XML データからの意味情報抽出支援システムの開発

古川　夏子[†]　　上村　匡稔[††]　　大河原俊明[†††]　　森嶋　厚行[††††]　　杉本　重雄[††††]

† 芝浦工業大学大学院 工学研究科　〒 337–8570 埼玉県さいたま市見沼区深作 307
†† 筑波大学大学院 図書館情報メディア研究科　〒 305–8550 茨城県つくば市春日 1–2
††† 図書館情報大学 図書館情報学部　〒 305–8550 茨城県つくば市春日 1-2
†††† 筑波大学 知的コミュニティ基盤研究センター　〒 305–8550 茨城県つくば市春日 1-2
E-mail: †m102186@sic.shibaura-it.ac.jp, ††{s156,mori,sugimoto}@slis.tsukuba.ac.jp, †††k115@ulis.ac.jp

あらまし　XML はインターネットにおけるデータ交換のための標準フォーマットとしての地位を確立している．XML データの量が劇的に増加するにしたがい，XML データに対するデータ統合が重要な問題となっている．データ統合における重要な事項として，データ変換プログラムの開発がある．データ変換プログラムの開発には，大きなコストがかかる事が知られている．我々は，XML のデータ変換プログラムの開発を支援するために，XML データから意味情報 (クラス図) を抽出するためのシステムの開発を行っている．具体的には，本システムは既存の XML スキーマとインスタンスを入力として受け取り，それらの意味情報としてクラス図を出力する．本システムの特徴は二つある．第一に，出力として，クラス図だけでなく XML インスタンスとクラス図を結びつけるための問合せも同時に生成するため，データ変換プログラム開発への利用が容易なことである．第二に，形式的に定義されたモデルに基づき開発しており，アドホックでない枠組みを提供することである．本稿では，本システムが利用するモデルの概要，意味情報の抽出手法，システム設計，プロトタイプの実装について説明する．

キーワード　XML，情報統合，データ変換，意味情報，リバースエンジニアリング

Development of a System to Extract Semantic Information from XML Data

Natsuko FURUKAWA[†], Tadatoshi KAMIMURA[††], Toshiaki OHKAWARA[†††], Atsuyuki MORISHIMA[††††], and Shigeo SUGIMOTO[††††]

† Grad. Sch. of Eng., Shibaura Inst. of Tech.,　307 Fukasaku, Saitama-City, Saitama 330-8570 Japan
†† Grad. Sch. of Lib, Info. and Media Studies., Univ. of Tsukuba,　1–2 Kasuga, Tsukuba, 305–8550, Japan
††† Dept. of Lib. and Info., Univ. of Lib. and Info.,　1–2 Kasuga, Tsukuba, 305–8550, Japan
†††† RCKC., Univ. of Tsukuba,　1–2 Kasuga, Tsukuba, 305–8550, Japan
E-mail: †m102186@sic.shibaura-it.ac.jp, ††{s156,mori,sugimoto}@slis.tsukuba.ac.jp, †††k115@ulis.ac.jp

Abstract　XML has become a standard format for data interchange on the Internet. As the amount of XML data grows, integration of XML data becomes one of the crucial problems. While the development of data transformation programs is a key issue to achieve data integration, it is well-known that the development requires tremendous efforts in general. We are developping a system for extracting semantic information (class diagrams) from XML data, which is aimed at helping the program development process. The system takes as input a pair of an XML schema and its instance, and outputs a class diagram to represent its semantic information. This system has the following two features: First, the system outputs not only class diagrams but also queries to relate the given XML instances with the generated class diagrams. This facilitates the development of the data transformation programs. Second, the system is designed based on a formal model of data transformations so that it can provide a systematic framework to support the development. The paper explains an overview of the model of data transformations, a method of extracting semantic information, and the design implementation of our prototype system.

Key words　XML, Information Integration, Data Transformation, Semantic Information, Reverse Engneering

1. はじめに

XML はインターネットにおけるデータ交換のためのデファクトスタンダードとしての地位を既に確立している．現実に作成・管理される XML データの量が劇的に増加するにしたがい，XML データに対するデータ統合が重要な問題となっている．データ統合の実現は，スキーマの分析や変換プログラムの実装など，大きな開発コストがかかる．我々は，データ統合の重要な事項であるデータ変換の問題に着目する．データ変換問題の最もわかりやすいデータ変換の例としては，ローカルなデータベースのデータを，統合スキーマに合わせるための変換がある．我々は，意味情報を利用した，データ変換プログラムの開発支援の研究を行っている [2] [3]．我々のアプローチでは，変換対象となる XML データから意味情報を抽出し，それをデータ変換に利用する．本稿における意味情報とは，直観的には XML データが表す概念モデルに関する情報である．例えば，図 1(b) の XML スキーマ Sch_B が表す意味情報は，構造は異なるものの本質的には図 2 のクラス図として表現できる．本稿では，XML データから意味情報を抽出するためのシステムの設計・実装について説明する．

データの意味情報は，本プロジェクトだけでなく様々なデータ統合・変換プロジェクトで利用されており [1] [4] [6]，一般的にも XML から意味情報を抽出する事は重要な問題である．

本システムの新規性は主に二つある．第一に，クラス図を抽出するだけでなく，XML インスタンスとクラス図を結びつけるための問合せも同時に生成する事である．これにより，概念モデルの各構成要素がどのように XML として表現されているかがわかるため，データ変換への利用が容易である．我々の知る限り，データ（RDB，XML など）からクラス図などを抽出するための他のシステム/手法 [7] [8] [9] では，XML データとの関連は生成しない．第二に，形式的に定義されたモデルに基づき開発されている事である．これにより，次の利点がある．(1) 意味情報の抽出作業が，形式的に定義されたプリミティブな操作の組合せで行われるため，一貫性のあるアドホックでない枠組みを提供できる．(2) システムのモジュール化，機能拡張が容易である．

2. 写像としてのデータ変換と意味情報の抽出

我々の提案するデータ変換モデルでは，データ変換を写像としてモデル化する (図 3). このとき，データ変換の問題は次の様に記述する事ができる．

Given: 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．

Output: $I(sch_a)$ にデータ変換写像 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．

データ変換問題の具体例を次に示す．

例 1. 図 1(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．これらは構造が異なるものの，実はどちらも図 2 のクラス図 cd_0 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．

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

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

図 1　2 種類の XML スキーマ Sch_A(a) と Sch_B(b)

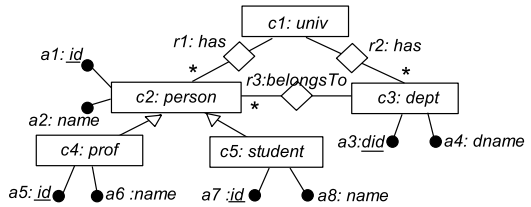


図 2　クラス図 cd_0

図 3　データ変換の写像によるモデル化

2.1 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 2) が規定する値の集合 $dom(cd)$ である．値域については次節で説明する．

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 [5]．(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できる．(2) スキーマ間の対応関係では自明でない関係を発見できる．

2.2 意味写像の値域

意味写像 $f : sch \rightarrow cd$ の値域は，クラス図 cd によって規定される $dom(cd)$ である．本稿ではクラス図の定義および値の定義は省略し，直観的な説明だけを行う．直観的には，$v \in dom(cd)$ である値 v は，cd 中の各ノード (クラス，関連(注1)，属性) のインスタンス集合を持つタグ付きレコードである．例えば，$v \in dom(cd_0)$ (cd_0 は図 2 参照) は，次の形式をしている．

$[c_1$: univ 要素, c_2 : person 要素の集合, c_3 : dept 要素の集合, $\ldots, r_1 : \ldots, r3$: person と dept の関係表, $\ldots, a_8 : \ldots]$

(注1)：汎化は関連ではない．

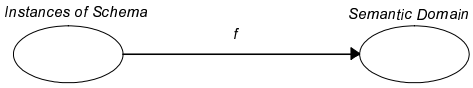


図 4　意 味 写 像

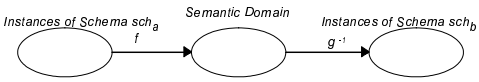


図 5　意味領域を介したデータ変換

2.3　意 味 写 像

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．具体的には，f は，v 中の各ノードに対するインスタンス集合をそれぞれ計算するため，複数の XQuery 風の式から構成される．例えば，意味写像 $f_A : Sch_A \rightarrow cd_0$ (Sch_A は図 1(a)，cd_0 は図 2 参照) において，cd_0 の dept クラス (c_3) のインスタンス集合を計算する式は，for \$$k$ in /univ/depts/dept return \$$k$ as dept である．

2.4　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 を入力としてデータ変換写像 $F : sch_a \rightarrow sch_b$ を構築する問題は，一般に次の 4 つの部分問題に分割する事ができる．

[1. 意味情報の抽出] sch_a を基にクラス図 cd_a を導出し，意味写像 $f : sch_a \rightarrow cd_a$ を構築する．例えば，例 1 においては，Sch_A を基に cd_0 を導出し，$f_A : Sch_A \rightarrow cd_0$ 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_0 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．また $g : sch_b \rightarrow cd_b$ も同様に構築する．同じく例 1 では，Sch_B からも cd_0 が導出され，$f_B : Sch_B \rightarrow cd_0$ が構築される．

[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 ($cd_a = cd_b = cd_0$) 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] データ変換写像 $F : sch_a \rightarrow sch_b \equiv g'^{-1} \cdot f'$ を求める（図 5）．

本稿で説明するシステムは，上の **[1. 意味情報の抽出]** の支援を行う．意味情報の抽出のために，本システムでは我々が提案するデータ変換モデルで定義している写像の操作系を利用する．次節ではこの概要を説明する．

2.5　写像操作系

本データ変換モデルの写像操作系は，写像を操作するための 7 つの操作子 (逆写像 f^{-1}，意味合成 $f^{-1} \circ g$，ラベル変更，意味領域射影，意味領域拡張，値域制限，名前変更) を持つ．本稿では，特に **[1. 意味情報の抽出]** で重要な操作子である，意味領域拡張と名前変更について説明する．他の演算子の説明は [5] にある．

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 に新たなノード n_i (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 (図 6(a)〜(d))．具体的には次のように指定する．意味写像 $f : sch \rightarrow cd$ が与えられたとする．n_i をクラス c，関連 $r(c_1, \ldots, c_m)$，属性 a のいずれかとし，e を導出の方法を指定するパラメータとする．このとき，意味領域拡張の結果 f' は操作子 γ を用いて $\gamma_e^{n_i}[f] : sch \rightarrow cd'$ で求められる．本稿では，式の可読性を高めるため，γ の代わりに C, R, A と表記し，追加するノードの種類を明示する．以下では，本稿に現れるいくつかの演算子についてのみ説明する．$C^{c_2}_{\sqsubseteq_p c_1}[f]$ は，選択条件 p を用いてクラス c_1 からサブクラス c_2 を導出する (図 6(a))．例えば，$f_{A0} : Sch_A \rightarrow cd_2$ (Sch_A は図 1(a)，cd_2 は図 9(a)) が与えられたとき，$C^{student}_{\sqsubseteq_{@type='s'} person}[f_{A0}]$ は，student クラスを追加した写像 $f_A : Sch_A \rightarrow cd_{0'}$（$cd_0$ から prof を除いたもの）を作成する．$R_a^{r(c_1,c_2)}[f]$ は属性値 a の値が等しい事を表す関連 r を作成する (図 6(b))．$A^{c_2.a_2}_{c_1.a_1}[f]$ はクラス c_1 の属性 a_1 を c_2 の属性 a_2 としてコピーする (図 6(c))．$R^{r_3(c_1,c_3)}_{r_1(c_1,c_2),r_2(c_2,c_3)}$ は関連の合成を作成する (図 6(d))．

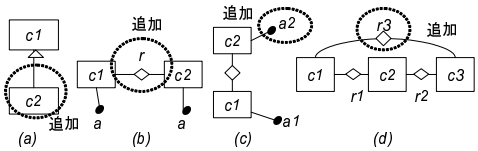


図 6　意味領域拡張操作子による cd から cd' への変更

名前変更.$f : sch \rightarrow cd$ が与えられたとき，名前変更 $\delta_n^s[f] : sch \rightarrow cd'$ は，f と本質的に同一であるが，値域として，cd 中のノード n の名前 (クラス名や属性名など) を s に変更した cd' を持つ．例えば，$f_{A0} : Sch_A \rightarrow cd_2$ (Sch_A は図 1(a)，cd_2 は図 9(a)) が与えられたとき，$\delta^{id}_{pid}[f_{A0}]$ は，cd_2 の pid 属性を id 属性として名前を変更した cd'_2 を値域として持つ写像 $f_A : Sch_A \rightarrow cd'_2$ を作成する．

3.　意味情報抽出支援システム

本システムは，**[1. 意味情報の抽出]** を支援するシステムである．すなわち，XML スキーマ sch からクラス図 cd を導出し，意味写像 $f : sch \rightarrow cd$ を構築する事を支援する．

3.1　アーキテクチャ

本システムのアーキテクチャを図 7 に示す．基本的な考え方は，意味情報の抽出のうち自動化できる部分はシステムが行い，それ以外は人が実行できるような環境を提供することである．意味情報の抽出は次の 3 段階で行われる．

(フェイズ 1) 入力として XML スキーマ sch を受け取り，**単純な意味写像** f_{simple} を出力する (図 7(a))．ここで，単純な意味写像 $f_{simple} : sch \rightarrow cd$ の cd は sch と同型の構造を持つ．すなわち，sch 中の各 XML 要素に対しては cd 中の一つのクラスが対応し，各要素属性に対しては cd 中の一つのクラスの属性が対応する．また，sch 中の要素の親子関係が cd の関連に直接対応する．例えば，図 8(a) は，単純な意味写像 $f_{A.simple} : Sch_A \rightarrow cd_1$($Sch_A$ は図 1(a) に示す) における cd_1 である．また，図 8(b) は意味写

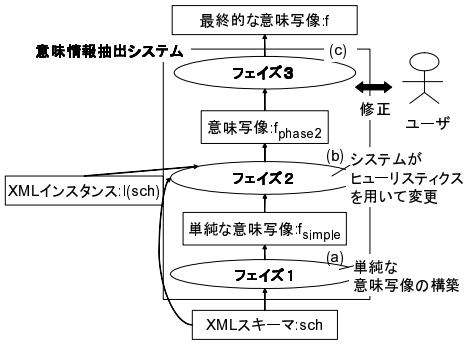
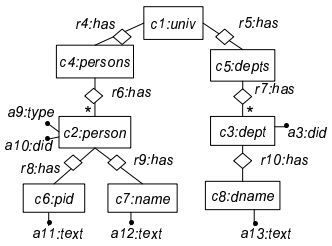


図 7　アーキテクチャ



(a) クラス図 cd_1

c_1 : $\$x_1$ in /univ return $\$x_1$ as univ

c_4 : $\$x_2$ in /univ/persons return $\$x_2$ as persons

r_4 : $\$x_3$ in /univ, $\$x_4$ in $\$x_3$/persons return $\$x_3$ as univ, $\$x_4$ as persons

a_{10} : $\$x_5$ in /univ/persons/person, $\$v_1$ in $\$x_5$/@did return $\$x_5$ as person, $\$v_1$ as did

(b)

図 8　フェイズ 1 の出力例 $f_{A.simple}$ を構成する式の一部

像 $f_{A.simple}$ を構成する式の一部である．

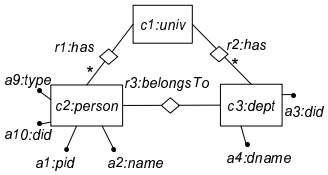
(フェイズ 2) 入力としてフェイズ 1 の結果の f_{simple} と，XML インスタンス $I(sch)$ および XML スキーマ sch を受け取り，ヒューリスティクスを用いて写像操作子を f_{simple} に適用し，新たな意味写像 f_{phase2} を構築し，出力する (図 7(b))．例えば，図 9(a) は，フェイズ 2 が先の $f_{A.simple}$ を入力として構築する意味写像 $f_{A.phase2} : Sch_A \rightarrow cd_2$ における cd_2 である．また，図 9(b) は $f_{A.phase2}$ を構成する式の一部である．

(フェイズ 3) フェイズ 2 で作成される意味情報は必ずしも十分であるとは限らないので，ユーザが写像操作子を f_{phase2} に適用し (図 7(c))，最終的な意味写像 $f : sch \rightarrow cd$ を得る．例えば，図 2 は，フェイズ 3 で $f_{A.phase2}$ を入力として構築した結果の意味写像 $f_A : Sch_A \rightarrow cd_0$ における cd_0 である．

3.2　フェイズ 2 におけるヒューリスティクスの利用

フェイズ 2 は，フェイズ 1 で構築された単純な意味写像 f_{simple} に対しヒューリスティクスを用いて写像操作子を適用し，f_{phase2} を求める．ヒューリスティクスには様々なものが考えられるが，以下では，例 1 の意味情報抽出に必要なものを説明する．

ヒューリスティクス 1. XML において，関連はしばしば ID 属性と IDERF 属性を用いて実装される．したがって，逆に ID 属性と IDREF 属性から関連を作成すれば良い．この目的のために，同じ属性名 a の ID 属性と IDREF 属性を持つクラス c_1, c_2 の間に関連を追加する．これは，操作子 $R_a^{r(c_1,c_2)}[f]$ を用いて行う (図 6(b))．例えば，図 8(a) のクラス図 cd_1 において，クラス person の属性 did は IDREF 属性であり，クラス dept の属性 did は ID 属性である．また，これらの属性は，同じ属性名であるため，関連を追加する．このとき，関連を追加する操作子の式は，$R_{did}^{belongsTo(person,dept)}[f_{A.simple}]$ となる．上記の操作子の式を適用すると，クラス図は図 10 のように変更される．



(a) クラス図 cd_2

c_1 : $\$x_1$ in /univ return $\$x_1$ as univ

c_2 : $\$x_2$ in /univ/persons/person return $\$x_2$ as person

r_1 : $\$x_3$ in /univ, $\$x_4$ in $\$x_3$/persons/person return $\$x_3$ as univ, $\$x_4$ as person

a_1 : $\$x_5$ in /univ/persons/person, $\$v_2$ in $\$x_5$/pid return $\$x_5$ as person, $\$v_2$ as pid

(b)

図 9　フェイズ 2 の出力例 $f_{A.phese2}$ を構成する式の一部

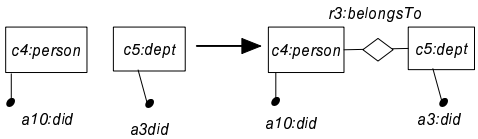


図 10　クラス図 cd_1 に対するヒューリスティクス 1 の適用前と適用後

ヒューリスティクス 2. クラスの属性に対応する情報は，XML においては，しばしば要素属性ではなく独立した要素として表現される．したがって，ある XML 要素がテキストだけを持ち，子要素や属性を一切持たなければ，その要素に対応するクラスの内容を他のクラス属性としてコピーする．これは，操作子 $A_{c_1.a_1}^{c_2.a_2}[f]$ を用いて行う (図 6(c))．例えば，図 1(a) の XML スキーマ Sch_A において，要素 pid はテキストノードのみを持ち，且つ子要素や属性は持っていない．この要素 pid に対応するクラス pid（図 8(a) のクラス図 cd_1）をその親クラス person の属性としてコピーする．このとき，操作子の式は，$A_{pid.text}^{person.pid}[f_{A.simple}]$ となる．上記の操作子の式を適用すると，クラス図は図 11 のように変更される．

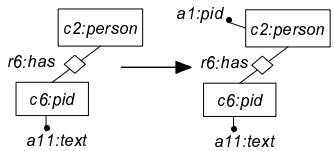


図 11　クラス図 cd_1 に対するヒューリスティクス 2 の適用前と適用後

ヒューリスティクス 3. XML のスキーマを設計するとき，しばしば，それ自体は特に意味を持たないが，単に他の要素をまとめたり形式を整えるための目的で XML の要素を導入する事がある．そのような要素に対応するクラスの特徴として，他のクラスと 1 対 1 関連でつながっていることと，もう一方の関連と 1 対多関連でつながっていて，属性を持たない事がある．したがって，そのようなクラスを次のように除去する．クラス c_1, c_2 の間に関連名 has を持つ関連 r_1 があり，c_2, c_3 の間には関連 r_2 があるとする (図 12)．この時，r_1 か r_2 の一方の関連が 1 対 1 関連且つもう一方の関連が 1 対多関連であり，c_2 が属性を持たないとき，操作子 $R^{r_3(c_1,c_3)}_{r_1(c_1,c_2),r_2(c_2,c_3)}[f]$ を用いて c_1 と c_3 を直接結ぶ関連 r_3 を追加する (図 6(d))．

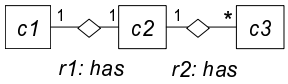


図 12　意味のない XML 要素を発見するためのヒューリスティクス

例えば，図 8(a) のクラス図 cd_1 においては，クラス univ とクラス persons の間に関連名 has を持つ関連 r_1 があり，クラス persons とクラス person の間には関連名 has を持つ関連 r_3 がある．ここで，関連 r_1 は 1 対 1 関連であり，r_2 は，1 対多関連である．また，クラス persons は属性を持たないので，クラス univ とクラス person を直接結ぶ新しい関連 r_8 を追加する．このとき，操作子の式は，$R^{has(univ,person)}_{has(univ,persons),has(persons,person)}[f_{A.simple}]$ となる．上記の操作子の式を適用すると，クラス図は図 13 のように変更される．

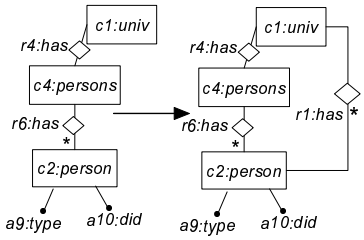


図 13　クラス図 cd_1 に対するヒューリスティクス 3 の適用前と適用後

例 1 のフェイズ 1 で得られた $f_{A.simple}$ に対し，以上のヒューリスティクスを用いて操作子を適用すると，次のような意味写像 $f_{A.phase2} : Sch_A \rightarrow cd_2$ が得られる．

$$
\begin{aligned}
f_{A.phase2} \equiv \quad & R^{has(univ,person)}_{has(univ,persons),has(persons,person)}[\\
& R^{has(univ,dept)}_{has(univ,depts),has(depts,dept)}[\\
& R^{belongsTo(person,dept)}_{did}[A^{person.name}_{name.text}[\\
& A^{person.pid}_{pid.text}[A^{dept.dname}_{dname.text}[f_{A.simple}]]]]]]
\end{aligned}
$$

3.3　フェイズ 3 での操作

フェイズ 2 の例（図 9(a)）と最終的に欲しいクラス図 cd_0（図 2）を比べると，例えば prof クラス等は発見できていない．フェイズ 3 では，フェイズ 2 で得られた f_{phase2} に対してユーザが操作子を利用し，最終的な f を求める．以下の式では，例 1 に対するフェイズ 3 の結果 $f_A : Sch_A \rightarrow cd_0$ を得るための式を示す．ここでのポイントの一つは，フェイズ 2 とフェイズ 3 で共に，本モデルで形式的に定義された操作子を用いているため，システムによる操作と人手による操作が同じ言葉で記述出来る事である．

$$
\begin{aligned}
f_A \equiv \quad & M^{person.belongsTo:*}[C^{student}_{\sqsubseteq_{@type='s'person}}[C^{prof}_{\sqsubseteq_{@type='p'person}}[\\
& \delta^{id}_{pid}[f_{A.phase2}]]]]
\end{aligned}
$$

上記の式において，まず person クラスの pid 属性を id 属性に名前変更するために，δ^{id}_{pid} を適用している．次に，person クラスのサブクラス prof クラスや student クラスを追加するために，$C^{student}_{\sqsubseteq_{@type='s'person}}$ と $C^{prof}_{\sqsubseteq_{@type='p'person}}$ を適用している．最後に，関連 belongsTo の多重度を 1 対多に変更するために，$M^{person.belongsTo:*}$ を適用している．

4.　意味情報抽出支援システムの実装

4.1　システム設計のポイント

本システムの設計にあたっては，次に挙げる 3 つの点に留意した．

システムの拡張性: 機能ごとにモジュールを分割し，モジュール間でやりとりするデータの構造の標準化を図ることにより，システムの拡張性を高くする．これにより，モジュールの交換・新モジュールの追加などが容易になる．例えば，4.7 節で説明するプロトタイプシステムではクラス図の表示に Graphviz [11] を用いているが，クラス図の表示に別のツールを利用したい場合に，表示部分がモジュール化されているため，モジュールの交換が行いやすい．

他ツールとの連携: 標準に準拠することにより，他ツールとの連携を容易にする．例えば，クラス図は XMI 形式 [10] で出力できるようにする．XMI 形式は UML において標準となっている形式である．このため，本システムで出力した XMI 形式のファイルを各種 UML ツールで利用することや，逆に UML ツールで出力した XMI 形式のファイルを本システムで利用することが可能になる．

ユーザによる作業の効率化の支援: ユーザによる作業の効率化を支援するため，操作子の省略記法や，各種複合オペレータを用意することによる入力作業の軽減，および UML による簡潔なクラス図の表示の実現などの取り組みを行う．通常，ユーザがクラス図を操作するためにシステムに入力する式は複雑な記述となるため，ユーザによる入力作業が負担となる．これに対し，ユーザが入力するための省略記法をあらかじめ組み込んだことで，入力作業の軽減を実現する．また，本システムが利用している変換モデルでは，クラス図の表記に ER 図をベースにした形式を用いている．しかし，ER 図は関連や属性をすべてノードで表現するため，複雑になる傾向がある．そこで，クラス図の表示は UML によって行うことにより，簡潔なクラス図の表示を実現する．

4.2　概　　要

本システムの設計を図 14 に示す．本システムは 4 つのコンポーネントから構成される．**(a) 単純な意味写像生成コンポーネ**

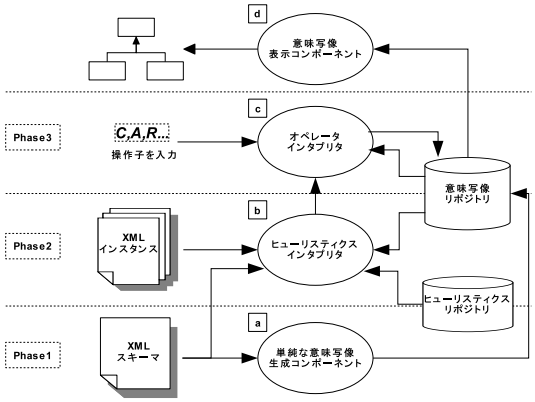


図 14　意味情報抽出支援システムの設計

ント: 3.1 節で説明したフェイズ 1 を行う．すなわち，XML スキーマから単純な意味写像を生成する．**(b) ヒューリスティクスインタプリタ**: フェイズ 2 を行う．具体的には，ヒューリスティクスリポジトリ中のヒューリスティクスを用いて，与えられた意味写像に適用する操作子群を決定する．ヒューリスティクスには 3.2 節で挙げたもの以外にもドメイン固有のヒューリスティクスなど，様々なものが考えられる．したがって，これらヒューリスティクスはハードコーディングせず，リポジトリに保存しそれをインタプリタが実行する設計とする．**(c) オペレータインタプリタ**: 意味写像と意味写像の操作子を受け取り，適用結果を出力する．**(d) 意味写像表示コンポーネント**: 意味写像のクラス図等をユーザに提示する．クラス図の表現形式としては，ER 図，UML のクラス図，OWL 等多様な表現で出力可能とする．以下では，各コンポーネントの各モジュールの説明を行う．

4.3　単純な意味写像生成コンポーネント

単純な意味写像生成コンポーネント (図 14(a)) は，XML スキーマを受け取り，XML スキーマを解析して，単純な意味写像を生成し，意味写像リポジトリに出力する．このコンポーネントは，大きく分けて，XML スキーマ解析モジュール，単純な写像生成モジュールの二つで構成される．

- XML **スキーマ解析モジュール**は，入力として XML スキーマを受け取り，単純な写像を生成するために，XML スキーマの要素，要素属性，要素間の親子関連などを解析し，クラス図のクラス，クラス属性，関連に対応させた構文木に変換する．
- **単純な写像生成モジュール**は，XML スキーマ解析モジュールから受け取った構文木から，クラス図のクラスや属性，関連と対応させて単純な意味写像を生成する．

4.4　ヒューリスティクスインタプリタ

ヒューリスティクスインタプリタ（図 15）は，XML インスタンスと XML スキーマ，単純な意味写像を受け取り，適用すべきヒューリスティクスを発見する．そして，このヒューリスティクスに基づき具体的に適用する操作子を決定する．これらの作業は，ヒューリスティクス決定モジュールおよびオペレータ決定モジュールによって行われる．

4.5　オペレータインタプリタ

オペレータインタプリタ（図 16）は，意味写像とそれに適用

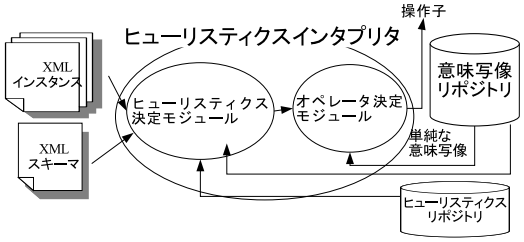


図 15　ヒューリスティクスインタプリタの各モジュール

するための操作子を受け取る．次に，その操作子を適用して意味写像を変換し，それを意味写像リポジトリに出力する．ここで，入力として受け取る操作子は，省略記法や複合操作子などで記述された様々な形式が考えられるので，それをプリミティブな操作子群に展開する処理が必要がある．このコンポーネントは，大きく分けて，操作子解析モジュール，複合操作子展開モジュール，通常表記モジュール，オペレータ適用モジュールの四つで構成される．

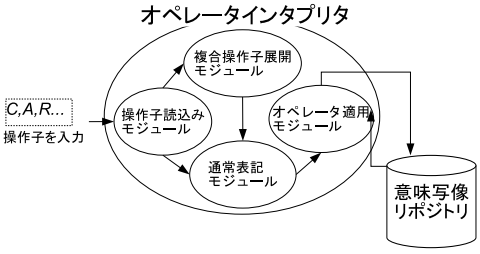


図 16　オペレータインタプリタの各モジュール

- **操作子読込みモジュール**では，ヒューリスティクスインタプリタや直接ユーザから入力された操作子を，複合操作子かどうか，解析する．複合操作子なら，プリミティブな操作子に展開する必要があるので，複合操作子展開モジュールに渡す．そうでない場合は，省略記法で記述されている操作子を通常の表記に変換する必要があるため，通常表記モジュールに操作子を渡す．
- **複合操作子展開モジュール**では，操作子読込みモジュールから入力として受け取った操作子が複合操作子であるときは，その複合操作子を構成しているプリミティブな操作子群に展開する．展開された各操作子は，まだ省略記法で記述されている可能性があるので，さらに通常表記モジュールに渡される．
- **通常表記モジュール**では，操作子読込みモジュールから入力として受け取る省略記法で記述された操作子を，通常表記に変換して，操作子を実行するためのオペレータ適用モジュールに出力する．
- **オペレータ適用モジュール**では，通常表記モジュールから入力として受け取った操作子を，意味写像リポジトリから受け取った意味写像に適用するモジュールである．操作子によって変換された意味写像は，意味写像リポジトリに出力される．

4.6　意味写像表示コンポーネント

意味写像表示コンポーネント（図 17）は，各フェイズ毎に構築された意味写像のクラス図をユーザに表示する．意味写像リポジトリから渡された意味写像は，この目的のために，出力形

式に変換される必要がある．このコンポーネントは，大きく分けて，出力形式変換モジュールとクラス図表示モジュールの二つで構成される．

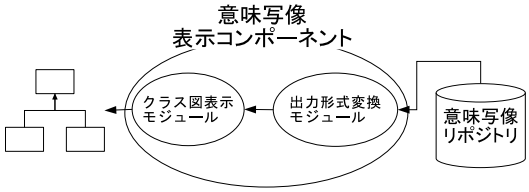


図 17　意味写像表示コンポーネントの各モジュール

- **出力形式変換モジュール**は，クラス図を表示するために，意味写像リポジトリから受け取った意味写像をクラス図を表示するための出力形式に変換する．出力形式は，クラス図を表示するためのツールによって異なる．クラス図を表示するための形式としては，ER 図や UML クラス図など様々である．
- **クラス図表示モジュール**は，入力として，出力形式変換モジュールから出力形式に変換された意味写像を受け取り，それをクラス図として表示し，出力するモジュールである．

4.7　プロトタイプシステム

前節までの設計に基づき，意味情報抽出支援システムのプロトタイプを作成した．記述言語は Java であり，約 3800 行のコードで構成される．Java のコードにおけるクラスの数は 19 である．入力する XML スキーマは RELAX NG で記述する．クラス図としては，現時点では UML のクラス図 (XMI 形式) を出力可能である．フェイズ 3 のユーザインターフェースとしては，現在，操作子を直接記述するコマンドラインインターフェースだけをサポートしている．

図 18 は，本プロトタイプシステムの実行画面を示している．上の部分がコマンドラインインタフェース，左下の部分は意味写像を構成する式の表示であり，右下はクラス図の表示である．

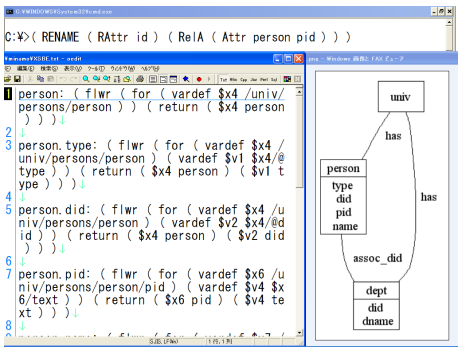


図 18　プロトタイプシステムの実行例

5.　お わ り に

本稿では，データ変換の応用を目的とした XML データからの意味情報抽出支援システムのアーキテクチャ，意味情報の抽出手法，およびプロトタイプシステムの実装についての説明を行った．今後の課題としては，フェイズ 2 で適用するヒューリスティクスの拡張機構の設計や，ドメインオントロジを利用した意味情報の抽出手法の開発などが挙げられる．

謝　　辞

ゼミなどでご議論いただきました芝浦工業大学工学部情報工学科古宮誠一教授に感謝いたします．同じく，筑波大学図書館情報学系永森光晴講師に感謝いたします．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] S. Bergamaschi, S. Castano, M. Vincini: Semantic Integration of Semistructured and Structured Data Sources. SIGMOD Record 28(1): 54-59 (1999)
[2] 古川夏子，森嶋厚行．意味情報を用いた XQuery 問合せ作成支援システムの開発．情報処理学会第 65 回全国大会講演論文集 (3), 2003 年 3 月．
[3] 古川夏子，森嶋厚行．情報統合を目的とした XML の意味情報の抽出．日本データベース学会 Letters, Vol. 2, No. 2, 2003 年 10 月．
[4] R. S. Mello, C. A. Heuser: A Bottom-Up Approach for Integration of XML Sources. Workshop on Information Integration on the Web 2001: 118-124
[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] P. F. Patel-Schneider, J. Simeon: The Yin/Yang web: XML syntax and RDF semantics. WWW 2002: 443-453
[7] S. Ramanathan and J. Hodges. Extraction of object-oriented structures from existing relational databases. ACM SIGMOD Record, 26(1), March 1997.
[8] William J. Premerlani and Michael R. Blaha. An approach for reverse engineering of relational databases. Communications of the ACM, 37(5):42-49, May 1994.
[9] M. Mani, D. Lee, R. R. Muntz: Semantic Data Modeling Using XML Schemas. ER 2001: 149-163
[10] OMG. XMI. http://www.omg.org/technology/xml/index.htm.
[11] AT&T. Graphviz. http://www.research.att.com/sw/tools/graphviz.