CREATIVE MATH. & INF.
17 (2008), No. 2, 115 - 120

Online version available at http://creative-mathematics.ubm.ro/
Print Edition: ISSN 1584 - 286X Online Edition: ISSN 1843 - 441X

Electronic Transaction System

COSMIN SABO

ABSTRACT.

This paper presents an Electronic Transaction System (ETS) that can interact with different communication languages using a context-free language. ETS can be used in many economical and cultural projects. This concept is already implemented in construction of two projects from commodities exchange and library information exchange.

1. INTRODUCTION

Z39.50 is a client server protocol for searching and retrieving information from remote computer databases.

ANSI/NISO Z39.50 is the American National Standard Information Retrieval Application Service Definition and Protocol Specification for Open Systems Interconnection. The National Information Standards Organization (NISO), an American National Standards Institute (ANSI) that accreditates standard developers that serve the library, information and publishing communities, approved the original standard in 1988 (referred to as Z39.50-1988 or Version 1). NISO published a revised version of the standard in 1992 (referred to as Z39.50-1992 or Version 2).

ANSI/NISO Z39.50 defines a standard way for two computers to communicate for the purpose of information retrieval. Z39.50 makes it easier to use large information databases by standardizing the procedures and features for searching and retrieving information. Specifically, Z39.50 supports information retrieval in a distributed, client and server environment where a computer operating as a client submits a search request (i.e., a query) to another computer acting as an information server. Software on the server performs a search on one or more databases and creates a resulting set of records that meet the criteria of the search request. The server returns records from the resulting set to the client for processing. The power of Z39.50 is that it separates the user interface on the client side from the information servers, search engines and databases.

Z39.50 provides a consistent view of informations from a wide variety of sources and offers to client implementors the capability to integrate information from a range of databases and servers.

Z39.50 sacrifices different databases functionality for user interface.

Z39.50 was built using Open Systems Interconnection (OSI) protocols and in 1992-1993, a program called the Z39.50 Interoperability Testbed was launched under the sponsorship of the Coalition for Networked Information (CNI). The purpose of this project was to facilitate the development of a large number of interoperable implementations of Z39.50 which run over TCP/IP and are accessible through the Internet. This effort was a substantial success and led to a number of demonstrable Z39.50 clients and servers which could be seen to communicate with each other at trade shows like the American Library Association's exhibits.

In Figure 1 is described how an interrogation over TCP/IP is made.

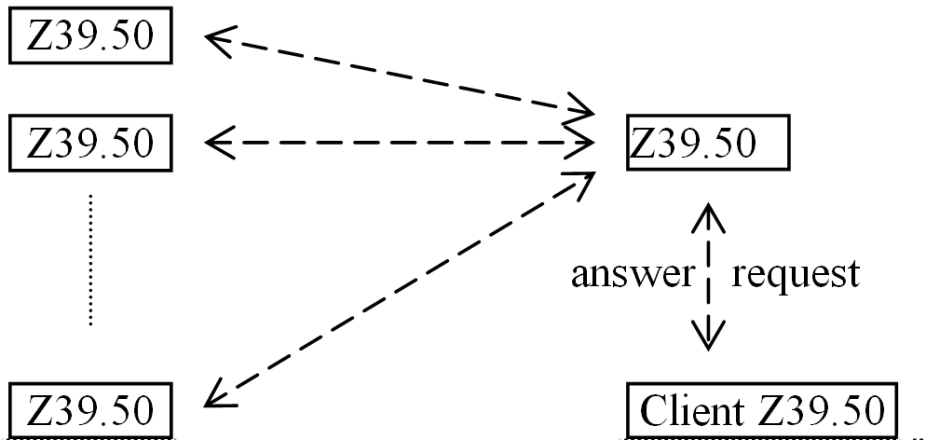

Figure 1. Interrogation over TCP/IP

The main inconvenient of this system is that it can not interact with other systems.

Received: 16.09.2008; In revised form: 12.11.2008.; Accepted:
2000 *Mathematics Subject Classification.* 94C99, 68N15.
Key words and phrases. *Electronic transaction system, context-free languages.*

In this paper, we built a communication system that makes possible for different communication languages to interact. Also, we design the interaction with the client as a communication that can be defined using a free-context language.

2. Preliminaries

A formal grammar $G = (N, \Sigma, P, S)$ consists of:

(1) a finite set of terminal symbols Σ;
(2) a finite set of nonterminal symbols N;
(3) a finite set P of production rules with a left- and a right-hand side consisting of a sequence of these symbols, $P \subseteq (N \cup \Sigma)^* N (N \cup \Sigma)^* \times (N \cup \Sigma)^*$;
(4) a start symbol $S \in N$.

A formal grammar defines (or generates) a formal language, which is a (possibly infinite) set of sequences of symbols that may be constructed by applying production rules to a sequence of symbols which initially contains just the start symbol. A rule may be applied to a sequence of symbols by replacing an occurrence of symbols on the left-hand side of the rule with those that appear on the right-hand side acording to the production ruls. A sequence of rule applications is called a derivation. Such a grammar defines the formal language of all words consisting of terminal symbols that can be reached by a derivation from the start symbol.

Type-0 grammars (unrestricted grammars) include all formal grammars. They generate exactly all languages that can be recognized by a Turing machine. These languages are also known as the recursively enumerable languages. Note that this is different from the recursive languages which can be decided by an always-halting Turing machine.

Type-1 grammars (context-sensitive grammars) generate the context-sensitive languages. These grammars have rules of the form

$$\alpha A \beta \rightarrow \alpha \gamma \beta$$

with A a nonterminal, $A \in N$ and α, β and γ strings of terminals and nonterminals, $\alpha, \beta, \gamma \in (N \cup \Sigma)^*$ and $(A, \gamma) \in P$. The strings α and β may be empty, but γ must be nonempty. The rule $S \rightarrow \varepsilon$ is allowed if S does not appear on the right side of any rule. The languages described by these grammars are exactly all languages that can be recognized by a linear bounded automaton (a nondeterministic Turing machine whose tape is bounded by a constant times the length of the input.)

Type-2 grammars (context-free grammars) generate the context-free languages. These are defined by rules of the form $A \rightarrow \gamma$ with A a nonterminal and γ a string of terminals and nonterminals and $(A, \gamma) \in P$. These languages are exactly all languages that can be recognized by a non-deterministic pushdown automaton. Context free languages are the theoretical basis for the syntax of most programming languages.

Type-3 grammars (regular grammars) generate the regular languages. Such a grammar restricts its rules to a single nonterminal on the left-hand side and a right-hand side consisting of a single terminal, possibly followed (or preceded, but not both in the same grammar) by a single nonterminal, $A \rightarrow aB$, $A \rightarrow a$, $a \in \Sigma$, $A, B \in N$. The rule $S \rightarrow \varepsilon$ is also here allowed if S does not appear on the right side of any rule. These languages are exactly all languages that can be decided by a finite state automaton. Additionally, this family of formal languages can be obtained by regular expressions. Regular languages are commonly used to define search patterns and the lexical structure of programming languages.

3. Main results

The communication system called Electronic Transaction System (ETS) can interact with different communication languages using a context-free language. ETS is create to make direct exchanges and transactions between each other for a wide variety of services and goods. In this way, the ties between people can be strengthened, mutual assistance and recycling of resources can be promoted, and communities can be revitalized. ETS can be used in many economical and cultural projects.

The Electronic Transaction System has the next features:

(1) The server application can interpret informations from other servers based on context-free language received in CSV or XML formats.
(2) The server can send and receive data over TCP/IP using socket connections or HTTP request.
(3) The translation from one language to another are defined in XML format.
(4) The actions are described in XML format.

The solution of the problem from above is to define a new communication language and a translation schema over actions to communicate with a new remote client/server as presented in Figure 2.

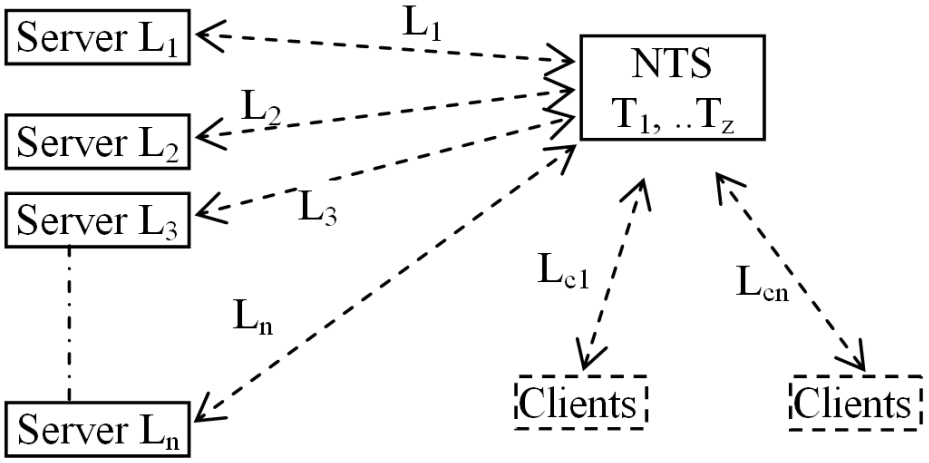

Figure 2. Communication language and translation schema

In Figure 2, we have:

- Server L_1, server L_2, ..., server L_n– represent servers for languages L_1, L_2, ..., L_n
- L_1, L_2, ..., L_n, $L_{c1}, \dots L_{cn}$– represent languages based on context-free grammars
- $T_1, \dots T_z$– translation tables

One implementation of ETS is for financial services. One client needs to receive information simultaneous from more financial exchange institutions. Each of them uses its own communication protocol. The clients want to receive information using different protocols.

In this case to communicate with each financial exchange institution means to define another language. The results are send to clients based on different communication protocols and languages.

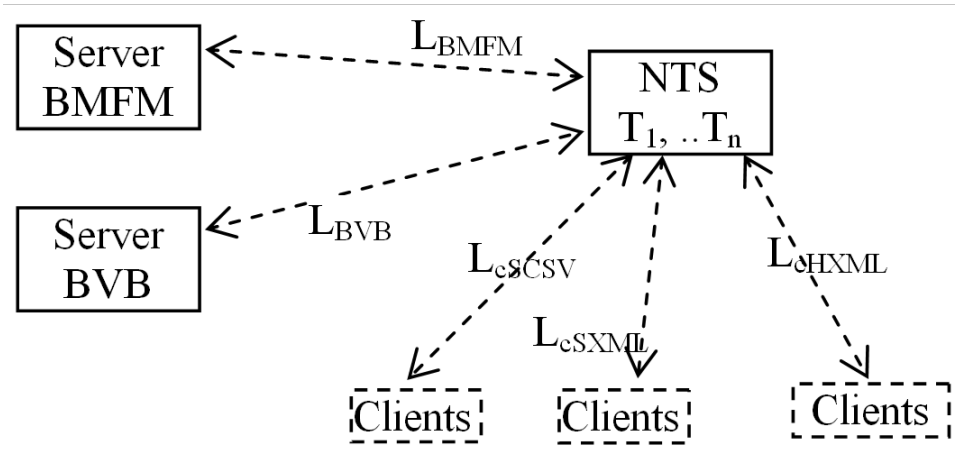

Figure 3. Example of ETS

The resulting design is presented in Figure 3, where

- L_{BMFM}, L_{BVB}- represent languages based on context-free grammars
- $LcSCSV$, $LcSXML, \dots, LcHXML$– represent languages based on different send and request type (S– socket, H– HTTP) and different type of data format (CSV– Comma-Separated Variable, XML – Extensible Markup Language)
- $T_1, \dots, T_n$– translation tables.

Considering this the description schema for a free-context language is presented.

```
"lang.xml''
lang
  header
    name name/name
    type csv|hhtp|xml/type
    separator :|\t|&|.../separator
    version version/version
  /header
  body
    field
        name name/name
        [regex REGEX/regex ]
        [dataType Integer|String|Float| ...  | Word/dataType ]
       [repeat null|int/repeat ]
       [types
         [detByField int/detByField ]
           TYPE
 [regex REGEX/regex ]                word WordName.xml/word
      /type
.....
    /types
   /field
  /body
 /lang
```

Using this description we can identify the elements of the grammar:

(1) a finite set of terminal symbols defined by
```
field ...  regex REGEX/regex .../field
```
(2) a finite set of nonterminal symbols defined by
```
word WordName.xml/word .
```
(3) a finite set of production rules defined by files like WordName.xml where the left-hand is WordName and the right-hand side is the description schema.
(4) the start symbol is lang.

In the next example we will take a simple input and we will analyze him:

```
f44cafe7c043a0f4a50a18796d3f731c:simple:ticker_deal:1   REGS    BRD 1,000   39.0000 0.0000  0
6,000 39.0000 2   2007-04-05   13:29:33 1.04
```

```
"lang.xml''
lang
   header
      name bvb/name
      type csv/type
      separator :/separator
   /header
   body
      field
name check/name
regex [\p{ASCII}]+/regex
dataType String/dataType
repeat 2/repeat
      /field
      field
name message/name
regex [\p{ASCII}]+/regex
dataType Word/dataType
types
      detByField 2/detByField
   count 2/count
   type
      regex ticker_bet/regex
word TickerBet.xml/word
   /type
   type
regex ticker_deal/regex
word TickerDeal.xml/word
separator \t/separator
repeat /repeat
   /type
 /types
       /field
    /body
 /lang
"TickerDeal.xml''
lang
   header
      name Ticker/name
      type csv/type
      separator ;/separator
    /header
    body
      field ....
      field ....
      field
name symbol/name
regex [a-zA-Z0-9]+/regex
```

```
dataType String/dataType
      /field
....
      field
name change/name
```

```
regex [0-9,]+.[0-9]+/regex
dataType Float/dataType
      /field
   /body
/lang
```

REFERENCES

[1] Chomsky, N., *Three models for the description of language*, 1956, IRE Transactions on Information Theory, (2), 113-124
[2] Chomsky, N., *On certain formal properties of grammars*, Information and Control, (2), 1959, 137-167.
[3] Chomsky, N., Schützenberger, Marcel P., *The algebraic theory of context free languages*, in Braffort, P.; Hirschberg, D.: Computer Programming and Formal Languages, Amsterdam, North Holland, 1963, 118-161
[4] OSI , *Application Service Definition and Protocol Specification for Open Systems Interconnection*, 1994
[5] Clifford A. Lynch, *The Z39.50 Information Retrieval Standard*, D-Lib Magazine ,1997
[6] Popa, E. M., *Mechanisms generation of economic processes* (in Romanian), Vol. I, II, Alma Mater, 2002, Sibiu
[7] Popa, E. M., *Modern technologies of information processing* (in Romanian), Alma Mater, 2005, Sibiu
[8] Popa, E. M., *Processing semantics with microprocessors* (in Romanian), Vol. I, II, Alma Mater, 2000

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
FACULTY OF SCIENCES
NORTH UNIVERSITY OF BAIA MARE
BAIA-MARE 430122, ROMANIA
E-mail address: cosmin_sabo@scream.ro