'C2:O2,

iii) C1≧C2かつO1={O11,…,O1n}, O2={O21,…,O2m} であるとき，任意のO2jに対してO1iが存在しO1i≧'O2jならC1:O1≧'C2:O2.

次にオブジェクトの冗長性であるが，これは包含関係によって定義できる．任意のオ

ブジェクトC:Oに対しI[C:O]={t1,…,tn}とすると，

ti⊐'tj

なるi,j(i≠j)が存在することを言う．この冗長性はC:Oのどの部分オブジェクト {C1:O1,…,Cm:Om} も，i≠jならばCi:Oi⊐'Cj:Oj なる関係を許さないことに対応する．たとえば

人:{名前/"中浜"*年齢/"28"*住所/"京都",
名前/"中浜"*年齢/"28"*住所/{}}

なるオブジェクトには冗長性があるが，

人:{名前/"中浜"*年齢/{}*住所/"京都",
名前/"中浜"*年齢/"28"*住所/{}}

には冗長性はない．

データベースは冗長性のない１つのオブジェクト

DB:{C1:O1,…,Cn:On}

として定義される．ただしこのオブジェクトは

ｉ）i≠jならCi≠Cj，

ii）O1,…,On中の任意のクラスはC1,…Cn の中に現れる，

iii）任意の部分オブジェクトには冗長性はない

の３つの条件を満足している．文字列や整数というデータ型のクラスも上のCiとして現れる．そのクラスはちょうど領域閉包としての役割を果たす．

5. 非正規関係とオブジェクト

非正規関係は直観的には以下のように定義される：

R⊂E1×…×En, Ei::=D|P(R'),

ただしDは属性値集合，R'は他の非正規関係，P(*) はベキ集合を表わしている．この非正規関係は，正規関係

R⊆D1×…×Dn

の要求する第1正規形(2節(1)の形式)がCAD/CAM, OAなどの種々の応用に不向きなことから，その条件を否定することを出発点にしている．このデータ・モデルについては，構造データの表現の容易さはもちろん，データの冗長性の減少，処理の効率化などの長所を多く持っており，最近多くの研究が行われている．

上のオブジェクトの定義(1),(1')は正規レコード，(2),(2')は列ネスト（属性の階層構造）の非正規レコード，(3),(3')はさらに行ネスト(集合)を持った非正規レコードに対応している．上のオブジェクトの定義に対応させれば

人:　{名前/"中浜"*年齢/"28"*家族/妻子}
妻子:(妻/"夏"*子供/{名前/"明",名前/"薫"})

と表現できる．埋め込まれた(非正規)関係は別の独立したオブジェクトとなる．

この非正規関係をオブジェクトとして表現することによって，その意味はPTT の集合となる．たとえば

DB1:{a/c1*b/{c2,c3}},
DB2:{a/c1*b/c2, a/c1*b/c3}

の２つのデータベースがあったとき，上のPTT 割当ては同じ意味を持ったデータベースであることを示している．このような集合の分解，合成によるデータベースの変換をそ

れぞれ行アンネスト，行ネスト操作と呼ぶ．一般の非正規レコードは，このネスト操作がデータベース全体に渡って'一様に'行われることを仮定しない．たとえば

DB3:{a/{c1,c2}*b/c3, a/c2*b/{c4,c5}}

のように,行ネスト/行アンネスト操作だけでは構成できないデータベースを作ることができる．この操作はちょうど部分因数展開が一意的でないことに対応している．非正規関係の(ネストを持った)正規化の研究もあるが，ユーザ言語という視点から考えれば，実世界のオブジェクトを表現するとき，このネストの順番を規制するのは不自然であろう．そこでここで定義したように， 行ネスト/行アンネストから独立した意味を与えることは,非正規関係のデータベースを考えるときに重要である.

6. CRL

データベースに対する操作は，データベース・オブジェクトから，

- クラスの操作,
- 制約の操作,
- 包含関係と部分関係によるオブジェクトの取出し

の組合せと考えられる．オブジェクトからなるデータベースをモデル論的にとらえ，それに対する(関係)演算を定義したものとして[Bancilhon 86]と[Maier 86]がある.

しかしここではデータベースを形式理論の一部として捉える証明論的アプローチを考えたい．関係モデルの場合，レコードを通常の一階述語と対応させ，データベースを一階理論と考えることによって，演繹データベースとして形式的な拡張が可能になる．そしてその表現としてPrologを用いている．ところが非正規関係の場合，上で述べた行ネスト操作があるために通常の一階述語として考えるのは難しい．たとえば

DB1: a/c1*b/c4, a/c2*b/c4, a/c2*b/c5,
DB2: a/{c1,c2}*b/c4, a/c2*b/c5,
DB3: a/c1*b/c4, a/c2*b/{c4,c5}

の３つはいずれも同じ意味を持ったデータベースであるが，a/{c1,c2}*b/X?という質問に対して，(属性,値)-対を述語記号，関数記号に対応させると，通常のPrologではX を求めることはできない.

そこで，CRL は現在ICOTで研究開発中のKappa システム[横田 87] の中で，上記のような非正規関係の形式的枠組として検討している言語である．現在はクラスの導入は行ってなく，

(3") 〈オブジェクト〉 ::= 〈制約〉{'*'〈制約〉}* | '{'{〈属性値〉{','〈属性値〉}*}'}'
〈制約〉 ::= 〈属性名〉'/' 〈オブジェクト〉

から構成される非正規関係の部分クラスに限っている．そしてデータベースはそのようなオブジェクトの集合である.

そのデータベースに対して演繹データベースと同じく証明論的な枠組を提供するために，NAV を以下のように定義する:

i) 属性値の集合あるいは変数はNAVである.
ii) a1,…,an が属性で(i≠jならai≠aj)，t1,…,tnがNAVであれば

(a1/t1*…*an/tn)もNAVである.
この意味は変数のPTTへの割当てを追加して3節と同様に定義する.

このNAV間の単一化は, PTTの集合の合併操作'+' に対応させて定義し, 単一子を求めることができる (現在はoccur checkを行っている). 通常の単一化と違うのは集合値の場合それらの共通部分集合の非空の任意の部分集合が単一子を構成できる点である. したがってその最大のものを '損失のない' 単一化としていることである. mguは一意的に決定できる.

NAVに基づいたプログラム節は, NAVとNAV集合の対であり, Prolog同様p←p1,…,pnと書かれる. たとえば

親/{"薫"}*子供/{"孝","闇"}.
親/{"徹","夏"}*子供/{"薫"}.
先祖/X*子孫/Y←親/X*子供/Y.
先祖/X*子孫/Y←親/X*子供/Z,先祖/Z*子孫/Y.

はCRLプログラムである. このモデルはPTTの集合として定義すると, 最小モデルは一意的に構成できる. たとえば上のプログラムの場合, PTT

({<>,親,子供},{(親,u),(子供,v)}),
({<>,先祖,子孫},{(先祖,u),(子孫,v)})

をそれぞれpc(u,v),ad(u,v)と略記すれば

{pc("薫","孝"),pc("薫","闇"),pc("徹","薫"),pc("夏","薫"),
ad("薫","孝"),ad("薫","闇"),ad("徹","薫"),ad("夏","薫"),
ad("徹","孝"),ad("徹","闇"),ad("夏","孝"),ad("夏","闇")}.

が最小モデルとなる. ゴールはNAVの集合で←g1,…,gnと書く.

手続き的意味論としてSLD-導出法を定義するが, 通常のSLD-導出法とは集合の処理で異なっており, 以下の付加的手続きが必要となる:

i) 残ゴール
ゴール中のNAV が他のプログラム節と単一化できたとき, それに含まれる集合で共通部分に含まれない部分から新たにNAVを生成し, それを新たなゴールとする.

ii) 再置換
いったん単一化によって具象化された集合が, 他のゴールの単一化でさらに縮小されたとき, それを関連の集合に伝播する.

たとえば以下のデータベース

a/c1*b/{c3,c4}. (1)
a/c2*b/c3. (2)
a/c2*b/c4. (3)

に対し, ゴール←a/{c1,c2}*b/Xが与えられたとき, まず(1)との単一化により新ゴール

←a/c1*b/<X>:{c3,c4},a/c2*b/<X>:{c3,c4}

が生成される('<X>:' は再置換のためのマークである). 'a/c2*b/<X>:{c3,c4}' が生成された残ゴールである. 次にその残ゴールが(2)と単一化され, その情報が第1のゴールに伝播され

←a/c1*b/<X>:c3,a/c2*b/<X>:c3

として成功する. つまり(1)との単一化でいったん成功した<X>:{c3,c4} が(2)との単一化で再置換され, X=c3 となる. そして同様にしてX=c4を求めることができる.

CRLの特徴として以下の点を挙げることができる:

i) CRLプログラムはデータ独立である．つまりスキーマの(属性入替え，属性追加，などの)変更によってプログラムの変更は不要である．

ii) CRLデータベースは行ネスト/行アンネスト操作から独立で，質問はデータベースのネストの順番を意識する必要はない．また一般の非正規レコードを扱える．

iii) 処理対象の非正規関係が一様にネストされていた場合，正規関係に比べて効率的な質問処理が可能である．

iv) ユーザは質問として，スキーマ全体ではなく，必要な情報のみを書くことができる．

CRL はオブジェクトの1つとしての非正規関係の，さらに小さなクラスを対象に始めたばかりの言語であり，今後さらに広いクラスへの拡張やデータベースとしての最適化，あるいは普遍関係(universal relation)との関係など，検討を続ける予定である．

7. おわりに

Kentは，実体(オブジェクト)とレコード構造の関係について考察し [Kent 79]，特に関係モデルでの正規レコードについて，レコードの持つ均質性，記述(description) の扱い，型の扱い，関連情報の表現などについて問題点を指摘している．ここでの制約の集合としてのレコード表現は，その大部分の制限を克服している．しかしデータベースは量の問題に対する工学的技術という側面を持っており，均質性や記述の簡略化は，レコード構造の制限というより，そのための制限ということができる．したがってレコード構造とその演繹を行うシステムのデータベース・システムとしての実装は，今後の課題である．

参照文献

[Abiteboul 86] Abiteboul, S. and Hull, R. "Restructuring of Complex Objects and Office Forms", ICDT'86, LNCS-243, pp.54-72, 1986

[Ait-Kaci 84] Ait-Kaci, H., "A Lattice Theoretic Approach to Computation Based on a Calculus of Partially Ordered Type Structures", Dissertation, Univ. of Pennsylvania, 1984

[Bancilhon 86] Bancilhon, F. and Khoshafian, S., "A Calculus for Complex Objects", PODS'86, pp.53-59, 1986

[Kent 79] Kent, W., "Limitations of Record-Based Information Models", TODS, vol.4, no.2, pp.107-131, 1979.

[Maier 86] Maier, D., "A Logic for Objects", Proc. of the Workshop on Foundation of Deductive Database and Logic Programming, pp.6-26, 1986

[Mukai 87] Mukai, K., "Anadic Tuples in Prolog", ICOT-TR-239, 1987.

[Rounds 86] Rounds, W.C. and Kasper, R., "A Complete Logical Calculus for Record Structures Representing Linguistic Information", 1986

[Yokota 87] Yokota, K., "Deductive Approach for Nested Relations", ICOT-TR, to appear.

[横田 87] 横田, "知識ベース管理基本ソフトウェアKAPPA", 第5回第5世代コンピュータに関するシンポジウム, June, 9-10, 1987.