

データ変換プログラムのモデル化と構築手法の提案

Proposal of Modeling and Construction Methods for Data Transformation Programs

森嶋 厚行[♡]

Atsuyuki MORISHIMA

現在のソフトウェア開発において、データ統合の重要性がますます高まっている一方で、開発コストの大きな負担となっている。このコスト削減を目指すことは重要な研究課題であるといえる。データ統合処理の重要な部分問題の一つに、データ変換の問題がある。データ変換とは、与えられた二つのスキーマ間のマッピングを行い、さらにインスタンスを移行するものである。我々は、データ変換プログラムの構築コストを下げる問題に着目し、この問題を議論するためには、データ変換プログラム構築のための形式言語が有効であると考えている。本稿では、データ変換モデルとそれをを用いたデータ変換プログラムの構築手法を提案する。

Today, many software development projects involve data integration. Since implementing data integration costs a lot, reducing the cost offers significant benefits. The data transformation is a key issue in data integration, which is a process of matching two given schemas and transforming instances of one schema into instances of the other. We focus on the problem of reducing the development cost of data transformation programs. We believe that having a formal language would be effective to discuss the problem. This paper proposes a data transformation model as a possible formal basis for the problem and a construction method based on the model.

1. はじめに

近年、ソフトウェアシステムの開発において、データ統合が重要な問題となっている。具体的には、アプリケーションに必要な複数の情報源からのデータ統合や、旧データベースと新データベースの混在した環境でのシステム開発等がある。各種調査によると、IT 予算の 40～60%は統合関連に関して費やされると言われている。一般に、データ統合の実現は簡単ではなく、大きな開発コストがかかる。この労力を下げることができれば、ソフトウェア開発コストの低減に寄与できると考えられる。

我々は、データ統合の重要な要素技術の一つであるデータ変換の問題に着目する。データ変換の問題とは、入力としてスキーマ A とそのインスタンス $I(A)$ 、スキーマ B が与えられたとき、 A と B 間でのマッピングを行い、 $I(A)$ をスキーマ B のインスタンス $I(B)$ に変換するものである。最もわかりやすい例としては、ローカルなデータベースのデータを、統合スキーマにあわせるためのデータ変換がある。他にも、データ統合に関連する様々な局面でデータ変換は必要となる。例えば、Web サービスシステムの開発に当たっては、あるソフトウェアが出力したデータを、他のソフトウェアが入力として受け付け可能な、もしくは処理可能なデー

[♡] 正会員 筑波大学 知的コミュニティ基盤研究センター 知の共有基盤研究部門 / 図書館情報メディア研究科
amorishima@acm.org



図 1 データ変換の写像によるモデル化

Fig. 1 Modeling a Data Transformation as a Mapping

タに変換しなければならない。問題は、データ変換の実装をどう効率良く行うか、言い換えると、データ変換プログラムをいかに効率良く構築するかである。

我々は、データ変換プログラム構築の理論、技術、道具などの発展のためには、この問題を議論するための形式言語の開発が有効と考えた。その効果としては次のようなものが考えられる。(1) 形式的定義を持ち、性質 (properties) が知られているポキャブリとして、データ変換に関する問題定義および議論を明確にするための道具として利用する。(2) 各種データ変換技術の統合を実現するメタな枠組みとして利用する。(3) データ変換の問題をより扱いやすい小問題に分割するための道具として利用する。

本稿では、この目的のために、いくつかの望ましい性質を持ったデータ変換モデルを提案する (性質については 5 章で説明する)。本提案モデルのポイントは、(1) データ変換を写像としてモデル化すること、(2) 基本的な写像として、意味写像の概念を導入すること、(3) データ変換写像 (プログラム) 構築の作業を、写像の操作としてモデル化すること、である。

関連研究。これまで、データ変換プログラム構築支援 [4][7]、データ変換記述言語 [2] や、Schema Evolution[1] などの研究が行われてきた。これらに対する本研究の新規性は次の通りである。

(1) 以前に提案されたデータ変換記述言語は「データ変換そのものを記述する言語」であり、言語で提供されている操作の対象はデータ、もしくはスキーマであるが、本モデルは「データ変換プログラムの操作を記述するモデル」であり、操作の対象は変換プログラム (写像) である。すなわち一段メタなレベルにあり、この点が決定的に他の言語と異なる。データ変換の構築が近年の情報システム開発のボトルネックの一つであり、取り組むべき重要な問題であるという観点から、データ変換を“first class citizen”として定義した操作系であるという点に新規性がある。(2) 本研究ではデータ変換プログラムの構築過程を記述するためのポキャブリを用意し、「データ変換プログラムの構築プロセスを支援」する事が目的であり、そのために必要なモデル、データ変換プログラムの各種性質、等の議論を行う。(3) スキーマの構成要素間のマッピングを直接行うのではなく、いったん意味領域を介した変換というアプローチを採用する。これにより、スキーマレベルに直接現れない意味領域の構成要素や制約を考慮した変換が容易になり、単純なスキーママッピングと比較して、より現実的なデータ変換の問題への対応が可能になると考えている。

2. 写像としてのデータ変換

本モデルでは、データ変換を写像としてモデル化する (図 1)。このとき、データ変換の問題は次の様に記述することができる。

入力: 変換前のスキーマ sch_a 、変換前のインスタンス $I(sch_a)$ 、変換後のスキーマ sch_b 。

出力: $I(sch_a)$ に写像 F を適用した結果 $F(I(sch_a))$ 。

したがって、写像 F の発見がデータ変換の問題の鍵である。

2.1 問題設定

データ変換のモデル化にあたり、次のように問題設定を行う。

- データを XML としてモデル化する。理由は、XML がデータ交換用フォーマットの標準として確立されており、多くの種類のデータが、XML での表現形式を持つからである。
- 単純化のため、次の制約をつける。(1) スキーマ定義は再帰的な定義 (section=(section*)) など) を含まないものに限定する。これらは入れ子の要素名に別名 (section1, section2 など) をつけることで実質的には対応できる。(2) 順序無し XML を扱う。すなわち、XML 要素間の順序関係を考慮しない。

```

univ=(dept*,person*)
dept=(@did:ID, dname)
person=(pid, @type,
        name, @did:IDREF)
univ=(dept*)
dept=(did, dname, people)
people=(prof*,student*)
prof=(pid,name)
student=(sid, name)
    
```

図 2 2 種類の XML スキーマ Sch_A (a) と Sch_B (b)
Fig. 2 Two XML schemas Sch_A (a) and Sch_B (b)

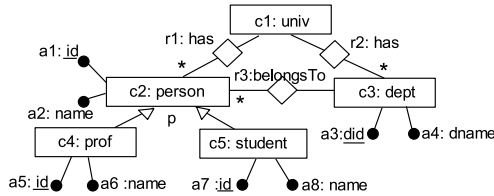


図 3 クラス図 cd_1
Fig. 3 Class Diagram cd_1

データ変換問題の具体例を次に示す。

例 1. 図 2(a)(b) は、2 つの大学における XML データのスキーマ Sch_A と Sch_B である。これらの構造は異なるものの、本質的にはどちらも図 3 のクラス図 cd_1 で表現される同種の情報を表す (Sch_A の XML では、person の type 属性が “s” の時 student, “p” の時 prof を表す)。この時、 $I(Sch_A)$ から $I(Sch_B)$ への写像 $F_{AB} : Sch_A \rightarrow Sch_B$ を求めたい。

2.2 意味写像の導入

我々は、データ変換の問題に対して、意味領域を介した変換というアプローチを取る。具体的には、XML から意味領域への写像 (以下、意味写像) を利用する (図 4)。ここで意味領域とは、直観的にはデータベース設計における概念モデルの層に対応する。すなわち、実際のデータ表現の詳細を捨象し、概念レベルでどのようなデータを保持しているかを表す領域である。意味写像は $f : sch \rightarrow cd$ と表す。この写像の定義域は、XML スキーマ sch を満たす XML インスタンスの集合 $dom(sch)$ である。また、値域はクラス図 cd (例えば図 3) が規定するインスタンスの集合 $dom(cd)$ である。値域については 2.4 節で説明する。

意味写像を利用すると、データ変換は図 5 の様にモデル化することができる。ここで f および g はそれぞれ意味写像である。

意味領域を介した変換には次の利点がある。(1) 意味領域の知識 (オントロジなど) を変換写像の構築に利用しやすい。(2) スキーマ間の関係だけでは自明でないもの (例 1 の prof 等) を扱いやすい。

2.3 クラス図

クラス図は、直観的には図 3 のような、クラス、関連¹、属性をノードとするグラフである。

2.4 クラス図のインスタンス

型としてクラス図 cd が与えられたとき、型 cd の値 $v \in dom(cd)$ を次のように定義する。

定義 1. クラス図 cd の値 $v \in dom(cd)$ はタグ付きレコード

$$\{n_1 : R_{n_1}, n_2 : R_{n_2}, \dots, n_m : R_{n_m}\}$$

である。ただし、 $n_1 \dots n_m$ はクラス図 cd 中のノード (クラス、属性、関連に対応する各ノード) である。また、 R_{n_i} は n_i の種類に応じて決まる次のいずれかのリレーションである。

n_i がクラス c の場合: R_{n_i} はクラス c の各インスタンスを一意に決める値を持つ単項リレーション $R_c(attr)$ である。図 3 中の c_3 ノード (dept クラス) に対応する R_{c_3} の例を図 6(a) に示す。

関連 r の場合: $R_{c_1} \dots R_{c_n}$ を r に接続されるクラスとする。このとき、 R_{n_i} はこれらのクラス間の関連を表すリレーション $R_r(attr_1, \dots, attr_n)$ である。図 6(b) に R_{r_3} の例を示す。

¹ 汎化は関連ではない。

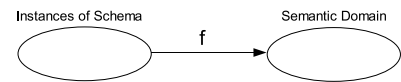


図 4 意味写像

Fig. 4 Semantic Mapping



図 5 意味領域を介したデータ変換

Fig. 5 Data Transformation through Semantic Domain

クラス c の属性 a の場合: $R_c(attr)$ を c のリレーションとする。このとき、 R_{n_i} は c を表す値と a の値の組を表す 2 項リレーション $R_a(attr, attr_a)$ である。ここで、 R_a は関数従属性 $attr \Rightarrow attr_a$ を満たす。また、 $R_c = \pi_{attr}(R_a)$ が成立する。図 6(c) に R_{a_8} の例を示す。 n 項関連 r の属性 a の場合は $(n + 1)$ 項リレーションになる。

さらに、これらのリレーション群が、 cd に記述された全ての制約 (キー制約や多重度など) を満たしていなければならない。

2.5 意味写像の定義

意味写像 f は $w \in dom(sch)$ なる XML インスタンス w から値 $v \in dom(cd)$ を計算する式である。

定義 2. 意味写像 $f : sch \rightarrow cd$ は次の形式を持つ。

$$\{n_1 : (e_{n_1}, l_{n_1}), n_2 : (e_{n_2}, l_{n_2}), \dots, n_m : (e_{n_m}, l_{n_m})\}$$

ここで、 n_i は cd 中の各ノードであり、 e_{n_i} は、 $w \in dom(sch)$ から、 $v \in dom(cd)$ の R_{n_i} を計算する式である。 e_{n_i} の例として、 $f_A : Sch_A \rightarrow cd_1$ (ただし Sch_A は図 2(a)、 cd_1 は図 3 参照) において、 R_{c_3} (図 6(a))、 R_{r_3} (図 6(b))、 R_{a_8} (図 6(c)) を計算する式 $e_{c_3}, e_{r_3}, e_{a_8}$ を次に示す。これらは、XQuery 風の式 (XSM(XML-Semantics Mapping) 式と呼ぶ) で記述する。

```

e_c3: for $k0 in /univ/dept/@did return $k0 as dept;
e_r3: for $x1 in /univ/person, $k1 in $x1/pid, $k2 in
univ/dept/@did where $x1/@did = $k2 return $k1 as per-
son, $k2 as dept;
e_a8: for $x3 in /univ/person, $v1 in $x3/name, $k3 in $x3/pid
where $x3/@type = "s" return $k3 as student, $v1 as name;
    
```

次に、写像定義における l_{n_i} は、 n_i が写像 $f : sch \rightarrow cd$ においてどのような役割を果たすかを表すラベルである。まず、 cd 中のノード n_i が実装ノードであるとは、 n_i に直接対応したスキーマの構成要素が sch に現れる事である。実装ノードにはラベル I を付ける。一方 n_i が概念ノードであるとは、意味領域に対応する概念が存在する事であり、ラベル C を付ける。この 2 つの役割は互いに相反する物ではなく、あるノードが 2 つを兼ねることもあるため、 l_{n_i} は集合値を取る。例えば、 $f_B : Sch_B \rightarrow cd_1$ では prof, student が Sch_B に現れる (図 2(b)) ため $l_{c_4} = \{C, I\}$, $l_{c_5} = \{C, I\}$ であるが、 $f_A : Sch_A \rightarrow cd_1$ では $l_{c_4} = \{C\}$, $l_{c_5} = \{C\}$ である。概念ノードでない n_i は、データ変換写像を構築する過程で現れる。

2.6 単純な意味写像

単純な (simple) 意味写像 $f_{simple} : sch \rightarrow cd$ は、特定の条件を満たす意味写像である。図 7 に単純な意味写像 $f_{A.simple} : Sch_A \rightarrow cd_{Sch_A}$ におけるクラス図 cd_{Sch_A} を示す。直観的には、 f_{simple} の値域は、 sch と同型の構造を持つクラス図 cd_{sch} で表される。 cd_{sch} では、 sch 中の各 XML 要素に対してクラス図中の一つのクラスが対応し、 sch 中の要素の親子関係がクラス間の関連に直接対応する。

3. 意味写像を利用したデータ変換写像の構築

意味写像を導入すると、 $sch_a, I(sch_a), sch_b$ に基づくデータ変換写像 $F : sch_a \rightarrow sch_b$ の構築は、次の 4 段階で出来る。

[1. 意味情報の抽出] まず sch_a からクラス図 cd_a を導出し、意味写像 $f : sch_a \rightarrow cd_a$ を求める。例えば例 1 では、 Sch_A を基

(a) R_{c3}		(b) R_{r3}		(c) R_{a8}	
dept	#pid1	person	#did1	student	name
#did1	#pid2	#pid2	#did1	#pid1	"furukawa"
#did2	#pid3	#pid3	#did1	#pid2	"nakamizo"
	#pid4	#pid4	#did2	#pid4	"ariyama"

図 6 意味領域の値 $v \in dom(cd_1)$ の断片
Fig. 6 Fragment of $v \in dom(cd_1)$

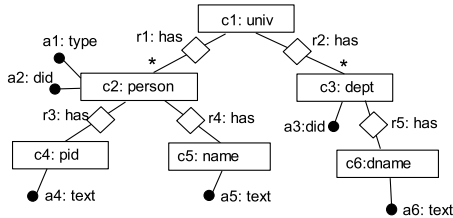


図 7 クラス図 cd_{Sch_A}
Fig. 7 Class Diagram cd_{Sch_A}

に cd_1 を導出し, $f_A : Sch_A \rightarrow cd_1$ を構築する. 一般論として, XML スキーマを設計する際には cd_1 の様な概念モデルを想定しながら XML での表現を決定, という過程をとると考えられる. [1. 意味情報の抽出] は, この過程に対する一種のリバースエンジニアリングを行うものである. 具体的には, まず単純な意味写像 $f_{simple} : sch_a \rightarrow cd_{sch_a}$ を求めた後, f_{simple} を変形することによって f を求める. また意味写像 $g : sch_b \rightarrow cd_b$ も同様に求める. 同じく例 1 では, Sch_B から cd_1 が導出され, $f_B : Sch_B \rightarrow cd_1$ が求められる.

[2. マッチング] 上ではどちらのスキーマからも cd_1 が導出される例 ($cd_a = cd_b = cd_1$) を示したが, 一般に, cd_a と cd_b には不一致 (conflicts) が存在する. そこで, f, g をそれぞれ変形し, 意味写像 $f' : sch_a \rightarrow cd'_a$ と $g' : sch_b \rightarrow cd'_b$ を作成する. ここで, cd'_a と cd'_b は不一致を解決した後のクラス図である.

[3. 逆写像の作成] $g'^{-1} : cd'_b \rightarrow sch_b$ を求める.
[4. 合成] 写像 $F : sch_a \rightarrow sch_b \equiv g'^{-1} \cdot f'$ (図 5) を求める.

本提案モデルでは, 以上の構築作業を行うために必要な, 写像の操作系を定義する. これについては次章で説明する.

4. 写像操作系

提案する写像操作系は, 写像を操作するための 7 つの操作子から構成される (図 8).

4.1 逆写像

意味写像 $f : sch \rightarrow cd$ が与えられたとき, 逆写像を $f^{-1} : cd \rightarrow sch$ とする. $f : sch \rightarrow cd$ が単純な意味写像であるときは, 必ず逆写像が存在する.

4.2 ラベル変更

ラベル変更操作子は, 意味写像 f において各ノードに付与されているラベルを変更する. 次のように定義する. 意味写像 $f : sch \rightarrow cd$ が与えられたとする. また, n_i を cd 中のあるノードとし, e を +I, -I, +C, -C のいずれかであるとする. このとき, f のラベル変更 $\tau_{n_i}^e[f] : sch \rightarrow cd$ は, 本質的に f と同じ写像であるが n_i のラベル (l_{n_i}) を指定 e に従って変更したものである. 例えば, $\tau_{c1}^{+I}[f]$ は f の l_{c1} に I を追加したものである.

4.3 意味領域射影

意味領域射影は, f のラベルを用いて値域の値 (タグ付きレコード) に対して射影を行う. 次のように定義する. $f : sch \rightarrow cd$ が与えられ, f の l_{n_i} のうち C を含むものを $l_{i1} \dots l_{im}$ とする. f の意味領域射影 $\pi_C[f] : sch \rightarrow cd_C$ は, $\pi_C[f] \equiv [n_{i1} : (e_{i1}, l_{i1}), \dots, n_{im} : (e_{im}, l_{im})]$ である. $\pi_I[f]$ も同様に定義する.

4.4 意味合成

意味写像 f, g が与えられ, $\pi_C[f] : sch \rightarrow cd_C, \pi_C[g] : sch' \rightarrow cd'_C$, かつ, cd'_C が cd_C の部分グラフであるとする. このとき,

名前	操作子
逆写像	f^{-1}
ラベル変更	$\tau_{n_i}^e[f]$
意味領域射影	$\pi_e[f]$
意味合成	$f^{-1} \circ g$
意味領域拡張	$\gamma_e^n[f]$
値域制限	$\omega_e^e[f]$
名前変更	$\delta_{n'}^n[f]$

図 8 写像の操作子
Fig. 8 Operators of Mappings

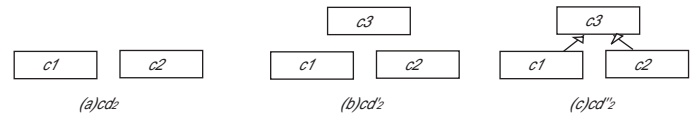


図 9 γ 操作 (C 操作) と ω 操作 (G 操作) の適用
Fig. 9 Applying γ (C) and ω (G) Operators

g^{-1} と f の意味合成を $g^{-1} \circ f : sch \rightarrow sch'$ と表記する.

4.5 意味領域拡張

意味領域拡張は, 意味写像 $f : sch \rightarrow cd$ が与えられたとき, cd に新たな構成要素 n (クラス, 関連, もしくは属性) を追加した cd' への写像 $f' : sch \rightarrow cd'$ を作成する操作である. 例えば, $f_{A0} : Sch_A \rightarrow cd_0$ (Sch_A は図 2(a), cd_0 は図 3 の cd_1 から student クラスを除去したクラス図) が与えられたとき, student クラスを追加した写像 $f_A : Sch_A \rightarrow cd_1$ を作成するといった操作である. それに伴い, 追加されたノード n のためのリレーション R_n の計算が必要になる. 次のように定義する. 意味写像 $f : sch \rightarrow cd, f \equiv [n_1 : (e_1, l_1), \dots, n_m : (e_m, l_m)]$ が与えられたとする. n をクラス c , 関連 $r(c_1, \dots, c_m)$, 属性 a のいずれかとし, e を SPJUM 代数式 (Selection, Projection, Join, Union, Map (関数適用演算子) からなる式) とする. このとき, f の意味領域拡張を $\gamma_e^n[f] : sch \rightarrow cd', \gamma_e^n[f] \equiv [n_1 : (e_1, l_1), \dots, n_m : (e_m, l_m), n : (xsm(e), \phi)]$ と定義する. ここで, $xsm(e)$ は SPJUM 代数式 e で表現されたリレーションを求める XSM 式である. 次に例を示す. $f_2 : Sch_2 \rightarrow cd_2$ (ここで cd_2 は図 9(a) で表されるクラス図. Sch_2 は省略), $f_2 \equiv [c_1 : (\text{for } S_i \text{ in } /a \text{ return } S_i \text{ as } x, \{C\}), c_2 : (\text{for } S_j \text{ in } /b \text{ return } S_j \text{ as } y, \{C\})]$ とする. この時, $\gamma_{R_{c_1 \cup R_{c_2}}}^{c_3}(f_2) : Sch_2 \rightarrow cd'_2$ は次のようになる. すなわち, cd'_2 は図 9(b) で表されるクラス図, $\gamma_{R_{c_1 \cup R_{c_2}}}^{c_3}(f_2) \equiv [c_1 : (\dots), c_2 : (\dots), c_3 : ((\text{for } S_i \text{ in } /a \text{ return } S_i \text{ as } x) \text{ union } (\text{for } S_j \text{ in } /b \text{ return } S_j \text{ as } y), \phi)]$ である. R_{n_i} は n_i のリレーションを表している.

どの種類のノードを追加するかを明示するために, 特に理由が無ければ γ の代わりに C, R, A と表記する. 上の式はクラスを追加するので, $C_{R_{c_1 \cup R_{c_2}}}^{c_3}[f]$ と書く. また, 式を簡潔にするため, 演算子のパラメータとして, SPJUM 代数式の代わりに意味を推測しやすい省略記法を用意する. それらのいくつかは 4.8 節で簡単に説明する.

4.6 値域制限

値域制限は, 意味写像を f が与えられたとき, f と同等ではあるが値域の狭い写像を求める. 操作子のパラメータとしては, キーの指定や関連の多重度など, クラス図に対する制約を指定する. 次のように定義する. $f : sch \rightarrow cd$ が与えられたとする. また, クラス図に追加すべき制約を e とする (例えば, c_1 が c_2 のサブクラスであるとき e として $c_1 \sqsubseteq c_2$ と書く). このとき, f の e による値域制限は $\omega_e^e[f] : sch' \rightarrow cd'$ と表される. ただし, $dom(cd') \subset dom(cd)$ であり, $dom(sch') \subseteq dom(sch)$ である ($dom(sch') \subset dom(sch)$ となるかは指定する制約などに依存). 例として図 9(c) に $\omega_{c_2 \sqsubseteq c_3}^{c_3}[\omega_{c_1 \sqsubseteq c_3}^{c_3}[C_{R_{c_1 \cup R_{c_2}}}^{c_3}[f_2]]] : Sch_2 \rightarrow cd''_2$ における cd''_2 を示す. 追加する制約の種類をより明確にするため, ω の代わりに G, M, K など で表記する (図 10).

操作子記述	追加する制約 (c, c_1, c_2 はクラス)
$G^{c_1 \sqsubseteq c_2}[f]$	汎化制約
$M^{c.r;x}[f]$	多重度・参加制約 (r は関連, x は 0,1,?,*)
$P^c[f]$	分割制約
$X^{c:e}[f]$	排他制約 (e は排他式)
$K^{c:a}[f]$	キー制約 (a は属性)

図 10 値域制限 (ω) における意味制約の指定 (一部)
Fig. 10 Semantic Constraints Specified by ω Operator

4.7 名前変更

意味写像 $f: sch \rightarrow cd$ が与えられたとき, $\delta_n^s[f]: sch \rightarrow cd'$ は, f と本質的には同一の写像であるが, 値域として cd 中の構成要素 n の名前 (クラス名など) を s に変更した cd' を持つ.

4.8 操作子の適用例

次の式は, 図 2(a) の Sch_A に対して [1. 意味情報の抽出] を行い $f_A: Sch_A \rightarrow cd_1$ を求める操作である. cd_1 は図 3 で示したものである. $f_{A.simple}$ は Sch_A に関する単純な意味写像である. 次の式は複合操作子を用いているためパラメータの説明を行う. 基本的に C,R,A はそれぞれ新しいクラス, 関連, 属性を作成する. $C_{p,c}^c[f]$ は条件 p を用いてクラス c' からサブクラス c を作成する. $R_a^{r(c_1,c_2)}[f]$ は属性値 a を基に関連 r を作成する. $A_{c',a'}^{c,a}[f]$ はクラス c' の属性 a' をクラス c の属性 a としてコピーする. この操作によって得られた写像の式の一部が 2.5 節の式である.

$$f_A \equiv C_{\text{student}}^{\text{student}} \left[\begin{array}{l} C_{\text{person}}^{\text{prof}} \\ \text{person} \left[\begin{array}{l} C_{\text{person}}^{\text{prof}} \\ \text{person} \left[\begin{array}{l} K_{\text{person.id}}^{\text{id}}[\delta_{\text{pid}}^{\text{id}}] \\ K_{\text{dept.did}}^{\text{did}} \left[\begin{array}{l} M_{\text{person.belongsTo}}^{\text{belongsTo}} \\ * \left[R_{\text{person.belongsTo}}^{\text{belongsTo}}(\text{person}, \text{dept}) \right] \end{array} \right] \\ A_{\text{person.name}}^{\text{name}} \left[\begin{array}{l} A_{\text{person.pid}}^{\text{pid}} \\ A_{\text{dept.dname}}^{\text{dname}} \end{array} \right] \left[f_{A.simple} \right] \end{array} \right] \end{array} \right] \end{array} \right] \end{array} \right]$$

5. 提案モデルに関する性質

本提案モデルは次の性質を持つ.

性質 1. 単純な意味写像 $f_{simple}: sch \rightarrow cd_{sch}$ に本モデルの操作子を適用することにより, 任意の意味写像 $f: sch \rightarrow cd$ を作成できる. すなわち本操作系は [1. 意味の抽出] プロセスを記述できる. これは次の 2 点から分かる. (1) XSM 式の構文 [5] で定義される式が, SPJUM 代数式と同等である. (2) 単純な意味写像 $f_{simple}: sch \rightarrow cd_{sch}$ の XSM 式には, $dom(sch)$ に含まれる XML の部分要素を抽出するために必要な全ての (XPath の) LocationStep が含まれている.

性質 2. 文献 [6] では, 異なるスキーマ間の関係として次を列挙している. (1) 不一致 (Identity の不一致, スキーマの不一致, 意味的不一致, データの不一致), (2) スキーマ間の関係 (集約関係, 汎化, 特化).

あるモデル X が上の不一致を全て解決可能であることを「 X は **Conflict-Resolution Complete** である」ということにする. ただし上記の不一致には形式的定義が無いため, この概念そのものは ill-defined である. 本変換モデルでは, 意味領域拡張操作子を利用することにより, (新しい各ノードに対応するリレーションは SPJUM 代数式を用いて他のリレーションから導出できる範囲であるという条件付きであるが) 先の不一致の解決を一通り記述可能である. これは [2. マッチング] で有効な性質である. 我々は, 本モデルが解決できる範囲 (これは形式的に決まる) の解決が可能であることを **Conflict-Resolution Complete in the scope of SPJUM Algebra** と定義する.

性質 3. 単純な意味写像 $f_{simple}: sch \rightarrow cd_{sch}$ には, 逆写像 $f^{-1}: cd_{sch} \rightarrow sch$ が必ず存在する. これは次の性質 4 と共に [3. 逆写像の作成] で有効な性質である.

単純な意味写像の逆写像は, XQuery 風の問合せで記述できる. 具体的には, cd_{sch} の木構造にしたがって, cd_{sch} 中の各ノード n_i に対応するリレーション R_{n_i} を用いた結合演算を行い, XML 要素群を出力する問合せである.

性質 4. 意味写像 $f: sch \rightarrow cd$ が, 次に説明するクラス **Invertible** に属する時, 写像 f から単純な写像 $f_{simple}: sch \rightarrow cd_{sch}$ を

$f_{simple} = o_1[\dots o_n[f]]$ として求めるための操作子列 $o_1 \dots o_n$ が決まる. パラメータは必ずしも決まらないが, これにより, cd_{sch} への写像を求める操作を, 各操作子のパラメータを求めるという部分問題群に分割することができる.

クラス **Invertible**. $f: sch \rightarrow cd$ がクラス **Invertible** に属する (もしくは invertible である) とは, cd_{sch} に含まれる cd には存在しないノード n_i 全てに関して, リレーション R_{n_i} の計算を, cd に含まれるノード n_j のリレーション R_{n_j} を用いた SPJUM 式として記述できることである.

性質 5. invertible な意味写像 f と意味写像 g が与えられたとき, 意味合成 $f^{-1} \circ g$ の結果は XQuery 問合せで閉じている. これは [4. 合成] で有効な性質である.

これは次の事から分かる. まず f が単純な意味写像の時, 意味合成 $f^{-1} \circ g$ の結果は XQuery 問合せになる. 次に, f が invertible であるとき, $f^{-1} \circ g = (f_{simple})^{-1} \circ o_1[\dots o_n[g]]$ であるから, 上記と同じ問題に帰着する.

6. おわりに

本稿では, データ変換プログラムの構築プロセスを記述し, 議論するための言語としてのデータ変換モデルを提案した. 本モデルは, データ変換プログラム構築に関する諸問題を明確にし, 理解を深め, アドホックではないアプローチを開発するための道具として利用できる可能性がある. 今後の課題としては, 本モデルの性質に関してより詳細な検討を行うことや, 現実のデータ変換に対して本モデルを適用し, 本モデルの適切性を検証することなどがあげられる. 我々は本モデルに基づくデータ変換プログラム構築支援システムも開発中である [3].

[謝辞]

本研究の一部は日本学術振興会科学研究費補助金若手研究 (B) (課題番号 15700108) による.

[文献]

- [1] Jay Banerjee, Won Kim, Hyoun-Joo Kim, Henry F. Korth: Semantics and Implementation of Schema Evolution in Object-Oriented Databases. SIGMOD Conference 1987: 311-322
- [2] S. B. Davidson, A. Kosky: WOL: A Language for Database Transformations and Constraints. ICDE 1997: 55-65
- [3] 古川夏子, 上村匡稔, 大河原俊明, 森嶋厚行, 杉本重雄「XML データからの意味情報抽出支援プロトタイプシステムの実装」日本データベース学会 Letters, Vol. 3, No. 1, 2004 年.
- [4] A. Morishima, H. Kitagawa, A. Matsumoto: A Machine Learning Approach to Rapid Development of XML Mapping Queries. Proc. ICDE 2004, March 2004.
- [5] 森嶋厚行「情報統合応用のためのデータ変換モデルの提案」第 15 回データ工学ワークショップ (DEWS2004), 2004 年 3 月.
- [6] E. Pitoura, O. A. Bukhres, A. K. Elmagarmid: Object Orientation in Multidatabase Systems. ACM Comput. Surv. 27(2): 141-195 (1995)
- [7] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernandez, R. Fagin: Translating Web Data. VLDB 2002: 598-609

森嶋 厚行 Atsuyuki Morishima

筑波大学 知的コミュニティ基盤研究センター 知の共有基盤研究部門 / 図書館情報メディア研究科 助教授. 1993 年筑波大学第三学群情報学類卒業. 1998 年同大学院工学研究科修了. 博士 (工学). 1998 年 ~ 2001 年日本学術振興会特別研究員. 1999 ~ 2000 年 AT&T Labs-Research 客員研究員. 2001 ~ 2003 年芝浦工業大学工学部情報工学科講師. 情報統合技術, XML/WWW データベースなどに興味を持つ. ACM, IEEE-CS, 電子情報通信学会, 情報処理学会各会員. 日本データベース学会正会員.