DEWS2004 5-C-05

情報統合応用のためのデータ変換モデルの提案

森嶋　厚行†

† 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・図書館情報学系
E-mail: †amorishima@acm.org

あらまし　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，データ変換の実装コストを下げる問題に着目する．現在まで，データ変換プログラム構築の様々な側面に焦点を当てた研究が数多く行われてきたが，この問題を議論するための「共通言語」が存在しなかった．我々は，データ統合技術の発展のためにはデータ変換プログラム構築のための共通言語が有効であると考えた．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．
キーワード　データ変換，XML，モデル, 問合せ開発

Proposal of a Data Transformation Model for Information Integration Applications

Atsuyuki MORISHIMA†

† Research Center for Knowledge Communities, Institute of Library and Information Science
University of Tsukuba
E-mail: †amorishima@acm.org

Abstract　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 cost of data transformation implementations. There have been many researches on various aspects of data transformations, but there has been no "common language" for the discussions on the problem. We believe such a common language would facilitate the development of theories and tools in this area. This paper proposes a data transformation model having several nice properties, as a possible formal basis for the development of data transformation programs.
Key words　Data Transformation, XML, Model, Query Development

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．Standish グループの調査 [14] によると，88%のデータ統合プロジェクトが，予定のコスト・時間をオーバーするか，もしくは失敗している．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 サービス系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．

現在までも，データ変換プログラム構築の様々な側面に焦点を当てた研究 [3] [8] [12] が数多く行われてきた．しかし，それぞれ異なるフォーマリズムが利用されており，データベースにおけるリレーショナルデータモデルのような「共通言語」が存在しなかった．我々は，データ変換実装の理論，技術，道具などの発展のためには，データ変換プログラム構築を議論するための共通言語が有効であると考えた．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) が知られているボキャブラリとして，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．

データ変換プログラムの構築は自明な問題ではなく複数の問題が絡んだ複雑な問題である．しかし我々は，これが取り組むに値する問題であり，このような共通言語の開発がこの問題に関する技術を発展させるための基本的な道具として貢献できる可能性があると考えている．

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. データ統合に関する形式的体系に関しては，Bernstein らによる研究 [9] [10] がある．メタデータ (スキーマやスキーマ間の対応関係など) の各種操作を定義しており，スキーマの更新や統合，ビュー定義の再利用等への応用を想定している．また，文献 [7] ではデータ統合に関する理論的考察を行っている．そこでは，統合スキーマとローカルスキーマ間の関係を表現するアプローチである local-as-view と global-as-view に関する比較などが，主に仮想統合ビューに対する問合せ処理の観点から行われている．これらに対し，本研究の新規性は，異なるスキーマ間のデータ変換を行う具体的なプログラムの構築の支援を目的としている点にある．したがって，これらの研究とは焦点が異なっており，補完関係にあると考えられる．

これまでも，様々なデータ変換記述言語 [4] や，Schema Evolution [1] の枠組みが提案されてきたが，これらに対する本研究の新規性は次の通りである．(1) 他の言語が主に「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．言い換えると，現在はデータ変換の構築が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ボキャブラリを用意し,「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．

図 1　データ変換の写像によるモデル化

```
univ=(dept*,person*)
dept=(@did:ID, dname)
person=(pid, @type,
        name, @did:IDREF)
```
(a)

```
univ=(dept*)
dept=(did, dname, people)
people=(prof*,student*)
prof=(pid,name)
student=(sid, name)
```
(b)

図 2　2 種類の XML スキーマ Sch_A(a) と Sch_B(b)

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．
- 簡単化のため，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．

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

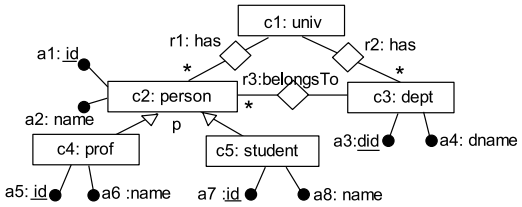

図 3　クラス図 cd_1

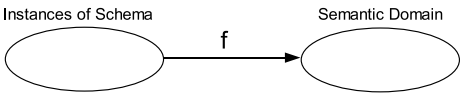

図 4　意味写像

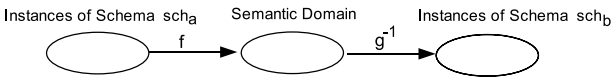

図 5　意味領域を介したデータ変換

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 $f : sch \to cd$ 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(注1)，属性をノードとするグラフである．クラス図の形式的な定義は付録 1. に示す．

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

$$[n_1 : R_{n_1}, n_2 : R_{n_2}, \ldots, n_m : R_{n_m}]$$

である．ただし，$n_1 \ldots n_m$ 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 $c3$ ノード (dept クラス) に対応する R_{c3} の例を図 6(a) に示す．

関連 r の場合: $R_{c_1} \ldots R_{c_n}$ 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 $R_r(attr_1, \ldots, attr_n)$ である．図 6(b) に R_{r3} の例を示す．

(注1)：汎化は関連ではない．

(a) R_{c3}

dept
#did1
#did2

(b) R_{r3}

person	dept
#pid1	#did1
#pid2	#did1
#pid3	#did1
#pid4	#did2

(c) R_{a8}

student	name
#pid1	”furukawa”
#pid2	”nakamizo”
#pid4	”ariyama”

図 6　意味領域の値 $v \in dom(cd_1)$ の断片

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_{a8} の例を示す．

関連 r の属性 a の場合: $R_r(attr_1, \ldots, attr_m)$ を関連 r のリレーションとする．このとき, R_{n_i} は a の値を持つ属性を追加した $(m+1)$ 項リレーション $R_a(attr_1, \ldots, attr_m, attr_a)$ である．

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 $f : sch \to cd$ は次の形式を持つ．

$$[n_1 : (e_{n_1}, l_{n_1}), n_2 : (e_{n_2}, l_{n_2}) \ldots, n_m : (e_{n_m}, l_{n_m})]$$

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，$f_A : Sch_A \to cd_1$ (ただし Sch_A は図 2(a)，cd_1 は図 3 参照) において，R_{c3}(図 6(a))，R_{r3}(図 6(b))，R_{a8}(図 6(c)) を計算する式 e_{c3}, e_{r3}, e_{a8} 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．また，XSM 式の構文は付録 3. に示す．

e_{c3}: for $\$k_0$ in /univ/dept/@did return $\$k_0$ as dept;

e_{r3}: for $\$x_1$ in /univ/person, $\$k_1$ in $\$x_1$/pid, $\$k_2$ in univ/dept/@did where $\$x_1$ /@did = $\$k_2$ return $\$k_1$ as person, $\$k_2$ as dept;

e_{a8}: for $\$x_3$ in /univ/person, $\$v_1$ in $\$x_3$/name, $\$k_3$ in $\$x_3$/pid where $\$x_3$/@type =”student” return $\$k_3$ as student, $\$v_1$ as name;

次に，写像定義における l_{n_i} は，n_i が写像 $f : sch \to cd$ 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，$f_B : Sch_B \to cd_1$ では prof，student が Sch_B に現れる (図 2(b)) ため $l_{c4} = \{C, I\}$, $l_{c5} = \{C, I\}$ であるが，$f_A : Sch_A \to cd_1$ では $l_{c4} = \{C\}$, $l_{c5} = \{C\}$ である．ここでは説明を省略するが，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．

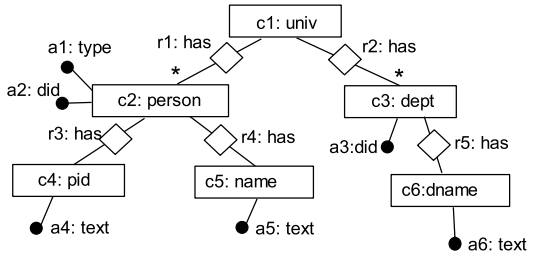

図 7　クラス図 cd_{Sch_A}

名前	操作子
逆写像	f^{-1}
ラベル変更	$\tau^e_{n_i}[f]$
意味領域射影	$\pi_e[f]$
意味合成	$f^{-1} \circ g$
意味領域拡張	$\gamma^n_e[f]$
値域制限	$\omega^e[f]$
名前変更	$\delta^{n'}_n[f]$

図 8　写像の操作子

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} 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ようなものである．単純な意味写像の形式的な定義は付録 2. に示す.

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 からも 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 $\tau^e_{n_i}[f] : sch \rightarrow cd$ 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，$\tau^{+I}_{c1}[f]$ 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 $l_{i1} \ldots l_{im}$ とする．f の意味領域射影 $\pi_C[f] : sch \rightarrow cd_C$ は，$\pi_C[f] \equiv [n_{i1} : (e_{i1}, l_{i1}), \ldots, n_{im} : (e_{im}, l_{im})]$ 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，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 $f : sch \rightarrow cd$, $f \equiv [n_1 : (e_1, l_1), \ldots, n_m : (e_m, l_m)]$ が与えられたとする．n をクラス c，関連 $r(c_1, \ldots, c_m)$，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 [2]) からなる式) とする．このとき，f の意味領域拡張を $\gamma^n_e[f] : sch \rightarrow cd'$, $\gamma^n_e[f] \equiv [n_1 : (e_1, l_1), \ldots, n_m : (e_m, l_m), n : (xsm(e), \phi)]$ 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)，$f_2 \equiv [c_1 : (\text{for \$i in /a return \$i as x}, \{C\}), c2 : (\text{for \$j in /b return \$j as y}, \{C\})]$ とする．この時，$\gamma^{c3}_{R_{c_1} \cup R_{c_2}}(f_2) :$

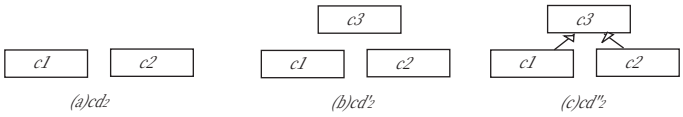

図 9　γ 操作 (C 操作) と ω 操作 (G 操作) の適用

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)

$Sch_2 \rightarrow cd'_2$ 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，$\gamma^{c3}_{R_{c_1} \cup R_{c_2}}(f_2) \equiv [c_1 : (\ldots), c2 : (\ldots), c3 : ($(for $i in /a return $i as x) union (for $j in /b return $j 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，$C^{c3}_{R_{c_1} \cup R_{c_2}}[f]$ 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 $\omega^e[f] : sch' \rightarrow cd'$ と表される．ただし，$dom(cd') \subset dom(cd)$ であり，$dom(sch') \subseteqq dom(sch)$ 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 $\omega^{c_2 \sqsubseteq c_3}[\omega^{c_1 \sqsubseteq c_3}[C^{c3}_{R_{c_1} \cup R_{c_2}}[f_2]]] : Sch_2 \rightarrow cd''_2$ 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)．

4.7 名前変更

意味写像 $f : sch \rightarrow cd$ が与えられたとき，$\delta^s_n[f] : sch \rightarrow cd'$ 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．$C^c_{\sqsubseteq_p c'}[f]$ 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．$R^{r(c_1,c_2)}_a[f]$ は属性値 a を元に関連 r を作成する．$A^{c.a}_{c'}[f]$ はクラス c' の値を持つ属性 a を c に追加する．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．

$$f_A \equiv C^{student}_{\sqsubseteq_{@type='s'}person}[C^{prof}_{\sqsubseteq_{@type='p'}person}[K^{person.id}[\delta^{id}_{pid}[K^{dept.did}[R^{belongsTo(person,dept)}_{did}[A^{person.name}_{name}[A^{person.pid}_{pid}[A^{dept.dname}_{dname}[f_{A.simple}]]]]]]]]$$

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．

5.1 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) 付録 3. の構文で定義される XSM 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．

5.2 性　質　2

文献 [11] 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．

5.3 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} を用いた入れ子ループ演算を行い，XML 要素群を出力する問合せである．例として，sch が `a=(b,c)` の場合の，$f_{simple} : sch \rightarrow cd_{sch}$ の逆写像 $f^{-1} : cd_{sch} \rightarrow sch$ を図 11 に示す．

5.4 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 $f_{simple} = o_1[\ldots o_n[f]]$ として求めるための操作子列 $o_1 \ldots o_n$ 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．

for $\$x_1$ in R_a
return <a>
{
for $\$x_2$ in R_b
where $(\$x_1, \$x_2)$ is contained in $R_{r(a,b)}$
return ...
}
{
for $\$x_3$ in R_c
where $(\$x_1, \$x_3)$ is contained in $R_{r(a,c)}$
return <c>... </c>
}
<a>

図 11　単純な意味写像の逆写像

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} に含まれるが 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．

言い換えると，f が invertible である時，$f_{simple} : sch \rightarrow cd_{sch}$ は次のように計算できる．

$$f_{simple} : sch \rightarrow cd_{sch} = o_1[\ldots o_n[f]]$$

ただし o_i が γ_e の時，e は SPJUM 式 $h_i(R_{n_1}, \ldots, R_{n_m})$ である．ポイントは，f が Invertible であるとき，$f^{-1} \circ g = (f_{simple})^{-1} \circ o_1[\ldots o_n[g]]$ と記述できることである．

次に **Invertible** のサブクラスである **Lossless** (図 12) について説明する．

クラス Lossless. $f : sch \rightarrow cd$ がクラス **Lossless** に属するとは，次の条件を満たすことである．

- f は，$f_{simple} : sch \rightarrow cd_{sch}$ を構成する XSM 式を全て含んでいる．

直観的には，$f : sch \rightarrow cd$ が *lossless* であるとは，$f_{simple} : sch \rightarrow cd_{sch}$ と比較したときに，cd と cd_{sch} 中のノードの名前と制約指定が異なる以外は，f が f_{simple} が持つ情報を全て含んでいる，ということである．(f は余分なクラスなどを含んでいる可能性もある．) したがって，f が lossless の時，$f_{simple} : sch \rightarrow cd_{sch}$ は次のように計算できる．

$$f_{simple} : sch \rightarrow cd_{sch} = o_1[\ldots o_n[f]]$$

ただし，o_i は γ 以外の単項操作子である．

意味関数のクラス Invertible, Lossless, Simple には図 12 の関係がある．これらのうち，Simple, Lossless クラスに属する写像の逆写像は容易に求められるため，最も興味深いクラスは Invertible である．invertible な意味写像 f から f_{simple} を求めるための各操作子 o_i のパラメータを，どこまで具体的に，システム側で決定可能かが問題となる．この問題への徹底的取り組みは今後の課題である．

5.5 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．

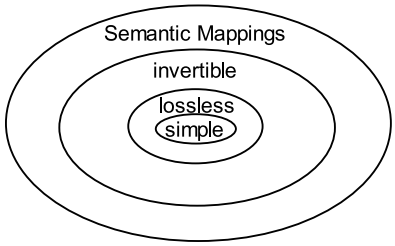

図 12　意味写像のクラス間の関係

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，$f^{-1} \circ g = (f_{simple})^{-1} \circ o_1[\ldots o_n[g]]$ であるから，上記と同じ問題に帰着する．

6. 議論 – このモデルを通じて分かること

本モデルを用いてデータ変換の問題を眺めると，データ変換写像を求めるために解決しなければならない各種部分問題が明確になる．本章では，それらの問題について議論を行う．

意味情報の抽出．意味情報の抽出はそれほど自明ではない．たとえば，必ずしもクラスはスキーマにエンコーディングされているとは限らない (図 2(a) の prof と student など)．一般に，意味領域にどのような要素が出現するかは，スキーマの他に，インスタンスやそのデータベースを利用しているコード中に現れる問合せなど，様々な手がかりから知ることができる可能性がある．また，現実のスキーマは必ずしもデータの制約を漏らさず記述しているとは限らず，スキーマに現れない制約が実際には存在することも多い．例えば，図 2(a) の@type の値は 2 種類しかなく null にはならない等である．したがって，スキーマのみならずインスタンス，各種問合せを入力として，本モデルにおける意味領域拡張，値域制限操作子を利用した意味情報の抽出が重要な問題となる．また，入力としてドメインオントロジが与えられた場合には，それを利用したトップダウン的な意味情報の抽出も一つのアプローチとして考えられる．

意味情報のマッチング．2 つの意味写像 $f : sch_a \rightarrow cd_a$ と $g : sch_b \rightarrow cd_b$ はそれぞれ別に作成されるため，一般に cd_a と cd_b に不一致がある．スキーママッチング [13] で利用されている手法が，ここで応用可能だと考えられる．例えば，名前 (クラス名など) の類似度を利用したマッチングや，インスタンスを利用したマッチングなどである．興味深い点として，本問題においてはスキーママッチングにおける仮定と異なり，cd_a と cd_b が基本的には似た構造であるという前提条件を利用できる可能性がある．例えば，クラス図構造の類似性を利用したマッチングも可能ではないかと考えている．

逆写像の作成．(1) invertible な意味写像 $f : sch \rightarrow cd$ から $f_{simple} : sch \rightarrow cd_{sch}$ を求める式 $f_{simple} = o_1[\ldots o_n[f]]$ において，o_i が γ_e^n の場合に，そのパラメータ $e = h_i(R_{n_1}, \ldots, R_{n_m})$ を具体化する問題がある．少なくとも，具体化のためのいくつかのパターンは存在する．例えば，o_i が sch のルート要素に対

応するクラスを作成するための γ 演算ならば，このクラスはただ一つのインスタンスを生成するため，パラメータとして定数 (シングルトンリレーション) を与えればよい．f が与えられたとき，このようなパターンを発見し，f を f_{simple} に変換するための具体的な式を可能な限り自動的に求める作業が必要である．一般に，f が lossless でなければ，この作業には f 自身が持つ XSM 式以外の手がかり (XML のスキーマ情報など) が必要なはずである．(2) $h_i(R_{n_1}, \ldots, R_{n_m})$ 中の Mapping 演算で利用する関数の具体的な値を求める必要である．例えば，スコーレムファンクションが現れた場合，具体的な値をどうやって求めるのか，等という問題がある．これに対しては，値を元のインスタンスから抽出するなどのアプローチが考えられる．

7. 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 [6].

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, Hyoung-Joo Kim, Henry F. Korth: Semantics and Implementation of Schema Evolution in Object-Oriented Databases. SIGMOD Conference 1987: 311-322
[2] R. Bird and P. Wadler. Introduction to Functional Programming. Prentice Hall International, 1988.
[3] Special Issue on Data Transformations. Data Engineering Bulletin, Vol. 22, No. 1, March 1999.
[4] S. B. Davidson, A. Kosky: WOL: A Language for Database Transformations and Constraints. ICDE 1997: 55-65
[5] R. Elmasri and S. B. Navathe. Fundamentals of Database Systems. Addison Wesley, 2000.
[6] 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システムの開発」第 15 回データ工学ワークショップ (DEWS2004), 2004 年 3 月.
[7] M. Lenzerini. Data Integration: A Theoretical Perspective. PODS 2002: 233-246
[8] A. Morishima, H. Kitagawa, A. Matsumoto. A Machine Learning Approach to Rapid Development of XML Mapping Queries. Proc. ICDE 2004, March 2004.
[9] S. Melnik, E. Rahm, P. A. Bernstein: Rondo: A Programming Platform for Generic Model Management. SIGMOD Conference 2003: 193-204
[10] R. Pottinger, P. A. Bernstein: Merging Models Based on Given Correspondences. VLDB 2003: 826-873
[11] E. Pitoura, O. A. Bukhres, A. K. Elmagarmid: Object Orientation in Multidatabase Systems. ACM Comput. Surv. 27(2): 141-195 (1995)
[12] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernandez, R. Fagin: Translating Web Data. VLDB 2002: 598-609
[13] Erhard Rahm, Philip A. Bernstein: A survey of approaches to automatic schema matching. VLDB J. 10(4): 334-350 (2001)
[14] Standish Group. Chaos 2000.

付　録

1. クラス図の定義

本モデルではクラス図を次のように定義する．

定義 3. クラス図は 6 つ組

$$(N, E, name, exclusive, key, partitioned)$$

である．ここで各要素は次のように定義される．

- $N = C \oplus A \oplus R$ はノードの集合である．ただし $c \in C$ はクラスノード，$a \in A$ は属性ノード，$r \in R$ は関連ノードである．$\oplus$ は disjoint な和集合を表す演算である．
- $E = T \oplus S \oplus P$ は辺の集合である．ただし，(1) $t \in T$ は関連の各終端を表す辺であり，組 $(r, c, role, (min, max))$ である．r は関連ノード，c はクラスノード，$role$ は関連 r における c の役割を表す．(min, max) は，c の各インスタンスが r に参加できる最小と最大の数を表すものであり，2 項関連以外の任意の n 項関連に関しても制約を精確に記述できる [5]．ただし，図 3 では簡単化のため，この表記ではなく $1, 0, *, +$ を用いた関連の多重度表記を用いて表している．また，図 3 では，role はクラス名と同じであるためこれも省略している．(2) 各 $s \in S$ は汎化を表す．s はクラスノードの組 (p, c) であり，c が p のサブクラスであることを表す．(3) $p \in P$ は属性を表す辺であり，組 (n, a) である．ここで n はクラスノードもしくは関連ノード，a は属性ノードである．
- $name : N \to NAME$ は各ノードに名前を付ける関数である．図 3 において，$name(c1) = \text{univ}$ である．
- $exclusive$ は関連間の排他制約を表す関数である．$exclusive : C \oplus R \to Expressions$ で定義される．$Expressions$ の例としては $(a, b|c, d|e)$ などがある．図 3 では，$c1$ に結びつく関連 $r1$ と $r2$ は互いに排他的ではないので，$exclusive(c1) = (r1, r2)$ と表記する．図 3 ではこの記述を省略している．
- $key : A \to \{T, F\}$ で定義される関数である．属性 a がクラスもしくは関連を一意に定めるとき，$key(a) = T$ とする．また a をキー属性と呼ぶ．図 3 ではキー属性に下線を引いている．例えば，$key(a1) = T$ である．
- $partitioned : C \to \{T, F\}$ で定義される関数である．あるクラス c のインスタンス集合を $ext(c)$ とする．このとき，c の全てのサブクラス $c_1, \ldots, c_n$ に対して $ext(c) = ext(c_1) \oplus \ldots \oplus ext(c_n)$ の時，$partitioned(c) = T$．それ以外は F．図 3 においては，$partitioned(c2) = T$ であることが，汎化の脇にかかれた P によって表現されている．

2. 単純な意味写像の定義

定義 4. $f : sch \to cd$ が次の条件を満たすとき，f は単純 ($simple$) であると定義する．

- sch 中で定義されている要素と，cd 中のクラスが一対一

対応する．

- *sch* 中で定義されている要素属性と，*cd* 中のクラス属性が一対一対応する．
- *sch* 中に定義されているテキストノードと，*cd* 中のクラス属性 (text という名前を持つ) が一対一対応する．
- *sch* において要素 a が要素 b の子要素であり，これらに対応する *cd* 中のクラスが n_a と n_b の時，n_a は n_b との間に関連 `has` が存在する．
- *sch* において a が要素 b の属性もしくはテキストノードのとき，*cd* において，n_a は n_b クラス属性である．
- *sch* において要素 a が要素 b の子要素であり，これらに対応する *cd* 中のクラスが n_a と n_b，これらの間の関連を r_{ab} とする．その時，各 XSM 式は次のようになる．
  - $e_{n_b} = \$v_b : \ldots$;
  - $e_{n_a} = \$i_a$ `in` $\$v_b/tag$ `return` $\$i_a$ `as` a;
  - $e_{r_{ab}} = \$i_a$ `in` $\$v_b/tag$ `return` $\$i_b$ `as` a, $\$i_b$ `as` b;
- *sch* において要素 a がテキストノード t を持つとき，対応する各 XSM 式は次のようになる．
  - $e_{n_a} = \$v_a : \ldots$;
  - $e_{n_{a.t}} = \$x_a$ `in` $\$v_a$`/text() return` $\$v_a$ `as` a `,` $\$x_a$ `as` `text`;
- *sch* において要素 a が要素属性 $attr$ を持つとき，対応する各 XSM 式は次のようになる．
  - $e_{n_a} = \$v_a : \ldots$;
  - $e_{n_{a.attr}} = \$x_a$ `in` $\$v_a$`/@attr return` $\$v_a$ `as` a `,` $\$x_a$ `as` $attr$;

3. XSM 式の構文

XSM 式の構文を図 A·1 に示す．XSM 式が Class を追加する γ 演算 (C 演算) で用いられるときには，外部参照用変数 v を先頭に追加することができる．例えば，$\$v$: `for` $\$x$ `in` p 等である．外部参照用変数は，他の式から参照でき，式を簡潔に書くために利用することができる．例えば，`for` $\$y$ `in` p, $\$z$ `in` $\$y/q$ `return` $\$y$ `as` A, $\$z$ `as` B は，先の$\$v$ を利用して，`for` $\$z$ `in` $\$v/q$ `return` $\$v$ `as` A, $\$z$ `as` B と書くことができる．

```
<xsme>::=  [ <var> : ] (<xsmef>) {  union (<xsmef>) }
           // <var>: は class 用 XSM 式のみ
<xsmef> ::=  <expr> { , <expr> } [ <whereExpr> ] <returnExpr>
<expr>::= <var>
        | <var>  in [ <var> ] <SimpleLocationPath>
        | <var>  in <function> ( <var> { , <var> } )
<whereExpr>::= where <predicates>
<returnExpr>::= return  <var> as <id> { , <var> as <id> }
```

図 A·1　XSM 式の構文