デバッグ支援のための文字列解析によるSQL構文木生成法改良について

Li Heng† 西田誠幸‡
拓殖大学工学部情報工学科§

1 はじめに

データベースの問い合わせに用いられるSQLクエリの組み立て処理の間違いを原因とする，Webアプリケーションの脆弱性が問題となっている．この問題を解決するために，文字列解析[1, 2]と呼ばれる技術が提案されている．この文字列解析は，プログラムリストを静的に解析し，実行中に組み立てられるSQLクエリを事前に予測する技術である．この方法は，SQLクエリが不適切だと判断されると，プログラムリストにおけるSQL実行文の位置を出力する．開発者はこの位置に従ってプログラムを修正することになるが，不適切なのはSQL実行文でなく，その引数SQLクエリを組み立てる処理にあるため，開発者がSQL実行文の位置を頼りに，複雑なプログラムの中の問題箇所を探し出すことを強いられる．

本稿では，SQLクエリ実行文を含むPHPプログラムリストの文や式と，文字列解析結果のSQLクエリの各項との対応関係を作る手法について述べる．この方法を使うことによって，文字列解析で見つかる問題箇所として，SQL実行文だけでなく，SQLクエリの組み立て処理のプログラムリストにおける位置を出力することが可能となり，開発者に対して問題箇所のより有益な情報を出力することが期待される．

2 文字列解析処理の概要

本稿の基礎である文字列解析[1]の処理についてここで述べる．文字列解析は大きく4つの処理からなる．

1. 解析対象(ホットスポット)の検出
2. ホットスポットに対するデータフローの生成
3. ホットスポットに対するSQLクエリを受理するオートマトンの生成
4. SQL構文木の生成

文字列解析器は，まず解析の対象となるSQL実行文の引数をホットスポットとして検出し，このホットスポットについてここに到るデータフローを生成する．例えば，次のPHPプログラムは，ユーザが入力した名前を処理を表す検索キーにしてユーザのすべての情報を取得する．ただし，escape_sq() は文字列 $n の中のシングルクォート ' を \' に置換する(すなわちエスケープ処理する)関数とする．

```
<?php
    $n = escape_sq("$_GET['name']");
    $sql = "SELECT * FROM t WHERE n='$n'";
    mysql_query($sql);
?>
```

このプログラムの最後の行のSQL実行文の引数 $sql に対するデータフローを図1に示す．データフローの各ノードはPHP構文木のノードであり，有向辺はノードの間で文字列データが受渡しされることを意味する．また concat のノードは文字列の連接を，それ以外の四角のノードは文字列定数を表す．なお丸はユーザ入力文字列 ($_GET['name']) を表すものとする．

文字列解析の第3段階として，データフローからオートマトンを生成する．このオートマトンは，プログラム実行中にホットスポットに代入される可能性のある文字列の集合を表す．例えば図1のデータフローから図2(a)のオートマトンが生成される．図中の四角はシングルクォートを除く任意の文字を表す．

最後に，オートマトンに対して字句解析と構文解析を行ってSQL構文木を生成する．

3 プログラムとSQLクエリとの対応関係の作成方法

プログラムリストの文や式と，文字列解析によって得られるSQLクエリの各項との対応関係を作る方法について，その概要を述べる．

文字列解析のうち，前述の処理のうちオートマトンを生成する直前までは，プログラムの構文木にしたがって解析が進められる．よってここまでに作られる中間データとプログラムリストの各項との対応付けが可能である．

一方オートマトンとプログラムの構文木との間には対応関係がもともと考慮されていない．これは，前述の第3段階の処理を高速化するためにデータサイズが最小になるようオートマトンを変形するからである．図2(a)では，文字列の始まりを表すシングルクォートと文字列中に現れるシングルクォートが同じ遷移にまとめられている．

そこで，対応関係を記録するために，オートマトンの各遷移に追加情報としてデータフローのノードの参照を付加することにする．その様子を図2 (b)に示す．この情報は次の性質を満たすように各遷移に付加

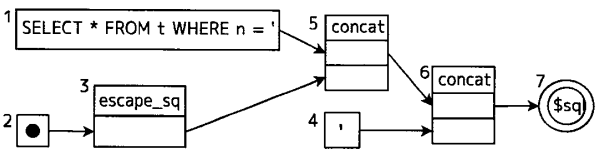
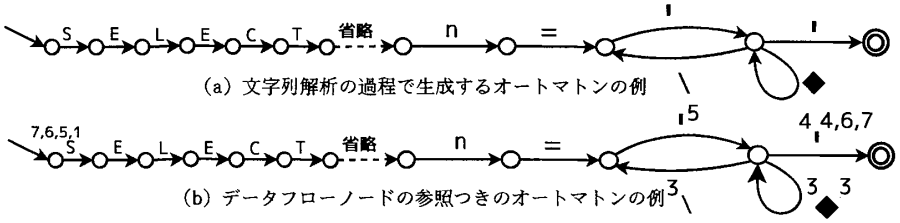


図1 変数 $sql におけるデータフロー

On Improvement of the Algorithm of SQL Syntax Tree Generation by String Analysis
†Li Heng
‡Seikoh NISHITA
§Dept. of Computer Science, Faculty of Engineering, Takushoku University

表 1　実験結果

名称 (バージョン)	解析成功数 (SQL 実行文取得数)	位置情報取得数 (SQL データ値数)	実行時間 (秒) (本手法を利用しない場合の実行時間)
Utopia News Pro(1.4.0)	143(147)	229(231)	105.3(66.8)
Bzeeet(2.1.3)	28(30)	31(41)	60.4(23.7)
TigerPhpNewsSystem(1.0)	53(118)	59(59)	1845.1(682.2)



(a) 文字列解析の過程で生成するオートマトンの例

(b) データフローノードの参照つきのオートマトンの例

図 2　文字列解析の過程で生成されるオートマトンと追加情報を付加したオートマトンの例

することとする．

> 各ノード n について，n を左肩に持つ遷移の遷移前状態を初期状態，n を右肩に持つ遷移の遷移後状態を受理状態とする部分オートマトンを An とする．このとき，ノード n が表す言語は An が受理する言語に含まれる．

例えば，データフローノード 4 の言語はシングルクォート 1 文字だけであるが，図 2(b) のオートマトンにおいて前述の部分オートマトン A4 が受理する言語も同じくシングルクォート 1 文字である．また，ノード 3 の言語は任意の文字列のシングルクォートをエスケープした文字列の集合であるが，これは A3 が受理する言語と一致する．一方ノード 1 の言語は文字列「SELECT * FROM t WHERE n = '」1 つだけであるが，A1 が受理する言語はこの文字列以外にも「SELECT * FROM t WHERE n = 'abc\'\'」などの文字列を含む集合となる．これは，解析処理の高速化のために，オートマトンが最小化されているためである．また，ノード 2 の参照がオートマトンには存在しないが，この理由は `escape_sq()` 関数の適用によって，上記性質を満たすように参照を付加することができなくなるためである．一般に文字列関数を適用すると，同じ理由から適用前の状態の付加情報は削除される．

4　実装と実験

本手法を組み込んだ文字列解析器を Java SE 6 を使って実装した．なお対象言語には PHP 4 を採った．またデータフロー解析に Pixy，オートマトン生成処理に String パッケージを用いた．

本手法は，無償の PHP パッケージ，Utopia News Pro, tigerPhpNews, Bzeeet に対する解析を行った．この実験では SQL クエリのリテラルのデータ値（整数，文字列，日付など）の位置を取得できるかどうかを調べた．また，本手法による手間の増大量を知るために，位置情報を持たない文字列解析に要した時間と本手法を用いた解析に要した時間を計測してその比較を行った．なお，実験に使用した計算機は，Xeon X5260 3.33GHz, メモリ 12GB であった．

実験結果を表 1 に示す．列「解析成功数」は文字列解析を正しく行えた SQL クエリの個数を表す．また括弧内は SQL 実行文の取得数を表し，この 2 つの数値の差が小さいほど，文字列解析の成功率が高いといえる．列「位置情報取得数」は文字列解析に成功した SQL クエリのデータ値のうち，位置情報を正しく取得できたデータ値の個数を表す．また括弧内は SQL クエリもともと含まれているデータ値の個数を表し，この 2 つの数値の差が小さいほど，位置取得の成功率が高いといえる．成功率が低い原因は配列を使った反復処理，未対応の関数などデータフロー解析の失敗によるものと考えられる．また，データ値が PHP の定数文字列に埋め込まれていることが原因で，位置情報を取得できない場合が多く見られた．

5　まとめ

本稿では，SQL クエリを組み立てる際に使うプログラムリストの文や式と，SQL クエリの各項との対応関係を作る方法について述べた．対応関係を作るために，文字列解析の過程で作られるデータフローの各ノードの参照を，オートマトンの遷移に付加する方法をとった．また，本方法を実装して，3 つの Web アプリケーションに対して適用した結果，約 70% についてプログラムの構文木のノードとの対応関係を正しく構築できた．

参考文献

[1] C.Gould, Z.Su, and P.Devanbu. Static Checking of Dynamically Generated Queries in Database Applications. In Proc. of the 26th Software Engineering International Conference, pp.645–654, 2004.

[2] A.Christensen, A.Moler, and M.Schwartzbach. Precise Analysis of String Expressions", In Proc. of the 10th International Static Analysis Symposium, pp.1–18, 2003.