XML 文書の双方向変換機構 ～住所録への適用～*

鳥井 修　木村 哲郎　瀬川 淳一
{osamu.torii, tetsuro2.kimura, junichi.segawa}@toshiba.co.jp
(株) 東芝 研究開発センター コンピュータ・ネットワークラボラトリー
〒 212-8582 神奈川県川崎市幸区小向東芝町 1

XML 文書を変換する代表的な手法の 1 つに XSLT がある．しかし，サーバに格納されている XML 文書をクライアント用 XML 文書に変換し，クライアントで編集した XML 文書をサーバ用 XML 文書に逆変換した上でサーバに反映するシステムでは，XML 文書の双方向変換が必要であるため XSLT は利用できない．本稿では XML 文書の双方向変換機構を提案する．本方式はサーバ用 XML 文書からクライアント用 XML 文書への簡単な変換規則を与えるだけで，クライアント用 XML 文書からサーバ用 XML 文書への逆変換が自動的に行われるという特徴を持つ．本稿ではまた，本方式の住所録への適用例の説明を行う．

A Bidirectional Transforming Technology for XML Documents ～Case Study: Address Book Application～

Osamu TORII　Tetsuro KIMURA　Jun-ichi SEGAWA
{osamu.torii, tetsuro2.kimura, junichi.segawa}@toshiba.co.jp
Computer & Network System Laboratory,
Coporate Research & Development Center, Toshiba Corporation
1, Komukai Toshiba-cho, Saiwai-ku, Kawasaki 212-8582, Japan

XSLT is powerful and widely-used technology of transforming XML document's structure into another one. However, current XSLT does not support bidirectional transformation. In this paper, we propose bidirectional transforming technology for XML documents. The only task necessary for users is to describe the forward transformation using simple rule, and the forward/backward transformation mechanism is established automatically. We also describe a case study for applying this technology to various address book applications.

*本研究は放送・通信機構の委託研究「スーパーインターネットプラットフォーム技術の研究開発」の一部として行った．

1 はじめに

近年の情報機器の普及にともない, 一人のユーザが PC, PDA, 携帯電話など複数の端末機器を保有するケースが増えてきている. これらの端末機器で扱う情報の中には非常に関連の深いデータが多く含まれており, 住所録はその代表例である. このような状況下において, サーバで一元的に情報を管理し, 必要に応じてサーバから個々の端末機器 (クライアント) に読み出して利用するシステムに対する需要が高まってきている.

また近年では, 情報を記録, 保存する場合や, インターネットやイントラネットなどの通信路を介して情報を交換する場合において, XML 文書 ([1]) が標準的な文書形式として利用されるケースが一般的になりつつある.

サーバで一元的に管理されている XML 文書をクライアントで利用するためには, サーバ用 XML 文書をクライアント用 XML 文書へ変換しなければならない. ある XML 文書を変換し, 別の XML 文書を得る代表的な変換方式の一つに XSLT ([2]) がある. クライアントが読み出し専用である場合には XSLT はこの目的に適している. しかし, クライアントで編集した結果をサーバ用 XML 文書に逆変換し, サーバに反映させる目的には適してない. なぜならば, サーバ用 XML 文書とクライアント用 XML 文書では一般的に前者の方が情報量が多いからである.

そこで, サーバ用 XML 文書とクライアント用 XML 文書の双方向変換に適した新しい変換方式が必要である. 新しい変換方式は, 簡単な設定を行うだけで実現可能であり, システムの拡張に柔軟に対応するものが望ましい. 本稿で提案する変換方式は, サーバ用 XML 文書からクライアント用 XML 文書への変換時に得られた情報を用いて, クライアント用 XML 文書からサーバ用 XML 文書への逆変換を自動的に行う.

2 XML 文書の双方向変換機構の概要

本稿で提案する変換方式では, サーバ用 XML 文書からクライアント用 XML 文書への変換はパス[1]の単位で行い, クライアント用 XML 文書からサーバ用 XML 文書への逆変換を簡単に実現するために, 変換はプリミティブなものに限定している. また, 変換実行時にサーバ用 XML 文書のどのパスに対してどんな変換を行ったかを詳細なログに残し, このログを用いて逆変換を行う. この際, パスを識別するためにサーバ用 XML 文書のすべてのエレメントに付与される一意な id 属性を用いる.

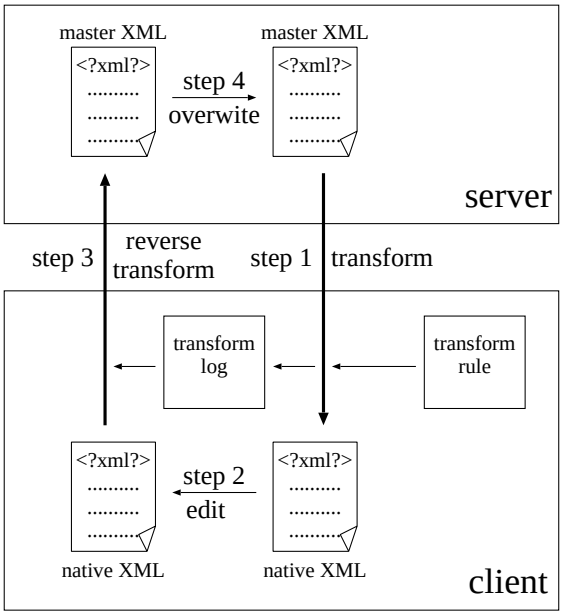

図 1: XML 文書の双方向変換の流れ

図 1 に本方式の流れを示した.

ステップ 1 : サーバ用 XML 文書 (以後マスター XML 文書と呼ぶ) からクライアント用 XML 文書 (以後ネイティブ XML 文書と呼ぶ) への変換を本方式独自の変換規則 (以後トランスフォームルールと呼ぶ) を用いて行う (以後マスター XML 文書からネイティブ XML 文書への変換をトランスフォームと呼ぶ).

ステップ 2 : ネイティブ XML 文書の編集を行う.

ステップ 3 : ネイティブ XML 文書からマスター XML 文書への逆変換をマスター XML 文書をネイティブ XML 文書に変換する際の変換記録 (以後トランスフォームログと呼ぶ) を用いて自動的に行う (以後ネイティブ XML 文書からマスター XML 文書への変換をリバーストランスフォームと呼ぶ).

[1] XML 文書におけるパスに関しては, 文献 [3] を参照のこと.

ステップ 4 : 変換前のマスター XML 文書を逆変換によって得られたマスター XML 文書によって置き換える.

トランスフォームルールは具体的には以下の 2 種類のいずれかである (ただしここで, P, P' はパス名を表すものとする).

パスの再命名 : P に合致するパスを P' に再命名する.

パスの削除 : P に合致するパスを削除する.

トランスフォームルール id	トランスフォームタイプ	変換前パス名	変換後パス名
TFR1	再命名	P1	P2
TRF2	削除	P3	———
...	...	...	...

図 2: トランスフォームルール

また, トランスフォームルールは図 2 に示す形式を取る.

サーバ用の XML 文書にトランスフォームルールを適用しクライアント用の XML 文書に変換する際, パスの変換を実行するごとに, どのパスに対してどのトランスフォームルールを適用したかをトランスフォームログに記録する.

トランスフォームログ id	パス id	パス名	親 id	トランスフォームルール id
TFL1	ID1	P1	ID2	TFR1
...	...	...	...	...

図 3: トランスフォームログ

トランスフォームログは具体的には図 3 に示す形式を取る. トランスフォームログ TFL1 は, エレメント id が ID2 であるエレメントを親としパス名が P1 でパス id が ID1 であるパスに, トランスフォームルール TFR1 を適用したことを意味する.

トランスフォームルールはパスの再命名やパスの削除以外にも, パスの移動, パスの複写, パスの作成, パスの分割, パスの併合などが考えられる. また, 再命名や削除をパスに限定せず, トランスフォームルールとしてツリーの再命名やツリーの削除を指定する方法も考えられる. 本方式では, トランスフォームルールをパスの再命名とパスの削除という単純なルールのみに限定したことにより, 対応する逆変換を簡単に求めることが可能である.

3 住所録への適用

3.1 簡単な具体例

本節において, 住所録の具体例に即してトランスフォーム, リバーストランスフォームが実行される様子を説明する.

図 4 にマスター XML 文書にトランスフォームルールを適用してトランスフォームを行い, ネイティブ XML 文書とトランスフォームログが生成される様子が示してある[2]. この際, 適用しなかったトランスフォームルールも記録する点に注意を要する.

ネイティブ XML 文書をクライアントで編集した結果, 図 5 に示したネイティブ XML 文書が新たに得られたとする. 編集において, 既存のメンバの連絡先を削除するのと同時に, 新規メンバを追加している.

図 6 に, 図 5 に示したネイティブ XML 文書を図 4 に示したトランスフォームログを参照しながらリバーストランスフォームを行い, マスター XML 文書が生成される様子が示してある. トランスフォームログには, パスの再命名とパスの削除の 2 種類が記録されている. パスの再命名に対応するリバーストランスフォームはトランスフォームで行ったパスの再命名の逆の再命名を行い, パスの削除に対応するリバーストランスフォームはトランスフォーム時に削除したパスを追加する. パスの再命名は, 単にパス名を元に戻す場合と, クライアントで新規に追加されたパス名の再命名を行う場合の 2 種類が存在する点に注意を要する. さらに, マスター XML 文書からネイティブ XML 文書への変換では適用されなかったトランスフォームルールに対応するリバーストランスフォームが行われる場合があったり, 逆に, マスター XML 文書からネイティブ XML 文書への変換では適用されたトランスフォームルールに対応するリバーストランスフォームが行われない場合がある点に注意を要する.

[2] XML 文書の終了タグはスペースの関係で, 省略形で表記する.

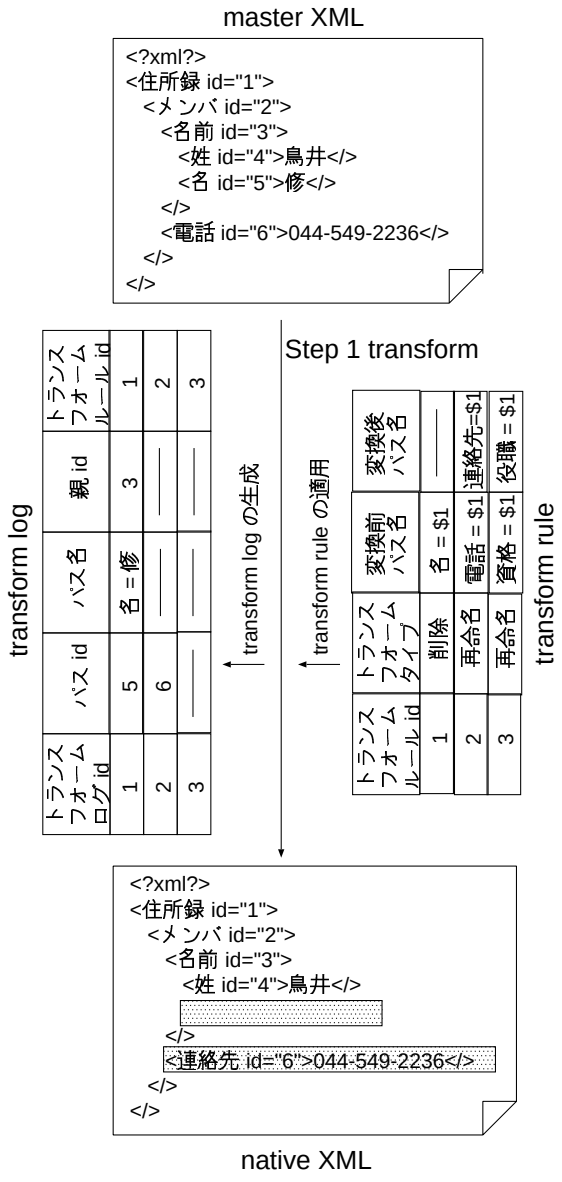

図 4: トランスフォームの簡単な具体例

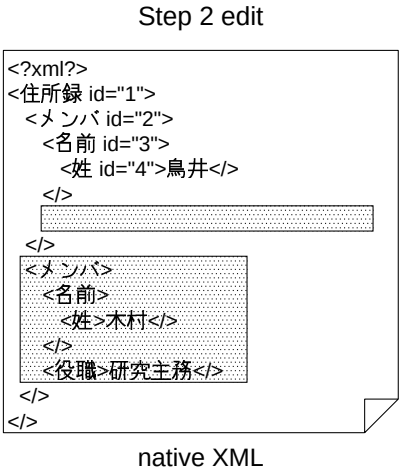

図 5: ネイティブ XML 文書編集の簡単な具体例

3.2 リバーストランスフォームにおける曖昧さの問題

3.1 節で示した簡単な住所録の例ではすべてのトランスフォーム, リバーストランスフォームが明確であったが, 実用的な住所録を扱う場合には, トランスフォーム, リバーストランスフォーム時に変換の曖昧さに関する様々な問題が生じる. 本節ではそれらの問題を列挙し, 解決方法について説明する.

3.2.1 パスの再命名における曖昧さの問題

マスター XML 文書のパスの再命名を実行する際に, トランスフォームの対象となるパスに複数の子供ノードを保持するエレメントが含まれている場合には, トランスフォームに曖昧さが残る. 例えば, 図 4 に図示したマスター XML 文書において, パス『名前 / 姓 = "鳥井"』[3] をパス『姓 = "鳥井"』に再命名するトランスフォームを実行した結果得られるネイティブ XML 文書は, 曖昧さが存在するために何通りか考えられる.

第一の場合を図 7 のネイティブ XML 文書 1 に示した. この場合, パスの再命名によって『名前』エレメントが削除されたので, トランスフォーム前のマスター XML 文書において『名前』エレメントの子供エレメントであった『名』エレメントも同時に削除されるべきであるという立場に立っている. 第

[3] パス表現は XPath [3] を用いている.

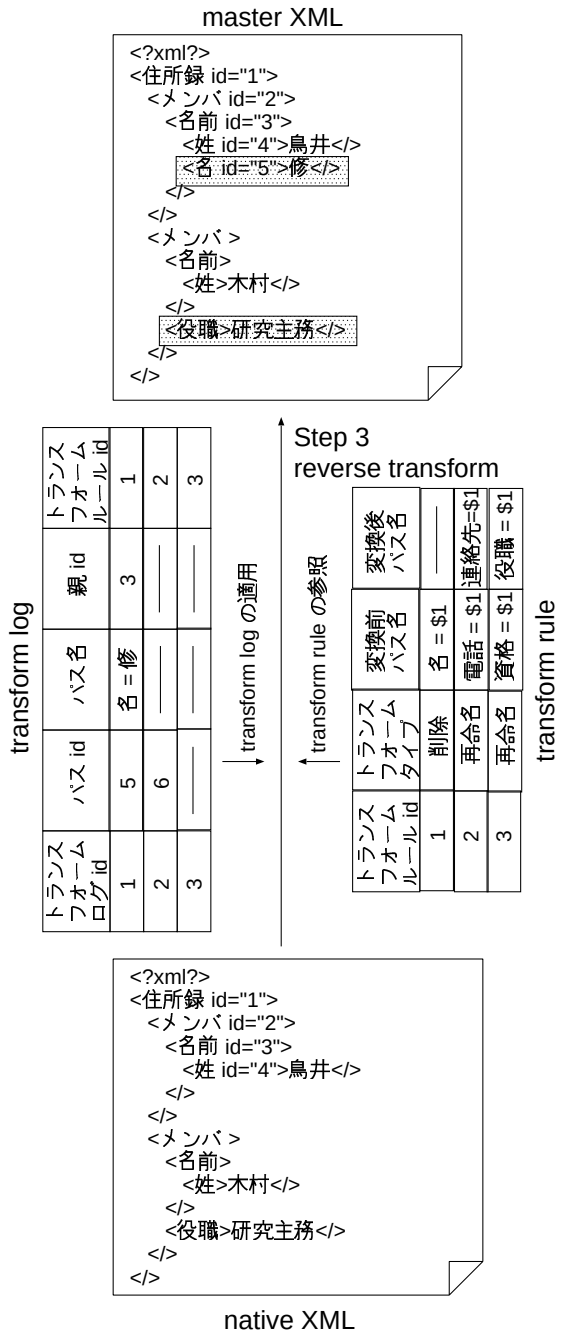

図 6: リバーストランスフォームの簡単な具体例

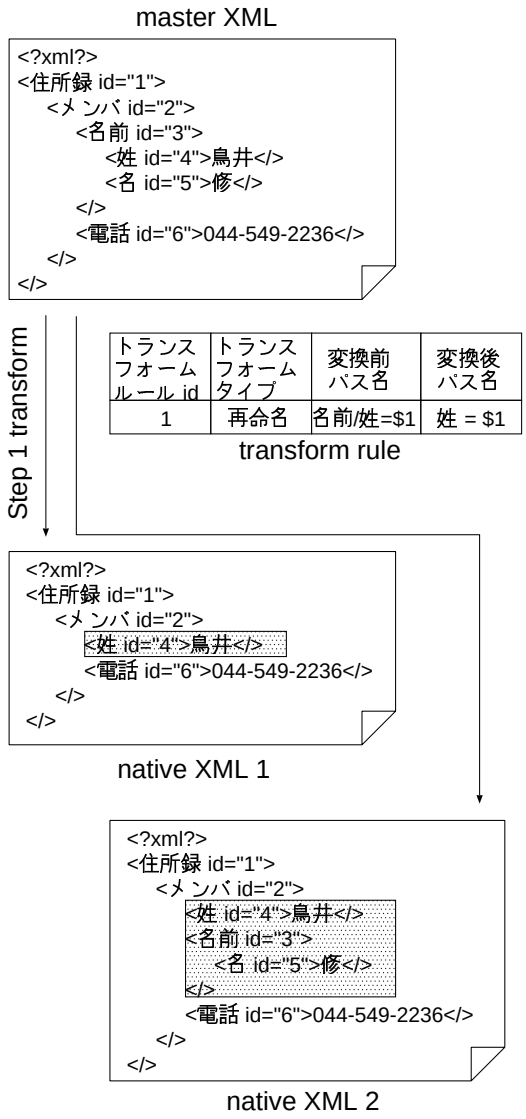

図 7: パスの再命名における曖昧さ

二の場合を図 7 のネイティブ XML 文書 2 に示した. この場合, 再命名命令を, パス『姓 = ”鳥井”』を『名前』エレメントの子供から削除し, パス『姓 = ”鳥井”』を『メンバ』エレメントの子供に追加する, という命令であると解釈し, 『名前』エレメントや『名』エレメントをトランスフォーム後のネイティブ XML 文書に残すべきであるという立場に立っている.

上記曖昧さは, マスター XML 文書中のエレメントが複数の子供を保持するために生じている. そこで本稿で提案する双方向変換方式では, トランスフォームルールを用いたトランスフォームを実行するのに先立ち, 複数の子供を保持するエレメントを子供の数に分割し, マスター XML 文書をエレメントが保持する子供が高々 1 個である XML 文書に変換する前処理を行うことで, トランスフォーム, リバーストランスフォームの曖昧さを排除している. なお, トランスフォームルールを用いたトランスフォームの実行終了後, 分割したエレメントを再度 1 つにまとめる後処理を行う.

ただし, マスター XML 文書のエレメントの中には, 分割することによって元のマスター XML 文書が保持していた意味のまとまりを崩すためにトランスフォームルールを用いたトランスフォームの実行終了後に再度 1 つにまとめることが困難なものが含まれる. 上記エレメントに関しては前処理において例外的に分割を行わない. また, 同様の問題はリバーストランスフォーム実行時にも起こるので, トランスフォームログを用いたリバーストランスフォームの実行の前後にもそれぞれ前処理と後処理を行う.

3.2.2 パスの再命名のリバーストランスフォームにおける曖昧さの問題

2 つの異なるトランスフォームルールを適用してトランスフォームを実行した結果, マスター XML 文書では異なるパス名がネイティブ XML 文書では同一のパス名に変換される場合がある.

図 8 に示すトランスフォームルールはこの例である. マスター XML 文書中のパス『会社住所 = ”川崎市幸区”』にトランスフォームルール 1 を適用してトランスフォームした結果得られるネイティブ XML 文書中のパス『住所 = ”川崎市幸区”』をリバーストランスフォームする場合には, トランスフォームログ中にどの id を保持するパスに対してどのトランスフォームルールを適用したか記録されているために曖昧さの問題は発生しない. しかしながら, クライアントでネイティブ XML 文書に新規に追加したパス『連絡先 = ”川崎市幸区”』をリバーストランスフォームする場合には, このパスをパス『会社住所 = ”川崎市幸区”』に逆変換されるべきであるか, それともパス『自宅住所 = ”川崎市幸区”』に逆変換されるべきであるか, 曖昧さの問題が発生する.

トランスフォームルール id	トランスフォームタイプ	変換前パス名	変換後パス名
1	再命名	会社住所 = $1	住所 = $1
2	再命名	自宅住所 = $1	住所 = $1

transform rule

図 8: パスの再命名のリバーストランスフォームにおける曖昧さ

正確なところはネイティブ XML 文書を編集したユーザに問い合わせなければ分からない. しかしながら, 逆変換の曖昧さが生じる度にユーザに問い合わせるのはユーザにとって煩わしいので, 本方式では自動的に逆変換を決定する. 具体的には, 完全に同一のパスを生成する可能性がある複数の (3 個以上である可能性もある) トランスフォームルールはルールグループと呼ばれるグループにまとめ, このグループに含まれるトランスフォームルールのうち 1 つをデフォルトトランスフォームルールに定める. リバーストランスフォームを実行する際にはデフォルトトランスフォームルールを優先的に用いてリバーストランスフォームを行う.

先のトランスフォームルールの例において, トランスフォームルール 1 とトランスフォームルール 2 を 1 つのルールグループにまとめ, トランスフォームルール 1 をこのルールグループのデフォルトトランスフォームルールに指定した場合, ネイティブ XML 文書中のパス『住所 = ”川崎市幸区”』はパス『会社住所 = ”川崎市幸区”』に逆変換される.

3.2.3 パスの削除における曖昧さの問題

マスター XML 文書とネイティブ XML 文書を比較した場合, 一般的に前者の方が情報量が多いので, マスター XML 文書をネイティブ XML 文書にトラ

ンスフォームする際, 複数のエレメントの候補の中から特定のエレメントのみを残し, それ以外のエレメントを削除する場合がある. マスター XML 文書中の会社の住所, 自宅の住所のうちいずれか一方のみを代表住所としてネイティブ XML 文書に含める場合や, マスター XML 文書中の電子メールアドレス, 電話番号のうちのいずれか一方のみを連絡先としてネイティブ XML 文書に含める場合などがこれに相当する. 住所録中のすべてのメンバに関して, 常に会社の住所を代表住所にする (または, 常に自宅の住所を代表住所にする) 場合や, 常に電子メールアドレスを連絡先にする (または, 常に電話番号を連絡先ににする) 場合にはパスの削除における曖昧さの問題は発生しない. しかしながら, メンバごとに会社の住所, 自宅の住所のうちどちらを代表住所にするか, 電子メールアドレス, 電話番号のうちどちらを連絡先にするか異なっている場合には曖昧さの問題が発生する.

本方式では, ポインタと呼ばれる XML 文書のエレメントに付与される属性を複数のエレメントの候補の中から特定のエレメントを選択する目的に用い, ポインタ属性が特定の値を保持しているエレメントのみをネイティブ XML 文書に含める.

例えば, 図 9 にマスター XML 文書にトランスフォームルールを適用した結果得られるネイティブ XML 文書を示してある. ここで, 『連絡先』属性が先に説明したポインタである.

ポインタを導入したことにより, リバーストランスフォームにも曖昧さの問題が発生するが, この問題に関しては紙面の都合により本稿では説明を省略する.

4 結論

XML 文書の双方向変換を, 本方式独自の簡単な変換規則であるトランスフォームルールを利用するトランスフォームと, トランスフォームログを利用するリバーストランスフォームにより実現した. 本方式は, 変換時に得られる情報を用いて逆方向の変換を自動的に実行可能であるという特色を持つ. 本方式を具体的な住所録データに適用し, その際に生じる様々な問題に対処した. 現在 XML 文書の双方向変換機構は住所録にのみ適用されているが, 今後は対応するアプリケーションの数を増やし, その際

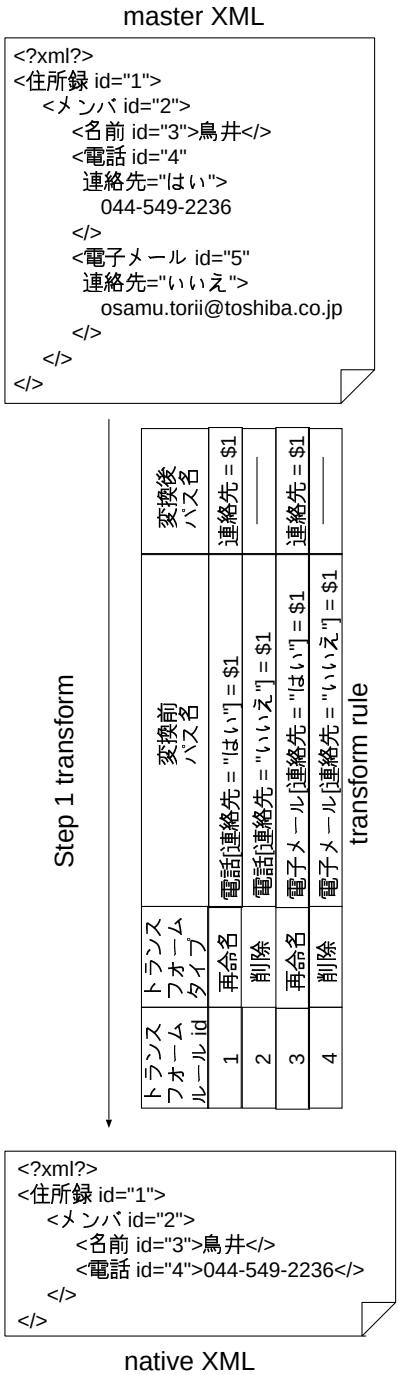

図 9: パスの削除における曖昧さ

に生じる様々な問題に対応する予定である.

参考文献

[1] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, "Extensible Markup Language (XML) 1.0 (Second Edition)", http://www.w3.org/TR/2000/REC-xml-20001006, W3C Recommendation, 6 October 2000

[2] James Clark, "XSL Transformations (XSLT) Version 1.0", http://www.w3.org/TR/1999/REC-xslt-19991116, W3C Recommendation, 16 November 1999.

[3] James Clark, Steve DeRose, "XML Path Language (XPath) Version 1.0", http://www.w3.org/TR/1999/REC-xpath-19991116, W3C Recommendation, 16 November 1999.