XMLデータベースへの関数従属性を用いた推論攻撃に対する安全性の定式化とその検証法の提案

阪野　公秀† 　橋本　健二† 　石原　靖哲† 　藤原　融†

† 大阪大学大学院情報科学研究科　〒 565-0871 大阪府吹田市山田丘 1-5
E-mail: †{k-sakano,k-hasimt,ishihara,fujiwara}@ist.osaka-u.ac.jp

あらまし　推論攻撃とは, ユーザが許可されている問合せとその結果から, 許可されていない問合せの結果 (機密情報) を推論し得ようとする攻撃である. これまで, 我々は攻撃者が利用できる情報として, ユーザに許可されている複数の問合せとその問合せ結果, 機密情報を求める問合せ, そしてデータベースがもつ関数従属性を考え, それらを用いた推論攻撃に対する XML データベースの安全性を定式化している. また, データベースのスキーマにおいて要素が再帰的に宣言されていないという条件下で, 安全性の検証法を提案している. 本稿では, スキーマにおいて要素が再帰的に宣言されている場合について, 一部の状況を除き安全性を検証する手法を提案する.

キーワード　データベースセキュリティ, XML, 推論攻撃, 関数従属性

A Formalization of the Security Against Inference Attacks Using Functional Dependencies on XML Databases and a Proposal of a Security Verification Method

Kimihide SAKANO†, Kenji HASHIMOTO†, Yasunori ISHIHARA†, and Toru FUJIWARA†

† Graduate School of Information Science and Technology, Osaka University,
1-5, Yamadaoka, Suita, Osaka, 565-0871, Japan
E-mail: †{k-sakano,k-hasimt,ishihara,fujiwara}@ist.osaka-u.ac.jp

Abstract　Inference attacks on databases mean that an attacker tries to infer the execution result of a query unauthorized to the attacker (i.e., secret information) from the execution results of authorized queries. We have formulated the security against inference attacks using functional dependencies on XML databases, and have proposed a security verification method under condition that all elements of the schema are not recursively defined. In this paper, we work on the case that some element of the schema may be recursively defined. We propose a security verification method excluding a certain situation.

Key words　Database Security, XML, Inference Attack, Functional Dependency

1. まえがき

現在, 企業や機関などの多くの組織ではデータベースシステムを用いて情報の蓄積・活用が行われている. それらのデータベースにはしばしば, 外部に漏れてしまうことで, 損失に繋がったり, 組織の信用を低下させてしまうような機密性の高い情報が保持されている. その情報の安全性を守ることは組織にとって重要な課題である. データベースでの機密情報の漏洩を防ぐ手法として問合せにおけるアクセス制御がある. この手法では, データベースはユーザに許可されている問合せに対してのみ, その問合せに対する答えを返す. これにより, 機密情報への直接のアクセスを制限し漏洩を防ぐ. しかし, 組織のデータベースシステムにおいて, アクセス権限の付与等のセキュリティ面での管理が正確に行われていないことが問題となっている. さらに, アクセス制御により機密情報への直接のアクセスを禁止していたとしても, ユーザが許可されている問合せとその結果, そしてデータベースに関する知識から, 機密情報（許可されていない問合せの結果）を推論し, 絞り込むことができてしまう場合がある. このような攻撃を推論攻撃と呼ぶ.

これまで我々は, XML データベースを対象とした推論攻撃に対する安全性を定式化し, その検証法を提案している [1]. この定式化では, 推論によって機密情報の値の候補が絞り込まれないことを安全と考えている. 具体的には, 与えられたデータベースのスキーマや, ユーザが許可されている問合せとその実

行結果, 機密情報を得るための問合せに対し, 機密情報の候補値が有限個にならないことを無限安全, $k-1$ 個以下に絞り込まれる可能性がないことを k 安全と定式化している. さらに, 文献 [2] では, より強力な推論を可能にする状況としてデータベースが関数従属性をもつ場合を考え, XML における関数従属性を新たに定義し, その関数従属性を用いた推論攻撃に対する安全性の定義, および検証法を提案している.

ここで, XML データベースにおける関数従属性を用いた推論攻撃の例を次に示す.

[例 1] ある病院に入院している患者の氏名と病名, 病室, そして担当医の対応を表す XML 文書 D を考える. D は以下のスキーマに従っているとする.

病院 → 患者*
患者 → 氏名, 病名, 病室, 担当医
氏名 → string
病名 → string
病室 → string
担当医 → “田中” | “佐藤”

すなわち, 病院要素は子として患者要素を 0 個以上もち, 患者要素は子に氏名要素と病名要素, 病室要素, そして担当医要素をそれぞれ 1 つずつもつ. 氏名, 病名, 病名要素は文字列 ($string$) を値としてとり, 担当医要素は “田中”, または “佐藤” を値としてとる.

また, XML 文書 D は次の関数従属性を満たしているとする.

(/病院/患者, [/患者/病室], [/患者/病名])

この関数従属性は, 「同じ病室の患者は同じ病気を患っている」ということを意味する. つまり, D 中の病院要素の子の全ての患者要素を通して, 患者要素の子の病室要素がとる文字列値が同じならば, 患者要素の子の病名要素がとる文字列値も同じである.

全ての患者の氏名と病室の組を取り出す問合せを T_1 とし, 担当医が “佐藤” である全ての患者の病室と病名を取り出す問合せを T_2 とする. これらの問合せはいずれも, 元の文書における要素の出現順序を保存したまま情報を取り出す. ここで, 機密情報への問合せ T_S を “阪野” という氏名の患者のかかっている病気を取り出す問合せとする.

今, T_1 及び T_2 が許可されており, 問合せの実行結果 $T_1(D)$, $T_2(D)$ がそれぞれ図 1(a), (b) に示すとおりであったとする. このとき, 各患者の病室要素の値の出現順序に着目すると, $T_2(D)$ に現れている病室 102 の患者と, 病室 101 の患者は, それぞれ $T_1(D)$ の廣田という患者と, 橋本という患者に対応することがわかる. このことから, 廣田の病気は心臓病であり, 橋本の病気は癌であることがいえる. また, $T_1(D)$ より, 阪野と橋本が同じ病室 101 に入院していることがわかるので, このことと XML 文書 D が満たす関数従属性から, 阪野と橋本が同じ病気を患っていることが推測できる. 以上のことから, 阪野の病気は橋本と同じ癌であることがわかり, 本来禁止されている T_S の実行結果が 1(c) のようになると結論付けることができる. □

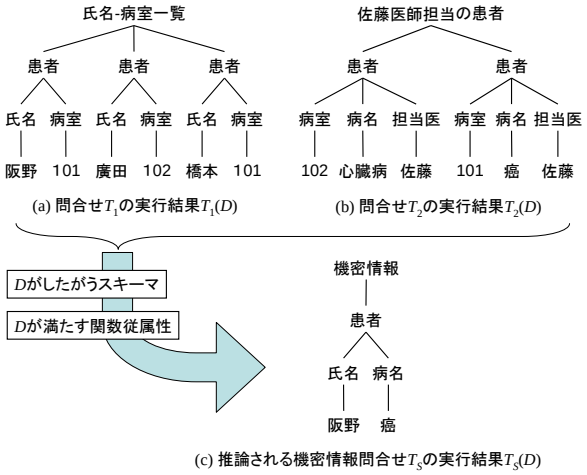

図 1 推論攻撃の例

例で示したような推論攻撃に対して, 文献 [2] の安全性定義は対応しているが, 提案された検証法では「与えられたデータベースのスキーマにおいて要素が再帰的に宣言されていない」という条件下でしか安全性の検証ができなかった. そこで本稿では, スキーマにおいて要素が再帰的に宣言されている場合でも, ある一部の状況を除き実行可能な安全性を検証する手法を提案する.

XML データベースへの推論攻撃に関する研究としては, 文献 [3], [4] などがある. 文献 [3] では, 攻撃者が機密情報を推論するのに有用な情報を含まない security view という概念を提案しているが, 攻撃者の推論の定式化が明確ではない. 文献 [4] では, 攻撃者がスキーマ情報と関数従属性を用いて推論を行うという仮定のもとで, 機密情報の漏洩無く公開できる安全な極大のビューを求めるアルゴリズムを提案している. これらはいずれもユーザに単一のビューのみを与える環境を想定した研究である. 文献 [5] では, 関係データベースにおいてユーザに複数のビューを与えたときの推論攻撃に対する安全性が研究されている. ユーザに複数のビューを与えるほうが, 単一のビューのみを与えるよりも可用性が高い環境である. 本稿では, XML データベースにおいてユーザに複数のビューを与えるという環境を想定している.

また, XML における関数従属性については, 先の文献 [4] の他に, 文献 [6], [7], [8] などでそれぞれの研究目的に適した定義, 形式が与えられている. しかし, 使用する XML に関するモデルの違い等の理由から, それらの定義を本研究に適用することが難しい. そのため, 我々は文献 [2] において, 本研究に適した形式の関数従属性を新たに定義しており, 本稿でもその定義を利用する.

2. 諸 定 義

本節では, 検証で用いる XML データベースに関するモデル, 及び XML データベースにおける関数従属性についての定義を与える. XML データベースに関するモデルについては, 文献 [1]

及び [2] で用いられているモデルを, 本稿の検証法においても使用する.

2.1 XML 文書のモデル

XML 文書のモデルとしてラベル付き順序木を用いる. ラベル付き順序木は, 以下の定義で与えられるものである.

[定義 1] あるアルファベット Σ について, Σ 上のラベル付き順序木の集合 $\mathcal{T}_\Sigma$ を, 以下を満たす最小集合と定義する.

- $a \in \Sigma$ ならば, $a \in \mathcal{T}_\Sigma$.
- $a \in \Sigma$, $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$ であるならば, $a(t_1 \cdots t_n) \in \mathcal{T}_\Sigma$.

ここで $\mathcal{T}_\Sigma^*$ は $\mathcal{T}_\Sigma$ の Kleene 閉包を表す. 以下ではラベル付き順序木を単に木と呼び, しばしば項により表現する. □

2.2 XML スキーマのモデル

XML スキーマのモデルには非決定性有限木オートマトンを用いる. まずは, 非決定性有限木オートマトンの定義で用いる非決定性有限オートマトンを定義する.

[定義 2] N 上の非決定性有限オートマトンは以下の 5 つ組 $e = (P, N, \delta, \hat{p}, P_f)$ である.

- P : 状態の有限集合.
- N : 有限アルファベット.
- δ : 遷移規則の集合.
  ただし, $p, p' \in P$, $a \in N$ とすると, 遷移規則は (p, a, p') の形式である.
- $\hat{p} \in P$: 初期状態.
- $P_f \subseteqq P$: 終了状態の集合.

N 上の非決定性有限オートマトン e が与えられたとき, e によって認識される N 上の言語を $L(e)$ と書く. □

続いて, 非決定性有限木オートマトンの定義を与える.

[定義 3] 非決定性有限木オートマトンは以下の 4 つ組 $A = (Q, \Sigma, q_0, R)$ である:

- Q : 状態の有限集合.
- Σ : 有限アルファベット.
- $q_0 \in Q$: 初期状態.
- R : 遷移規則の集合.
  ただし, $q \in Q$, $a \in \Sigma$ とし, e を Q 上の言語を受理する非決定性有限オートマトンとすると, 遷移規則は (q, a, e) の形式である.

以下, $A = (Q, \Sigma, q_0, R)$ の動作について述べる. 木 $t = a(t_1, \ldots, t_n)$ を考える. ここで, $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$ である. 木 t の根頂点に状態 q が割り当てられたとき $q(t)$ と書く. $q_1 \cdots q_n \in L(e)$ であるような遷移規則 $(q, a, e) \in R$ が存在するならば, A は $q(t)$ から $a(q_1(t_1) \cdots q_n(t_n))$ へ遷移可能であると定義する. 木 $t \in \mathcal{T}_\Sigma$ に対して, $q_0(t)$ から始めて, 以上のような遷移をトップダウンに各部分木について繰り返すことによって, 最終的に t へ到達可能である場合, A は t を受理するという. A が t を受理するということは, 木 t で表現される XML 文書が木オートマトン A で表現されるスキーマに従うことを意味する. 木オートマトン A によって受理される全ての木の集合を $TL(A)$ と書く. □

2.3 XML 文書への問合せのモデル

本稿では, 2 種類の決定性木変換器 (決定性 relabeling 木変換器, 決定性 deleting 木変換器) の合成を XML 文書への問合せモデルとして用いる.

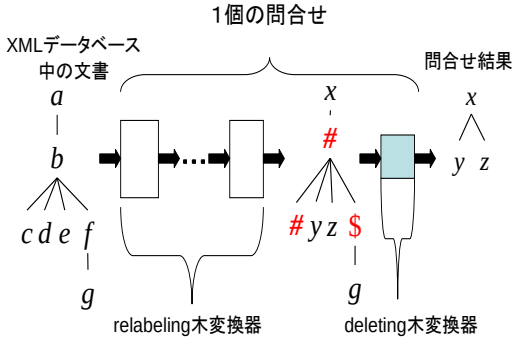

図 2 問合せのモデル

決定性 relabeling 木変換器 T^r は, ラベルの付け替えのみを行う木変換器であり, 動作する方向によってトップダウン型とボトムアップ型がある. トップダウン型は先祖の情報のみに基づいて, またボトムアップ型は子孫の情報のみに基づいて, ラベルの付け替えを行う. 一方, 決定性 deleting 木変換器 T^d は, 木をトップダウンに走査して, ♯ とラベル付けされている頂点を削除する機能と, $ とラベル付けされている頂点以下の部分木を削除する機能をもつ. 形式的には, 木変換器は, 状態の有限集合, 有限アルファベット, 初期状態, 変換規則の集合の 4 つ組で指定される.

そして木変換器の合成の形式, および木変換器の機能を次のように限定する. 合成の形式は, 図 2 のような 0 個以上の relabeling 木変換器の系列と 1 個の deleting 木変換器による合成を考える. また, ユーザに許可される問合せ中の deleting 木変換器は, ♯ とラベル付けされた頂点を削除する機能のみをもち, データベース内の機密情報への問合せ中の deleting 木変換器は, $ とラベル付けされた頂点を削除する機能のみをもつものと限定する.

以上のようにモデル化された問合せは, 周囲の頂点の情報に基づいて, 指定された位置にある頂点のラベルの付け替えや削除をするフィルタリング機能をもつ. データベースへの問合せの多くは, 一部の情報を切り出すフィルタリング機能をもつクラスで表されるため, このモデルは実用的なクラスを対象にしているといえる. この問合せクラスは, XSLT [9] の部分クラスである.

2.4 XML における関数従属性

まずは, 関数従属性の定義に用いるパス表現について述べる.

[定義 4] あるアルファベット Σ について, Σ 上のパス表現の集合 $Path_\Sigma$ を, 以下の構文で表現される集合と定義する.

$$Path_\Sigma ::= \text{`} / \text{'}a \mid \text{`} / \text{'}a\ Path_\Sigma \quad (a \in \Sigma)$$

以下ではパス表現を単にパスと呼ぶ. □

パス表現は,‘ / ’(child 軸) と Σ の要素から構成される. このパス表現のクラスは, XPath [10] の部分クラスである. 以下では, パス表現を単にパスと呼ぶ.

XML における関数従属性 FD は, 次の形式で表現する.

［定義 5］ XML における関数従属性の集合 FD を, 以下のように表現される集合と定義する.

$$FD ::= (H, [x_1, \ldots, x_n], [y_1, \ldots, y_m])$$
$$(H, x_1, \ldots, x_n, y_1, \ldots, y_m \in Path_\Sigma)$$

木 t と関数従属性 $f = (H, [x_1, \ldots, x_n], [y_1, \ldots, y_m])$ が与えられたとき, t が f を満たすとは, パス H で指定される t 中の任意の 2 つの部分木において, $[x_1, \ldots, x_n]$ で指定される値系列に共通部分があるならば, $[y_1, \ldots, y_m]$ で指定される値系列にも共通部分がある, ということであると定義する. このことを, 形式的に書くと以下のようになる. t のルート要素からパス H で指定される t 中の部分木の集合を $\{t_{q_1}, \ldots, t_{q_k}\}$ とする. これを $H(t)$ と書く. このとき, $t_{q_i} \in H(t)$ において, $[x_1, \ldots, x_n]$ で指定される値系列の集合を $\{a_1 \cdots a_n \mid a_1 \in x_1(t_{q_i}), \ldots, a_n \in x_n(t_{q_i})\}$ と定義し, これを $L(t_{q_i})$ と書く. 同様に, $[y_1, \ldots, y_m]$ で指定される値系列の集合を $\{b_1 \cdots b_n \mid b_1 \in y_1(t_{q_i}), \ldots, b_n \in y_m(t_{q_i})\}$ と定義し, $R(t_{q_i})$ と書く. 任意の 2 つの部分木 t_{q_α}, t_{q_β} について, $L(t_{q_\alpha}) \cap L(t_{q_\beta}) \neq \emptyset$ ならば $R(t_{q_\alpha}) \cap R(t_{q_\beta}) \neq \emptyset$ ということが成り立っているとき, 木 t は関数従属性 f を満たすという.

関数従属性 $f \in FD$ に対して, f を満たす木の集合を $TL(f)$ と書く. また, 関数従属性の集合 F に対して, $TL(F)$ は F に含まれる全ての関数従属性を満たす木の集合を表す. □

［例 2］ 図 3 で示される木は, 次の関数従属性 f を満たす.

$$f = (/\text{病院}/\text{患者}, [/\text{患者}/\text{病室}], [/\text{患者}/\text{病名}])$$

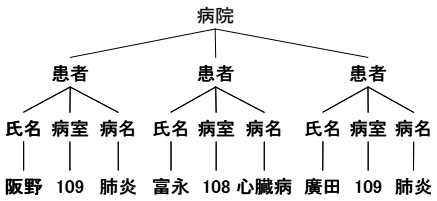

図 3 関数従属性を満たす木の例

3. 安全性の定義

本節では, XML データベースへの関数従属性を用いた推論攻撃に対する安全性を定義する. ユーザに与えられる情報は以下の通りである.

- $T_1, \ldots, T_n$: ユーザに許可されている問合せ.
- $T_1(D), \ldots, T_n(D)$: データベース中の XML 文書 D に対する問合せ $T_1, \ldots, T_n$ の結果.
- A_G: D が従うスキーマ.
- T_S: D 中の機密情報を返す問合せ.
- F: D が満たす関数従属性の集合.

このとき, これらから推論される機密情報の値の候補集合 C は次のように与えられる.

$$C = \{T_S(D') \mid D' \in TL(F) \cap TL(A_G),$$
$$T_1(D) = T_1(D'), \ldots, T_n(D) = T_n(D')\}.$$

そして, C を用いて以下の 2 種類の安全性を定義する.

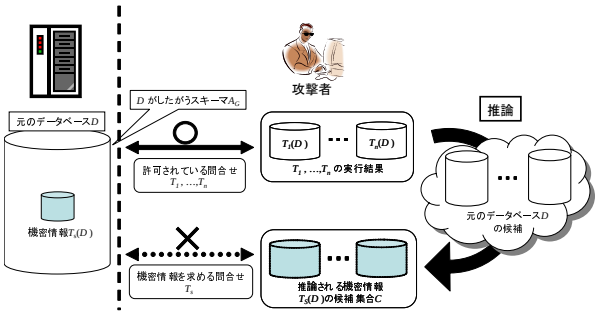

図 4 安全性の定義

無限安全性 C の要素数が無限個のとき, D の機密情報 $T_S(D)$ は無限安全であるという.

k 安全性 C の要素数が k 個以上のとき, D の機密情報 $T_S(D)$ は k 安全であるという.

4. 安全性検証法

本節では, データベースへの関数従属性を用いた推論攻撃に対する安全性の検証法について概要を述べる.

本検証法は次の 3 つのステップからなる (図 5).

1. ユーザに許可されている問合せ $T_1, \ldots, T_n$ とその実行結果 $T_1(D), \ldots, T_n(D)$, そしてデータベースが従うスキーマを表す木オートマトン A_G から, 元の XML 文書 D の候補を受理する木オートマトン A_D を求める.

$$TL(A_D) = \{D' \mid D' \in TL(A_G),$$
$$T_1(D) = T_1(D'), \ \ldots, \ T_n(D) = T_n(D')\}.$$

2. ステップ 1 で求めた A_D と機密情報への問合せ T_S, そしてデータベースが満たす関数従属性集合 F から, 機密情報の値の候補集合 C の要素数が無限 (無限安全) かどうかを判定する.

$$C = \{T_S(t) \mid t \in TL(A_D) \cap TL(F)\}.$$

3. ステップ 2 において, C の要素数が無限でないと判定された場合, C の要素数が k 以上 (k 安全) かを検証する.

以降, 各ステップについて概略を述べる.

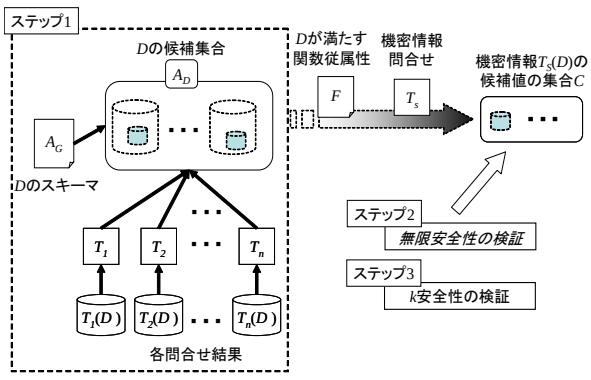

図 5 安全性検証の手順

4.1 ステップ 1: データベース中の XML 文書候補の計算

ステップ 1 は以下の 3 つの手順に分けられる.

(i) まず, 各問合せ $T_i = T_i^d \circ T_i^R$ の実行結果 $T_i(D)$ から, deleting 木変換器 T_i^D の出力が $T_i(D)$ となるような, T_i^D への入力木の候補を受理する木オートマトン $A_i^\sharp$ を構成する. ここで T_i^R は, 問合せ T_i における, 0 個以上の relabeling 木変換器の系列である.

(ii) 手順 (i) で得られた各木オートマトン $A_i^\sharp$ と relabeling 木変換器の系列 T_i^R から, 逆型推論を用いて, 問合せ T_i の実行結果が $T_i(D)$ となるような文書の候補を受理する木オートマトン A_i^D を構成する.

(iii) 手順 (ii) で得られた全ての木オートマトン A_i^D と, データベースのスキーマ A_G の交わりをとることにより, 元のデータベース中の XML 文書 D の候補を受理する木オートマトン A_D を構成する.

4.2 ステップ 2：無限安全性の検証

一般に関数従属性を満たす木の集合を木オートマトンで表現することができない. このため, 機密情報の値の候補集合 C を表す木オートマトンを構成することができない. そこで, 本稿では, 機密情報の値の候補集合 C を陽に求めることなく, ステップ 1 で求めた A_D と, 機密情報への問合せ T_S, そしてデータベースが満たす関数従属性集合 F から, 無限安全性の判定を行う.

また, 各検証手順において, 木オートマトンが受理するインスタンスに少なくとも 1 つは関数従属性を満たすものが存在するかどうかを判定する際には, ボトムアップ決定性の木オートマトンを用いる. ボトムアップ決定性木オートマトンとは, 任意の 2 つの遷移規則 $(q_1, a, e_1), (q_2, a, e_2) \in R$ について, $L(e_1) \cap L(e_2) = \emptyset$ または $q_1 = q_2$ のいずれか一方が成り立つような木オートマトンである. 全ての非決定性木オートマトンについて, それと等価なボトムアップ決定性木オートマトンを構成するできることが分かっている [11]. ボトムアップ決定性では 任意の異なる 2 つの状態にそれぞれ対応する木言語が互いに素であることが保証される. このことから, 関数従属性を満たすインスタンスが存在するための煩雑な条件を, 書き下すことができる.

ステップ 2 は以下の 4 つの手順に分けられる.

(i) 各関数従属性 $f_j = (H_j, [x_{j_1}, \ldots, x_{j_m}], [y_{j_1}, \ldots, y_{j_m}]) \in F$ に関して, インスタンス中に存在する H で指定される部分木の種類が同じであるようなインスタンス集合を表す木オートマトン $A_i^{f_j} (i = 1, \ldots, k)$ を A_D から構成する. これは, 直感的に言うと, A_D のインスタンスを, 関数従属性に関係する部分の構造が同じインスタンス集合に分割するということである (図 6). H で指定される部分木の種類が同じということは, 関数従属性の充足判定結果が同じである. また, k の値は, 有限に抑えられる.

そして, $A_i^{f_j}$ の中で, 以下の 2 条件を満たすものを木オートマトンの集合 $\mathcal{A}^{f_j}$ の要素とする.

- $TL(A_i^{f_j}) \cap TL(f_j) \neq \emptyset$ である.
- $\{T_S(t) \mid t \in A_i^{f_j}\}$ が無限集合である.

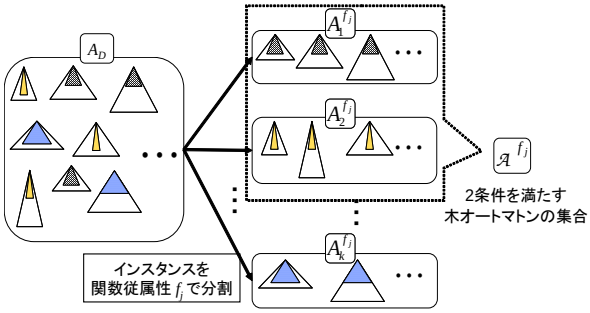

図 6 無限安全性のステップ 2 手順 (i)

このとき, $\mathcal{A}^{f_j}$ の各要素 $A_i^{f_j}$ について, $\{T_S(t) \mid t \in TL(A^{f_j}) \cap TL(f_j)\}$ が無限集合であるということが成り立つ.

$TL(A_i^{f_j}) \cap TL(f_j) \neq \emptyset$ の判定方法

ここで, $TL(A_i^{f_j}) \cap TL(f_j) \neq \emptyset$ の判定方法について簡単に述べる. 構成方法から $A_i^{f_j}$ は以下の性質をもつ.

- $A_i^{f_j}$ が受理する全ての木において, パス H で指定される部分木の頂点に割り当てられる状態の集合は同じである.

まず, 木オートマトン $A_i^{f_j}$ をボトムアップ決定性へと変換する. このとき, $A_i^{f_j}$ が受理する木は同じなので, 変換後も $A_i^{f_j}$ は先の性質をもつ. 変換後の $A_i^{f_j}$ が受理する任意の木において, パス H で指定される部分木の頂点に割り当てられる状態の集合 $A_i^{f_j}[H] = \{q_1, \ldots, q_r\}$ を求める. そして, 各状態 $q \in A_i^{f_j}[H]$ に対応する木言語を表す木オートマトン A^q を構成する. ここで, $A_i^{f_j} = (Q, \Sigma, q_0, R)$ とすると, $A^q = (Q, \Sigma, q, R)$ である. A^q が受理する木において, パス系列 $[x_1, \ldots, x_n]$ に対応する状態系列の集合を $q[X]$ とする. $q[X]$ は以下の式で求められる.

$$q[X] = \{q_1 \cdots q_n \mid q_1 \in A^q[x_1], \ldots, q_n \in A^q[x_n]\}$$

また, 状態 $q \in A_i^{f_j}[H]$ に対して, $q[X] = q'[X]$ となるような状態 $q' \in A_i^{f_j}[H]$ の集合を求め, それを $\langle q \rangle$ と書く. このとき, 任意の状態 $q \in A_i^{f_j}[H]$ について, $|q[X]| \geq |\langle q \rangle|$ が成り立つならば, $TL(A_i^{f_j}) \cap TL(f_j) \neq \emptyset$ であると判定する.

(ii) (i) で得られた各 $\mathcal{A}^{f_j}$ から, 1 つずつ候補を選び, それらの交わりをとることにより, 任意の関数従属性 $f_j \in F$ に対して, H_j で指定される部分木の種類が一意に決まるようなインスタンスの集合を表す木オートマトン $A^{F_i} (i = 1, \ldots, l)$ を構成する. 各 $\mathcal{A}^{f_j}$ の要素数が有限なので, l の値も有限に抑えられる.

(iii) (ii) で得られた $A_i^F (i = 1, \ldots, l)$ において, 次の 2 条件を満たすものがあれば, 無限安全と判定する.

- $TL(A_i^F) \cap TL(F) \neq \emptyset$ である.
- $\{T_S(t) \mid t \in A_i^F\}$ が無限集合である.

4.3 ステップ 3：k 安全性の検証

このステップでは, ステップ 2 の結果から機密情報の値の候補集合 C の要素数が無限にならないことがわかった場合に, C の要素数が k 個以上であるかどうかを検証する. 以下, (a) 木オートマトン A_D が受理するインスタンスの数が有限である場合と, (b)A_D が受理する各インスタンスに対して機密問合せ T_S を実行した結果が有限である場合について, それぞれ k 安全性の検証手順を述べる.

(a) $|TL(A_D)| \neq \infty$ の場合 (図 7)

各インスタンス $t \in TL(A_D)$ について, (1) まず t が関数従属性を満たしているかどうかを判定する. (2)t が関数従属性を満たしていた場合, 次に, $T_S(t)$ を求める. (3) そして, $T_S(t)$ を機密情報の値の候補集合 C に加える. C の要素数が k 個以上になったとき, k 安全と判定する.

(b) $|\{T_S(t) \mid t \in TL(A_D)\}| \neq \infty$ の場合 (図 8)

(1) まず, A_D と T_S から $TL(A_S) = \{T_S(t) \mid t \in TL(A_D)\}$ となるような木オートマトンを構成する. そして, 各 $t \in TL(A_S)$ について, それが機密情報候補であるかどうかを調べる. 具体的には, (2) 出力が t となるような T_S への入力 $t' \in A_D$ の候補を受理する木オートマトン A_D^t を構成する. (3) そして, A_D^t が受理するインスタンスに少なくとも 1 つは関数従属性を満たすものが存在するかどうかを判定する. 関数従属性を満たすものが存在するならば, t を機密情報候補として C に加える. C の要素数が k 個以上になったとき, k 安全と判定する.

$(a)(b)$ 以外の場合, つまりは $|TL(A_D)| = \infty$ であり, なおかつ $|\{T_S(t) \mid t \in TL(A_D)\}| = \infty$ の場合については, 問題が複雑となるため, $(a)(b)$ の場合のように機密情報候補を逐次生成・判定しながら数え挙げる方法では, C の要素数が k 個以上であるかどうかを調べることができない. この問題については, 現在, 検討中である.

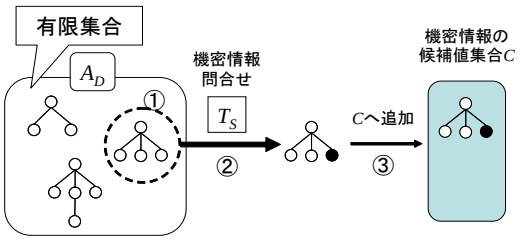

図 7 k 安全性の判定手順 ($|TL(A_D)| \neq \infty$ の場合)

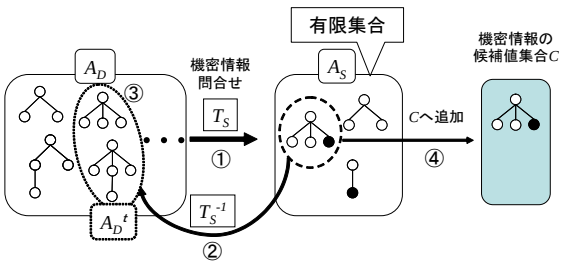

図 8 k 安全性の判定手順 ($|\{T_S(t) \mid t \in TL(A_D)\}| \neq \infty$ の場合)

5. あとがき

本稿では, XML データベースへの関数従属性を用いた推論攻撃に対して, 推論される機密情報の候補を陽に求めることなく, 無限安全性を検証する手法を提案した. また, 一部の場合を除き k 安全性を検証する手法を提案した.

今後の課題としては, まず, 元のデータベース候補が無限個であり, かつ, その各候補に対して機密情報を求める問合せを実行した結果が無限個である場合の k 安全性の判定方法の考案が挙げられる. 次に, 検証法の効率化が挙げられる. 特に, 機密情報の候補値を 1 つずつ生成し数え上げる方法は, 非効率であるので, 数え上げを行わずに機密情報の候補値の数を計算する手法を検討することが, 検証法の効率化を考える上で必要である.

また, 本稿では, 推論攻撃に対して XML データベースが安全であるかどうかの検証法を扱っているが, データベースが安全でないと分かった場合に, どのように安全性を確保すればよいのか, その手法を検討していくことも重要な課題である.

謝　辞

本研究の一部は科学研究費補助金 (課題番号 20500092) によるものである.

文　献

[1] K. Hashimoto, F. Takasuka, K. Sakano, Y. Ishihara and T. Fujiwara: “Verification of the Security against Inference Attacks on XML Databases,” Proc. the 10th APWeb, LNCS 4976, pp. 359–370, 2008.

[2] 阪野 公秀, 橋本 健二, 石原 靖哲, 藤原 融: “XML データベースへの関数従属性を用いた推論攻撃に対する安全性の定式化とある条件下での安全性検証法の提案,” コンピュータセキュリティシンポジウム 2008, B5-1, CD-ROM(論文集 pp. 461–466), 2008.

[3] W. Fan, C.Y. Chan, M.N. Garofalakis: “Secure XML Querying with Security Views,” Proc. ACM SIGMOD Conf, pp. 587–598, 2004.

[4] X. Yang, C. Li: “Secure XML Publishing without Information Leakage in the Presence of Data Inference,” Proc. VLDB, pp. 96–107, 2004.

[5] M. Gerome, S. Dan: “A Formal Analysis of Information Disclosure in Data Exchange,” Journal of Computer and System Sciences, vol. 73, pp. 507–534, 2007.

[6] M. L. Lee, T. W. Ling and W. L. Low: “Designing Functional Dependencies for XML,” Proc. the 8th EDBT, pp. 124-141, 2002

[7] M. Arenas, L. Libkin: “A normal form for xml documents,” Proc. ACM PODS Conf, pp. 85-96, 2002.

[8] M.W. Vincent, J. Liu: “Functional dependencies for XML,” Proc. the 5th APWeb, LNCS 2642, 2003.

[9] “XSL Transformations (XSLT) Version 1.0,” http://www.w3.org/TR/xslt.

[10] “XML Path Language (XPath) Version 1.0,” http://www.w3.org/TR/xpath.

[11] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison and M.Tommasi: “Tree Automata Techniques and Applications,” Detailed information at http://tata.gforge.inria.fr/, 2007.