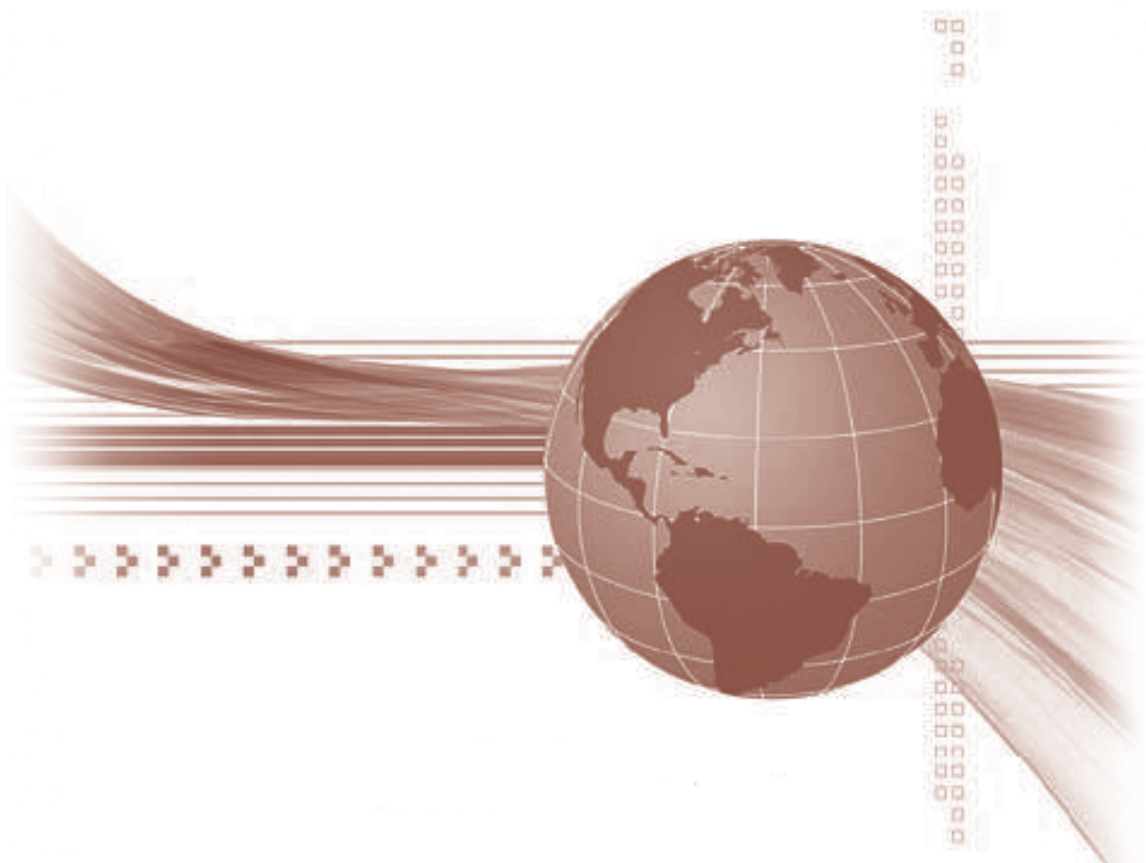
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

4/2011

YEAR **(LVI) 2011**
MONTH **DECEMBER**
ISSUE **4**

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI
INFORMATICA

4

EDITORIAL OFFICE: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 4, 2011

PROTEIN CHANNEL FORMATION IN P SYSTEMS

LAMIAA HASSAAN AHMED[1], AMR BADR[2], AND IBRAHIM FARAG[2]

Abstract. Membrane computing is an area of computer science aiming to abstract computing ideas and models from the structure and the functioning of living cells, as well as from the way the cells are organized in tissues or higher order structures. Membrane systems are usually accompanied by transition rules that represent cellular chemical reactions and transportation rules that represent the movement of cellular molecules through different membranes without changing those molecules. This paper is proposing the addition of channel formation rules to the theory of membrane systems. A simple software simulating the events leading to channel formation within the mitochondrial membrane is used to test the newly suggested rules and the results are satisfactory.

1. Introduction

A P system is one of the natural computing fields that is inspired from the structure and functions of a living cell. In a P system, chemical reactions of a cell are represented by rules. Transition rules and communication rules are the most common types of rules accompanying P systems. Transition rules transform the P system from a status to another by removing and generating objects, i.e., cellular molecules [4]. Communication rules symbolize the movement of objects through membranes [4]. Communication rules assume the preexistence of channels to transport molecules through. Also, some types of rules dealing with the membrane itself were proposed in [4], like membrane division rules and membrane dissolving rules. We are proposing a new type of rules that can accompany a P system. They are the channel formation rules. A channel formation rule allows us to add a new type of data to a P system

Received by the editors: June 12, 2011.

2010 *Mathematics Subject Classification.* 68Q10, 68U20, 92C37.

1998 *CR Categories and Descriptors.* I.1.4 [**Computing Methodologies**]: Symbolic and Algebric Manipulation – *Applications*; I.6.3 [**Computing Methodologies**]: Simulation and Modeling – *Applications*.

Key words and phrases. Channels in P systems, Membrane Computing, P Systems.

and also to use this data as a communication means within the system. This means that a new event in P systems can occur. In addition to membrane division and dissolving, a channel can be created within a membrane and used to transport objects between the two regions sharing this membrane. In this paper, this new type of rules is presented. An application of the channel formation rules is done on the case study of the mitochondrial pathway to cell death. Section 2 gives some of the basic concepts of the P system theory, section 3 explains the biological background needed to understand the case study, section 4 shows a brief history of simulators of cellular activities using P systems, section 5 shows the experiments and results of the proposed case study, and finally section 6 that introduces the results and recommendations of the paper.

2. P Systems

In this section, we are introducing some of the basic definitions and components of the P systems theory.

A P system is a branch of natural computing whose initial goal is to abstract computing models from the structure and the functioning of living cells [3]. It is a computational model that is based on the idea of cellular membrane structure and functions; it was presented by G. Paun [3]. The chemical reactions controlling the change of molecules are represented by evolution rules -also called multiset rewriting rules [3,4]- and the chemical reactions controlling transportation of molecules without changing them are represented by communication rules [3,4]. Communication rules can either be symport/antiport rules or rules with carriers [4]. The above different types of rules are employed and implemented by the P system with the target of transforming from a computational status to another. A simple transition P system is constructed of the form [4]:

$$\Pi = \langle O, C, \mu, w_1, w_2, ..., w_n, R_1, R_2, ..., R_n, i_o \rangle.$$

where:

O: The alphabet of objects, i.e. cellular molecules.

C: The alphabet of catalysts, if any.

μ: The membrane structure. It consists of n membranes labeled with 1,2,3,..,n.

$w_1, w_2, ..., w_n$: The strings over $O \cup C$, representing the multisets of objects initially present in all regions of the system membrane structure [4].

$R_1, R_2, ..., R_n$: The set of evolution rules associated with the regions of the system.
i_o: The output region. It will take one of the labels 1,2, ...,n.

Objects are assigned to rules by choosing rules and objects nondeterministically. Also, the chosen multiset of rules should be applicable to the chosen multiset of objects currently available. When no other rules can be applied on the current multiset of objects, the multiset of rules is said to be maximal. Different rules can be applied on different objects in parallel. We can conclude that P systems run in a maximal parallel nondeterminitic manner [4].

3. The biological background of cell death

This section is concentrating on the biological processes of a living cell that cause channel formation and cell death. Biological concepts, chemical reactions and events leading to channel formation are discussed.

3.1. **Protein Channels.** A protein channel is a watery pathway through the interstices of a protein molecule by which ions and small molecules can cross a membrane into or out of a cell. Protein channels play a vital role in depolarization and repolarization of nerve and muscle fibers, and may have physical characteristics such as shape or diameter that particularly attract certain ions [15]. Various cellular conditions and chemical reactions lead to the formation of a protein channel. Also, most of the protein channels obey certain rules to open and close in order to organize the concentration of different ions and molecules inside the cell.

3.2. **Apoptosis.** Apoptosis which is also known as Programmed Cell Death (PCD) is a conserved cell death mechanism essential for normal development and tissue homeostasis in multi-cellular organisms [11]. These cells are either superfluous or potentially harmful, for instance, apoptosis can serve a protective function by killing off virus-contaminated cells before they spill over with virus particles. This type of cell death is a natural, carefully planned part of the cell life cycle. During apoptosis, the cell shrinks and pulls away from its neighbors [6]. Then, the surface of the cell appears to boil, with fragments breaking away and escaping like bubbles from a pot of boiling water [6]. The DNA in the nucleus condenses and breaks into regular-sized fragments, and

soon the nucleus itself, followed by the entire cell, disintegrates [6]. A cellular cleanup crew rapidly mops up the remains [6]. Apoptosis is led to by a great deal of cellular signals, proteins and pathways. One of the most famous organelles that play an important role of regulation of apoptosis is the mitochondrion. We will focus on the mitochondrial pathway of apoptosis which is also known as the intrinsic pathway.

3.3. **The Mitochondrial Regulation.** There are various proteins that promote apoptosis such as bax protein and others that inhibit apoptosis like bcl-2 and bcl-X_L proteins. The bax protein exists in the cytosol of the cell and bcl-2 and bcl-X_L exist in the space between the outer and inner mitochondrial membranes. Bax has the tendency to form a channel through the outer mitochondrial membrane, and bcl-2 and bcl-X_L tend to prevent it. A trophic factor or a growth factor is a cellular signal that helps the cell to stay alive, figure 1-a [10]. It provides a signal blocking the path to apoptosis. The binding of the trophic factor to a specific trophic factor receptor on the cell surface triggers the signaling cascade resulting in the phosphorylation of the Bcl-2-associated death promoter, the Bad protein, see figure 1-b [10]. Bad is one of the apoptosis initiators. This phorphorylated Bad protein is unable to bind and inhibit the bcl-2-bcl-X_L complex at the outer mitochondrial membrane. That is, the anti-apoptotic activity of this complex is unhindered and the cell survives figure 1-c [10]. The chemical equations 1, 2 represent the reactions the cell takes to survive. In the absence of the trophic factor, most cells undergo apoptosis.

Trophic_factor+$Bad \rightarrow$ Phosphorylated_Bad. (1)
Phosphorylated_Bad +$bcl\_2\_bcl\_X_L \rightarrow$ Inactive_Bad. (2)

Without a trophic factor signal, the Bad protein is un-phosphorylated and associates with the bcl-2-bcl-X_L complex inhibiting anti-apoptotic activity [10], see figure 2-a [10] and figure 2-b [10]. When the bcl-2-bcl-X_L complex is inhibited, the pro-apoptotic regulator Bax is free to permit the influx of ions through the mitochondrial membrane by creating a channel [10]. This channel is called the Mitochondrial Apoptosis-Induced Channel or MAC [8], see figure 2-c [10]. Once the ions go through the channel into the mitochondrion, the release of cytochrome c -which is a protein found loosely associated with

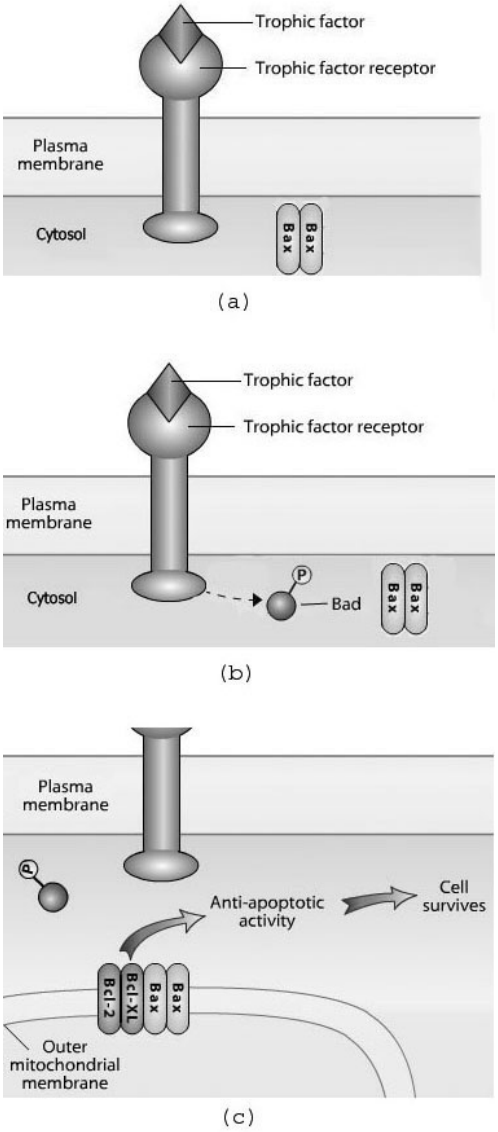


FIGURE 1. Signals of cell survival

the inner membrane of the mitochondrion- to the cytosol through the channel is triggered as in figure 2-d [10]. In cytosol, cytochrome-c binds to the Apoptotic Protease Activating Factor-1 (Apaf-1) which activates the caspase cascade, see figure 2-e [10]. Caspases are a family of special enzymes called

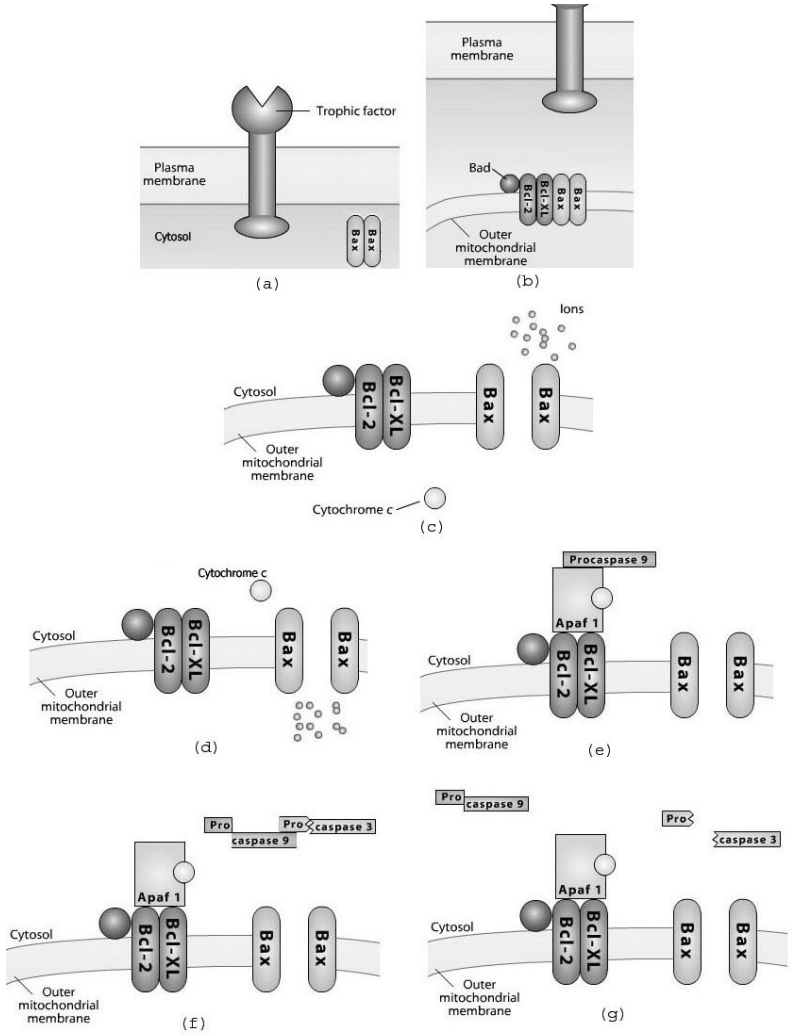


FIGURE 2. Signals of cell death

cysteine proteases that are targeting the proteins of the nuclear lamina and cytoskeleton in order to fragment them as a step of cell death. The complex Apaf-1-cytochrome-c binds to procaspase-9 to create a protein complex known as an apoptosome [2,8]. The apoptosome cleaves the procaspase to its active form of caspase-9, which in turn activates the effector caspase-3, see figure 2-f [10] and figure 2-g [10]. The chemical reactions 3, 4 and 5 represent the events

that lead to cell death:

Cytochrome_c + Apaf_1+ procaspase_9 → apoptosome. (3)
Apoptosome + procaspase_3→caspase_3. (4)
Caspase_3 → Apoptosis. (5)

4. Literature Survey

This section is about previous researches done in P system with channels as well as some of the P simulators of cellular activities.
Communication within a P system through channels was achieved by symport/antiport rules in [1]. Means of channel controllers were added to formulate channels that can be activated or prohibited in [13]. In [9] a variant of P Systems is defined in which transport of objects across the regions of the system is possible by means of rules associated with membranes and involving proteins attached to them. Such proteins can be attached/deattached to/from membranes by means of rules.
A biological model of Mechanosensitive Channels and their conformations (closed, extended closed, subconducting open and open) was represented in the framework of P systems in [14]. A P-simulator of the mitochondrial functions and chemical reactions to produce energy was published in [7]. The cellular respiratory system of the bacterium Escherichia Coli was presented using logic gates in [5]. There were other published metabolic simulators like MetaPlab which was produced by the Italian university Verdona [16]. It was a deterministic P system developed to model dynamics of biological phenomena related to metabolism. It added a new plugin based framework for processing metabolic P systems. Also, there was a simulator for Biological Processes produced by University of Sheffield, UK in 2006[18]. It simulated the evolution of Multi-compartmental Gillespie algorithm over a hierarchy of compartment structures. Another simulator, CytoSim simulator which was produced by the Microsoft Research - University of Trento Centre for Computational and Systems Biology, Trento, Italy. This simulator was a stochastic simulator of biochemical processes in hierarchical compartments. The compartments might be isolated or may communicate via peripheral and integral membrane proteins [19]. There are many other simulators that are concerned with the implementation of P systems on other research fields like neural networks, dynamical probabilistic P systems and membrane approximation algorithms [17].

5. The Problem and an Application

This section is defining the problem, a method to solve it and some experiments on this method. The experiments are employing the case of the mitochondrial activities that cause one protein channel or more to be formed within the mitochondrial membrane.

5.1. **The problem.** In a living cell, some chemical reactions may lead to form a channel within a given membrane. On the other hand, channels are assumed to be preexisting in P systems. Therefore, a new type of rules that can create a channel within a given membrane is to be introduced. Also, the type of objects passing through the formed channel is to be controlled by traditional evolution rules.

5.2. **The Construct of the P System.** We are proposing the following P system construct that models the channel formation within the mitochondrial membrane. This construct includes evolution rules in addition to the proposed channel formation rules. The following P system construct is illustrated in figure 3.

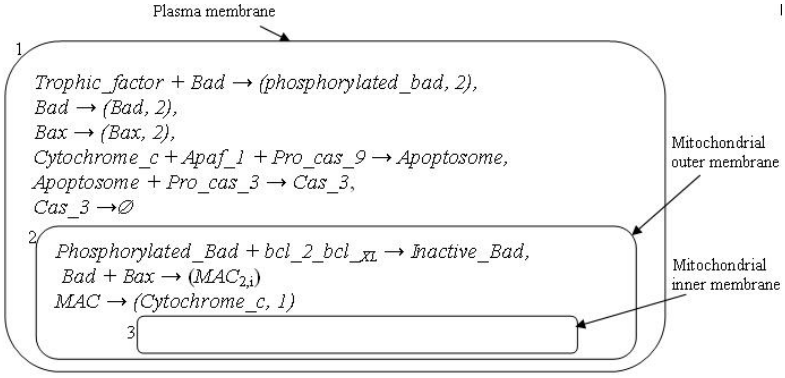


FIGURE 3. The mitochondrial pathway to apoptosis

The P system will be in the form:

$$\Pi_{sim} = \langle O, \mu, Ch_{2,n}, w_1, w_2, R_{e1}, R_{e2}, R_{c2} \rangle.$$

where:
- O: The alphabet of objects.
O= $\{Trophic\_factor, Bad, phorphorylated\_Bad, bcl\_2\_bcl\_XL, Bax, Apaf\_1, procaspase\_9, procaspase\_3, MAC, Cytochrome\_c, apoptosome, caspase\_3, Inactive\_Bad\}$.

-μ: The membrane structure [4]. The plasma membrane will take label 1, the outer mitochonrdial membrane will take label 2 and the inner mitochondrial membrane will take label 3. The structure is $\mu = [_1[_2[_3]_3]_2]_1$.

-$Ch_{2,n}$: The newly formed n channels over membrane two. Notice that for any given membrane m, the following equality shows the n number of channels belonging to m:

$$Ch_{m,n} = \{MAC_{m,1}, MAC_{m,2}, ..., MAC_{m,n}\}.$$

This means that the channels formed over membrane two are:

$$Ch_{2,n} = \{MAC_{2,1}, MAC_{2,2}, ..., MAC_{2,n}\}.$$

- w_1, w_2: The strings over O $\cup$ C [4], representing the multisets of cellular molecules and channels present in region 1 and 2 respectively.

-$w_1 = \{Bad, Bax, Trophic\_factor, phosphorylated\_Bad, Apaf\_1, procaspase\_9, procaspase\_3, apoptosome, Cytochrome\_c, caspase\_3\}$,

-$w_2 = \{Bad, Bax, bcl\_2\_bcl\_XL, MAC, Cytochrome\_c, Phosphorylated\_Bad, Inactive\_Bad\}$.

-R_{e1}, R_{e2}: The set of evolution rules associated with the three regions of the system. They are in the form u$\rightarrow$ v where u is a string over O and v is a string over O_{tar}, where $O_{tar} = O\times$ TAR [4] for TAR= 1,2. This means that every reaction indicates the output objects and the region to which the output objects will be moved. When there is no target indicator i.e. 1 or 2, this means that the outcome of the applied reaction will remain in the current region.

$-R_{e1} =$
$\{$
$r_1 = Trophic\_factor + Bad \rightarrow \langle Phosphorylated\_Bad, 2\rangle,$
$r_2 = Bad \rightarrow \langle Bad, 2\rangle,$
$r_3 = Bax \rightarrow \langle Bax, 2\rangle,$
$r_4 = Cytochrome\_c + Apaf\_1 + procaspase\_9 \rightarrow Apoptosome,$
$r_5 = Apoptosome + procaspase\_3 \rightarrow caspase\_3,$
$r_6 = caspase\_3 \rightarrow \oslash$
$\},$
$-R_{e2} =$
$\{$
$r_7 = Phosphorylated\_Bad + bcl\_2\_bcl\_XL \rightarrow Inactive\_Bad,$
$r_8 = MAC \rightarrow \langle Cytochrome\_c, 1\rangle$
$\}.$
-R_{c2}: The set of channel creation rules associated with membrane two, i.e., the outer mitochondrial membrane.

$-R_{c2} =$
$\{$
$r_9 = Bad + Bax \rightarrow \langle MAC_{2,i}\rangle$
$\}.$
where i is the number of the last formed channel.

5.3. **Experimental results.** Any experiment in the framework of P systems initially requires the predefined membrane structure, the set of rules that describe the functions of the proposed system and the multisets of data within each region of the P system. This application has the first and the second initial inputs, the membrane structure defined in figure 3 and the set of rules explained in section 5.2. We will assume some initial multisets in order to get to our aim which is observing the behavior of the channel formation rules and the results of using them. Figure 4 shows a simple diagram of the main processes of the proposed software. Notice that the right hand side of rule r_6 is $\oslash$. We used this symbol to indicate that the P system would halt because the cells died.

Table 1 shows the possible inputs to the proposed system. We focused on the *bcl_2_bcl_XL* and Bax molecules because they are the inhibitor and promoter

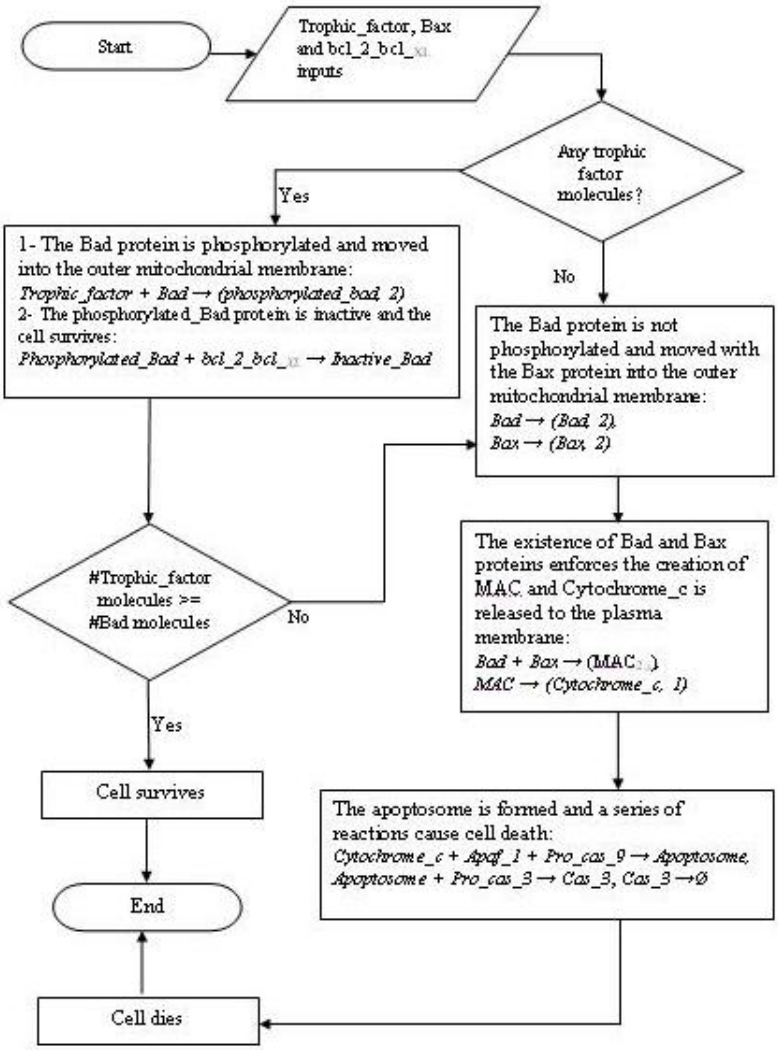


FIGURE 4. A flowdiagram of the proposed software

of apoptosis respectively. Combinations 1 and 3 in table 1 were experimented 4 times. For combination 1, experiments showed that a channel may be formed only if there is a molecule of Bad protein that can be moved from membrane one into membrane two or there are not enough $bcl\_2\_bcl\_XL$ molecules to inhibit the channel formation, otherwise the channel formation rule will not

TABLE 1. Possible inputs to the proposed P system and their outputs

Combination #	bcl_2_bcl_XL	Bax	Channel formation
1	Yes	Yes	Yes or No
2	Yes	No	No
3	No	Yes	Yes or No
4	No	No	No

be fired. For combination 2, it is impossible to create a channel in the absence of Bax protein. In combination 3, the experiments showed that it is possible to create a channel only if there are enough molecules of Bad, Apaf_1, Procaspase-9 and Procaspase-3 proteins. Finally, in combination 4 it is impossible to create a channel because of the absence of the Bad promoting protein. The following three points were observed:

1- Multiple channels can be formed within a single membrane. That is because of the condition of the maximal parallel non deterministic manner in which rules are executed.

2- A channel may be created within a membrane, but the cell still might not go through apoptosis. This means that a living cell does its best to stay alive. This feature can be applied to maintain the durability of a P system and it is already achieved in the current situation. A P system can not halt until it consumes the last chance to function.

3- In addition to membrane division and dissolving rules, channel formation rules became one type of rules that can affect the membrane itself.

In [12], two different types of channels were proposed, transport channels and diffusion channels. A diffusion channel definition is the most relevant definition to the channel formation rule. In diffusion channels, some proteins create a small hole through which molecules can freely pass [12]. The main purpose in [12] was to differentiate between the rules controlling diffusion channels and transport channels, not the process of channel creation itself. Although diffusion channels enable the passing through the membrane of an arbitrary number of objects [12], diffusion channels cannot appear more than

once [12]. On the other hand, we showed that a channel or more could be created within a membrane which is more close to biological systems, preserving the condition of the maximal parallelism of rule application. Also, in [12], a diffusion channel had to be opened and closed every time it was used, while we can create a channel once and control it by rules to constraint the passage to specific objects.

6. Conclusion and Future Work

Biological activities can be computerized using P systems. The structure and functions of a cell can be well used in different fields of computer applications. This paper is intended as a representation of a P system with channel formation rules that used the case study of the mitochondrial pathway to cell death. Results showed that one channel or more were created simultaneously in the same membrane according the current number of cellular molecules. Results also showed that a P system was able maintain its durability the same way a living cell did. The channel formation rules could result in causing the modification of the membrane structure. Instead of attempting to control the preexisting channels, promoters and inhibitors can be combined with the channel formation rules to formulate a new type of rules that creates and controls channels. Fuzzification of the rules controlling channel formation can be an extension of this paper too. The proposed software has successfully achieved the expected results and P systems proved its power. We think our work above contributes and adds to the field of membrane systems.

References

[1] A. Paun and G. Paun, *The power of communication: P systems with symport/antiport*, New Generation Computing, 2002, pp. 295-306.

[2] Diamond , McCabe , *The mitochondrion and plant programmed cell death.l*, Logan DC (Ed.), Plant mitochondria. Annual Plant Review, 2007, pp. 308-334.

[3] G. Paun,*Computing with membranes*, Journal of Computer and System Sciences, 2000, pp. 108-143.

[4] G. Paun,*Membrane computing. An introduction*, Springer-Verlag, 2002.

[5] I. Ardelean, D. Besozzi, C. Manara , *Aerobic respiration is a bio-logic circuit containing molecular logic gates.* The Fifth Workshop on Membrane Computing (WMC5), Milano, Italy, 2004.

[6] K. Saltsman, *Cell aging and death* in *Inside the cell*, National Institute of General Midical Sciences, 2005.

[7] L. Hassaan, A. Badr and I. Farag, *A P-Simulator with carriers of cellular respiration and mitochondrial oxidative metabolism*, International Journal of Computer Theory and Engineering, 2011, pp. 448-456.

[8] L. M. Dejean, S. M. Caballero, K. W. Kinnally , *Is MAC the knife that cuts cytochrome c from mitochondria during apoptosis?* in *Cell Death and Differentiation* 13 (8): 1387-95. doi:10.1038/sj.cdd.4401949. PMID 16676005. 2006.

[9] M. Cavaliere, S. Seawards, *Membrane systems with peripheral proteins: transport and evolution.* Proc.Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC06), ENTCS 171, 2007, pp. 37-53.

[10] M. Krieger, H. Lodish, A. Berk, P. Matsudaira, C. A. Kaiser, M. P. Scott, L. Zipursky, and J. Darnell, *Molecular Cell Biology, fifth edition.* 2003.

[11] Riedl and Salvesen, *The Apoptosome: signaling platform of cell death* , in Nature: Molecular Cell Biology, 2007, pp. 405-413.

[12] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo and S. Tini, *P systems with transport and diffusion membrane channels*, IOS Press, Fundamenta Informaticae XX, 2009, pp. 1-15.

[13] R. Freund, M. Oswald, *P systems with activated/prohibited membrane channels*, Proc. Workshop on Membrane Computing (WMC 2002), LNCS 2597, 2003, pp. 261-269.

[14] S. Aguzzoli, I. I. Ardelean, D. Besozzi, B. Gerla and C. Manara, *P systems under uncertainty: the case of transmembrane proteins*, First Brainstorming Workshop on Uncertainty in Membrane Computing, Palma de Mallorca, Spain, 2004, pp. 107-118.

[15] The Medical Dictionary Web Page: http://medical-dictionary.thefreedictionary.com/protein+channel.

[16] The MetaPlab Virtual Laboratory. Retrieved on October 1st, 2010 from http://mplab.scienze.univr.it/index.html.

[17] The P System Webpage. http://ppage.psystems.eu.

[18] University of Sheffield. Retrieved on October 1st, 2010 from http://staffwww.dcs.shef.ac.uk/people/M.Gheorghe/PSimulatorWeb/Tools.htm.

[19] University of Trento Centre for Computational and Systems Biology. Retrieved on October 14, 2010 from http://www.cosbi.eu/Rpty_Soft.php.

[1]Department of Computer Science, Modern Academy in Maadi, Road 304, New Maadi, Cairo, Egypt.

E-mail address: lamia_work@yahoo.com

[2]Department of Computer Science, Faculty of Computers and Information, Cairo University, 5 Dr. Ahmed Zewail Street, Postal Code: 12613, Orman, Giza,Egypt

E-mail address: a.badr.fci@gmail.com, i.farag@fci-cu.edu.eg

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 4, 2011

ON USING GENERICS FOR IMPLEMENTING ALGEBRAIC STRUCTURES

VIRGINIA NICULESCU

Abstract. Object oriented programming and design patterns form a very good framework for implementing a computational algebra system. Object oriented programming offers different kinds of mechanisms for obtaining high level of genericity; and these are also mechanisms for reusing.

The need to represent and work with abstract algebraic structures implies the need for genericity. This could be achieved by using parametric polymorphism or type polymorphism. We analyze in this article the suitability of using parametric polymorphism, in an object-oriented programming context, for implementation of algebraic structures, showing some advantages but also some problems. The implementation issues are discussed for the particular case of the square matrices of a certain order.

The analysis emphasizes also general advantages of using parametric types – such as type safety, but also some constrains forced by them.

1. Introduction

In modern programming, genericity and reusing are very important and much discussed issues, and so there are different mechanisms to achieve them. In object oriented programming we may distinct two main types of such mechanisms: one based on inheritance of type polymorphism (or polymorphic types), specific to pure object oriented languages, and another based on parametric types [1].

Computer algebra provides a compelling application of parametric polymorphism, where various algebraic constructions, such as polynomials, series, matrices, vector spaces, etc, are used over various different coefficient structures, which are typically rings or fields [2]. The work initiated by Jenks and

Received by the editors: March 25, 2011.

2010 *Mathematics Subject Classification.* 08A99.

1998 *CR Categories and Descriptors.* D.1.5 [**Software**]: Programming Techniques – *Object Oriented Programming*; I.1.4 [**Computing Methodologies**]: Symbolic and Algebric Manipulation – *Applications*.

Key words and phrases. genericity, OOP, templates, patterns, algebraic structures, abstractness.

Trager [11] led to the formulation of domain and category constructors in Axiom [24, 10], and higher-order domain and category producing functions in specialized language Aldor [3, 4].

Parametric genericity, initially represented in object oriented setting by source code reuse mechanism as C++ templates, became more and more popular and other object oriented languages as Java and C# enhanced their new versions with mechanisms that offer parametric types.

In C++ the template mechanism allows us not to create a single class, but to specify only once the pattern for creation of some classes that are different only by the type of some parameters. The template mechanism allows a very high degree of flexibility, but it is considered in some literature not a really parametric polymorphism mechanism since for each actual parameter a new class is created

Both the NTL library for number theory [22] and the Linbox library for symbolic linear algebra [6] use C++ templates to achieve genericity.

The mechanism which was included in Java since JDK 1.5 is considered more efficient since just one class is created for each parameterized class. Also, the mechanism of parameterized Java classes allows bounded polymorphism – the specification of a certain behavior of parameters by interface implementation.
A similar mechanism is implemented in C# too.

A comparison between C++ templates and the extensions for generics of the C# and Java languages based on their suitability for scientific computing was done in [8]. These measurements suggest that both Java(TM) and C# generics introduce only little run time overhead when compared with non-parameterized implementations. With respect to scientific application, C# generics have the advantage of allowing value types (including builtin types) as parameters of generic classes and methods. Also, in [5] there is study about the performance of generics for scientific computing in various programming languages, based on a standard numeric benchmarks. The conclusion was that in current implementations of generics must be improved before they are used for efficiency-critical scientific applications.

Design patterns [7] are also very important when we need genericity and reusing. They already showed there power, and for the case of a computer algebra system they bring important advantages. Creational design pattern are particularly important in this context, and the analysis in this paper is focused on the suitability of using them in the specified languages, too.

An efficient design for a computer algebra system with certain advantages has been presented in our previous papers [17, 18, 19]. This approach allows

working not only with concrete algebraic structures, but also with abstract algebraic structures. The advantages of this approach result from the usage of:

- *creational design patterns* – which allow us to build not only a flexible numerical algebraic system, but also a general abstract algebraic system (these bring the most important advantages of implementing abstract algebraic structures);
- *reflection and dynamic loading* – which allow *automatic conversions* between compatible structures, and also allow dynamic creation of new classes that correspond to different algebras specified by the user;
- *representation independence* – which allows us to operate with algebraic elements independently of how their component values are represented.

Implementation of this previous approach was based on type polymorphism. Polymorphic types based on inheritance offer several advantages (a greater flexibility, efficiency, conversions), but they have also important disadvantages: they cannot assure the homogeneity, and the possible errors are identified and eventually treated only at the run time. Because of this it has been advocated that the parametric polymorphism is more appropriate for the implementation of the algebraic structures.

The Java Algebra System (JAS)[14, 13, 12] provides a software library using generic types for algebraic computations implemented in the Java programming language. Our idea of using the creational design patterns has been used, but in the context of the Java generic types. The project emphasized nice advantages of using Java (object-orientation, type safety, multi-threaded), but also some inconveniences of using Java generics for a CAS related to type erasure, dependent types, and performance.

2. General requirements for the algebraic structures implementation

The main requirement is the possibility of working with abstract algebraic structures like groups, rings, fields, etc. The user has to be allowed to define concrete algebraic structures by using these abstractions.

This implies that we have to define abstract classes for elements of each abstract algebraic structure.

New abstract algebraic structures may be built over other algebraic structures; for example polynomials and matrices. Polynomials are built over a unitary commutative ring, and they also form a unitary commutative ring.

We may have polynomials or matrices that have different coefficient types: real, complex, etc. Also, a polynomial or a matrix can be built over any kind of algebraic unitary commutative ring $(R, +, *)$, and they also form rings.

The *Composite* design pattern may be used to implement these kinds of structures. Using the *Composite* pattern we may define polynomials over other polynomials. Similar examples may be given for matrices – we can define matrices over polynomials, or matrices over matrices. So, we may achieve the flexibility imposed by our goal.

We will analyze in what it follows the possible solutions for general algebraic structures implementation by considering two approaches: one based on the classic C++ templates, and one based on generic classes as they are implemented in Java.

First obvious conclusion is that the conversions will not be allowed any more, but very often the mathematicians' opinion was that it is enough if we are able to operate only inside one algebraic structure.

2.1. **Basic design.** The basic algebraic structures are algebras with one or two binary operations such as groups, rings, and fields.

Following the definition of the basic algebraic structures we have interfaces that correspond to semigroups, monoids, groups, rings, unitary rings, division rings, and fields. These interfaces characterize the elements of the algebraic structures and their behavior. So, we have interfaces such as `GroupElem`, `RingElem`, etc.

The operations defined by these interfaces such as `plus()`, `minus()`, `opposite()`, create new elements.

For concrete groups, rings, etc. concrete classes are built by implementing the corresponding interfaces.

An algebraic structure is defined by the domain of its elements and by a set of operations. The operations could have different properties (i.e. commutativity, neutral element, etc). These properties are characteristics of the entire algebraic structure; they do not characterize each element of the domain. So, these properties are specified into separate classes for which we normally have to apply the *Singleton* pattern [7]; only one single instance would be not only enough, but also necessary in order to assure a safe representation. These classes can also be viewed as *Abstract Factories* [7], since they are able to create *products* which are the neutral elements for the algebraic operations.

A factory class associated to a ring structure will define operations such as:

```
-: isCommutative()
-: isUnitary()
```

```
-: createZero();
-: createOne()
```

An important issue is represented by the way in which is implemented the connection between the instances of the class that represents the algebraic elements, and the class that characterized the entire algebraic structure and which is also the factory of the neutral elements.

If the *Abstract Factory* and *Singleton* design patterns are used, then the connection between the elements and the singleton instance of their algebraic structure is done through the single global access point assured by the *Singleton* pattern. An elegant solution to implement this is to define the factories as static nested classes within the classes that define the algebraic elements.

If static definitions are not available, then a simple association relation between each element instance and a factory instance have to be implemented. In this case, problems may arise since we may have more factory instances for the same algebraic structure. We have to verify the correct association between each element and the corresponding factory, too.

Since we will focus more on implementation decisions we will restrain the analysis just to the case of square matrices implementation over unitary rings. So, we will consider the interface `RingElem` and the associated factory. The class `MatrixElem` implements also the interface `RingElem` since the set of all matrices over a ring forms also a ring. So, we have to be able to create matrices of matrices, etc.

3. C++ templates

In C++ we have the possibility to overload the operators and so we will use the classic operators for defining addition, difference, multiplication, etc. The interface `RingElem` can be specified based on operators.

```
template<typename T>
class RingElem{
public:
...
    virtual T operator+( const T&) = 0;
    virtual T operator-(const T&) = 0;
    virtual T operator*(const T&) = 0;
    virtual T& operator-() = 0;
////////////////////////////////
    class Factory{
        public:
```

```
        virtual bool isCommutative()=0;
        virtual bool isUnitary()=0;
        virtual T createZero()=0;
        virtual T createOne()=0;
    };
////////////////////////////////
    virtual Factory* getFactory()=0;
};
```

The instance method `getFactory()` has to be defined in order to make the factory object accessible through each created element of the structure. This is particular important when the global access point to the factory object could not be implemented.

The parameter type T can be replaced by any type which may be considered as a unitary ring. An example is the class `Complex` for which we give a partial implementation.

```
class Complex:  public RingElem<Complex>{
    double re, im;
public:
    ...
    //the implementation of the operators
    ...
    class Factory:  public RingElem<Complex>::Factory{
        Factory(){}
        static Factory* instance;
    public:
        Complex createZero(){
            return Complex(0,0);
        }
        ...
        static Factory* getInstance(){
            if (instance==0) instance = new Factory();
            return instance;
        }
    };
    RingElem<Complex>::Factory* getFactory(){
        return Factory::getInstance();
    }
```

```
    static RingElem<Complex>::Factory* getFactoryStatic(){
        return Factory::getInstance();
    }
};
```

We may emphasize the fact that we were able to use the *Singleton* design pattern, and also to implement a static method `getFactoryStatic()` that returns the unique factory instance.

The class `MatrixElem` is built over a ring, so the parameter `T` has to implement the interface `RingElem`. But this constrain cannot be specified for a C++ template. This is a clear disadvantage.

```
template <typename T>
class MatrixElem:  public RingElem<MatrixElem>{
    Storage<T> mat;
    int n;
    RingElem<T>::Factory* factoryE;
public:
    ...
    MatrixElem (int n, RingElem<T>::Factory* f){
    //matrix Zero
        factoryE=f;
        ...
        for(int i=0;i<n;i++)
            for(int j=0;j<n;j++)
                mat.add(i,j, factoryE->createZero());
        this->n=n;
    }

    class Factory:  public RingElem<MatrixElem>::Factory{
        int n;
        RingElem<T>::Factory* f;
        Factory(int n, RingElem<T>::Factory* f){
            this->n = n;
            this->f = f;
        }
        static vector<Factory*> instance;
    public:
        ...
        MatrixElem createZero(){return MatrixElem(n, f);}

        static Factory* getInstance(int n, RingElem<T>::Factory* f){
            for(int i=0;i<instance.size();i++)
                if ((instance)[i]->n==n)return instance[i];
            Factory * fnew=new Factory(n,f);
            instance.push_back(fnew);
```

```
            return fnew;
        }
    };
    RingElem<MatrixElem>::Factory* getFactory(){
        return Factory::getInstance(n,factoryE);
    }
};
```

A general storage is used for the elements because we want to preserve the advantage of representation independence. The class `MatrixElem` has to work with a factory (`factoryE` for managing the type of its elementary elements of type `T`). This factory is used, for example, when the matrix *zero* and the matrix *one* are created.

Also, the `MatrixElem` is itself a `RingElem`, so it has to be related to a factory class corresponding to this new structure of ring type.

A matrix algebraic structure is characterized by the order of the matrix, too. In the proposed implementation we have only one class for all the matrix structures with elements of the same type. So, the associated factory has to create the *zero* and the *one* matrices with the order given as parameter. The factory class associated to the template class `MatrixElem` uses a generalization of the *Singleton* design pattern, and stores a vector of instances, one for each order of the matrices in use.

Each matrix has to be associated to a ring factory. Because we could use *Singleton* design pattern for the factory implementation, the creation of the factory objects is very well controlled; we will have only one factory for each algebraic structure (in our case one for each order and element type). But since we have a different factory for a different order we cannot implement a static method `getFactoryStatic()` – as we did for complex numbers.

All the operations are correctly defined if they operate only on elements from the same algebraic structure – matrices of the same order and with elements of the same type. The template parameter imposes the same type for the elements type but the order is not constrained. Exceptions have to be thrown when the operations are called with wrong parameters.

In order to solve this problem related to dependent types [21], a better solution defines the matrix order as a template parameter, too. This leads to the creation of one class for any particular algebraic structure. From the programming point of view this is not very efficient (a bunch of classes will be created), but assures a high degree of rigor from the mathematical point

of view, and also a static verification of the operands.

```
template <typename T, int order>
class MatrixOrdElem:  public RingElem<MatrixOrdElem>{
    Storage<T> mat;
    static RingElem<T>::Factory* factoryE;
public:
    MatrixOrdElem (){// matrix Zero
        factoryE=T::getFactoryStatic();
        for(int i=0; i<order;i++)
            for (int j=0; j<order;i++)
                mat.add(i,j,factoryE->createZero());
    }

    class Factory:  public RingElem<MatrixOrdElem>::Factory{
        Factory(){ }
        static Factory* instance;
    public:
        MatrixOrdElem createZero(){return MatrixOrdElem();}
        ...
        static Factory* getInstance(){
            if (instance==0) instance = new Factory();
            return instance;
        }
    };
    static RingElem<MatrixOrdElem>::Factory* getFactoryStatic(){
        return Factory::getInstance();
    }
};
```

With this variant we eliminate the necessity to give the associated factory instance (for the type T) as a constructor argument, and so, the necessity to create/access a factory instance each time a new matrix instance is created. The presented solution uses a static method `getFactoryStatic` that returns the singleton instance. This method is called inside the constructors of the class `MatrixOrdElem`.

For creating factories, this solution is safe because the responsibility of correct initialization of the factory is not let to the user. But this brings some other disadvantages: C++ predefined types (int, double, etc.) could not be used anymore as values of the parameter `T`. The possibility of using predefined type as a template parameter is one of the nicest characteristic of genericity based on templates. One solution to preserve this possibility (different from that with factory parameter of the constructor) is to set the static member

`factoryE` only from outside (the user has to do this setting properly before instantiation – which is an important disadvantage). Also, corresponding factories for the predefined types have to be defined.

```
class IntFactory: public RingElem<int>::Factory{
    IntFactory(){}
    static IntFactory* instance;
public:
    int createZero(){return 0;}
    static IntFactory* getInstance(){
                if (instance==0) instance = new IntFactory();
        return instance;
    }
    ...
};
```

4. Java Generics Approach

Using Java generics is similar in many aspects to C++ templates, but there are important differences imposed by the different mechanism for achieving parametric polymorphism.

In this case the implementation of the `RingElem` interface uses recursive bounded polymorphism –which is a clear advantage of this solution. Factory interfaces could also be defined as nested interface.

```
public interface RingElem <T extends RingElem<T>> {
    public interface Factory <T >{
        T createZero();
        ...
    }
    public T plus(T e );
    public boolean isZero();
    ...
    public Factory<T> getFactory();
}
```

For a simple implementation of a concrete type (such as `Int` for integer numbers, or `Complex` for complex numbers) the implementation of the factory could be based on *Singleton* pattern and on static members, in a similar way

as for C++ templates. The predefined types are used based on autoboxing properties.

The class `MatrixElem` is defined as a generic class with a type parameter bounded to `RingElem<T>`. All the matrices of a certain order and with a certain type of its elements type also form a ring, and so the class `MatrixElem` implements the interface `RingElem<MatrixElem<T>>`.

```
public class MatrixElem <T extends RingElem<T>> implements RingElem<MatrixElem<T>>{
    private Storage<T> a;
    private int n;
    private RingElem.Factory<T> factory;

    public MatrixElem(int n,RingElem.Factory<T> f){
    //create a zero matrix
        this.n=n;
        factory =f;
        a=...
        for(int i=0;i<n;i++){
            for(int j=0;j<n;j++)
                a.add(i,j, factory.createZero());
    }
    ...
    public MatrixElem<T> plus(MatrixElem<T> x) throws Exception{
        if (n!=x.getOrder())
            throw new Exception("illegal arguments");
        ...
    }

    public Ring.Factory<T> getFactory(){
        return new MatrixFactory<T>(n,factory);
    }
}
```

If we intend to follow the design based on which factories classes are defined inside elements classes, then for this kind of objects the factory is defined as an inner class. Since we work with generics we cannot define static members and so the nested class cannot be static. If we let the factory class to be an inner class, then each factory instance of type `MatrixElem<T>.Factory` is connected as a member with an instance of the outer class `MatrixElem`. This could lead to the creation of one factory instance for each matrix element – which is abnormal. Because of this, the general design decision to define the associated factory class as nested class has to be eliminated, and we have to

define the factory class outside the class `MatrixElem`. In any cases, the *Singleton* pattern cannot be used.

```
class MatrixFactory<T extends RingElem<T>> implements RingElem.Factory<MatrixElem<T>>{
    private RingElem.Factory<T> factory;
    private int n;
    public MatrixFactory(int n,RingElem.Factory<T> f){
        this.n=n;
        factory = f;
    }
    public MatrixElem<T> createZero(){
        return new MatrixElem<T>(n,f);
    }
    ...
}
```

The connection between elements and factories is done through the constructor arguments. (This solution has been also chosen in JAS.) In order to obtain a factory for elements which are matrices of a certain order and a certain elements type, we may use directly the factory constructor or the instance method of the `MatrixElem` class `getFactory()`. This solution is not a very safe solution since the creation of the factory instances is not controlled. Also, the responsibility for correct association between the elements and factories is let to the user and could become very complicated when matrices of matrices are created.

Another solution to assure an implicitly correct done association is to use reflection, but there are several drawbacks too. Implicit constructors (with no arguments) have to be defined for all elements, and also the problem related to dependent types (different structures for different orders of matrices) cannot be solved efficiently. The following code shows a possible definition based on reflection of a constructor for `MatrixElem`.

```
public MatrixElem( Class<T> c, int n){
    try{
        this.n=n;
        java.lang.reflect.Method meth=null;
        Object retobj=null;
        try{
            meth = c.getMethod("getFactoryStatic");
            retobj = meth.invoke(null);
        }
```

```
        catch(java.lang.NoSuchMethodException e){
            try{
                meth = c.getMethod("getFactory");
                retobj = meth.invoke(c.newInstance());
            }
            catch(java.lang.NoSuchMethodException e1){
                ...
            }
        }

    factory = (Ring.Factory<T>)retobj;

    ...
}
```

First a possible invocation of the method `getFactoryStatic()` is done; if this fails then the instance method `getFactory()` is invoked. This means that for the classes for which we may define the static method `getFactoryStatic()`, this method will be used, and for the others the instance method `getFactory()` will be used.

An example of using such a constructor is:

```
Matrix<Int> a = new MatrixElem<Int>(Int.class ,2);
```

It can be noticed that no factory instance creation is necessary.

5. Conclusions

The problem of implementing a general matrix over any kind of commutative and unitary ring emphasizes the fact that we need more than specifying a parameter with a certain behavior; we need to refer and work with some specific instances from the domain of the type parameter before knowing their exact concrete type.

The main emphasized problem related to the C++ templates was the fact that we cannot specify explicitly any restriction related to the parameters, to bound them. A step forward in this direction has been done for generic Java classes, where for a parameter we can specify that it implements a certain interface. But we may ask if restraining the behavior is powerful enough. The C++ templates mechanism is considered for implementing parametric polymorphism based on an "heterogeneous" approach. The "heterogeneous"

approach constructs a special class for each different use of the type parameters. The compiled code is fast, but the object code could become bulky since we have many different versions of each class.

Still, the analysis emphasized the fact that we have certain advantages using the old C++ templates over the new Java or C# generics, for this case. Java generics implement "homogenous" approach of the parametric polymorphism. Since is based on "type erasing" we have strong restrictions, and the most important one, for the analyzed case, was the impossibility of specifying static members for the generics, and so the impossibility of specifying properties related to an entire algebraic structure.

For creation and usage of the neutral elements at the abstraction level imposed by the generic type definition, there are solutions based on the creational design patterns: *Factory Method*, *Prototype*, or *Abstract Factory*. *Abstract Factory* is the best solution since all these special values can be considered products from the same family. *Abstract Factory* is usually used together with *Singleton* pattern that assures a global access point the factory instance. Still, we have to define a concrete factory for each concrete type in order to be used as an actual parameter.

Some problems could be observed also from the Java implementation of JAS library. For example, for the simple coefficients (elements) rings such as integer or rational numbers, there are classes that implement the both interfaces: one defined for the elements of the structure, and the second defined for the associated factory. These kinds of solutions may locally simplify things but they are not based on appropriate object-oriented design decisions. Factories have to be related to the type (class) and not to each element of an algebraic structure.

References

[1] Cardelli, L., Wegner, P. *On understanding types, data abstraction, and polymorphism* ACM COMPUTING SURVEYS, (1985).

[2] Chicha, Y.; Lloyd, M. Oancea C.; Watt, S.M. *Parametric Polymorphism for Computer Algebra Software Components.*

[3] Chicha,Y.; Defaix, F. ; Watt, S. TR537 - *The Aldor/C++ Interface: User's Guide.* Technical report, Computer Science Department - The University of Western Ontario, 1999.

[4] Chicha,Y.; Defaix, F. ; Watt, S. TR538 - The Aldor/C++ Interface: Technical Reference. Technical report, Computer Science Department - The University of Western Ontario, 1999.

[5] Dragan, L.; Watt, S.M. *Performance Analysis of Generics in Scientific Computing*, Proceedings of Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05), 2005, pp.93-100.

[6] J. G. Dumas, T. Gautier, M. Giesbrecht, P. Giorgi, B. Hovinen, E. Kaltofen, B. D. Saunders, W. J. Turner, and G. Villard. LinBox: *A Generic Library for Exact Linear Algebra*. In Proc. of ICMS, pages 40-50. A. Miola ed. Academic Press, 2002.

[7] Gamma, E., Helm, R., Johnson, R., Vlissides, J. *Design Patterns: Elements of Reusable Object Oriented Software*, Addison-Wesley, 1995.

[8] Gerlach, J.; Kneis, J. *Generic programming for scientific computing in C++, Java, and C#*. Lecture Notes in Computer Science. Proceedings of International Workshop on Advanced Parallel Processing Technologies (APPT) ¡5, 2003, Xiamen¿, pp.301-310

[9] Gilbert, J., *Elements of Modern Algebra*, PWS-Kent, Boston, 1992.

[10] Jenks, R. D.; Sutor, R. S. *AXIOM, The Scientific Computation System*. Springer-Verlag, 1992.

[11] R. D. Jenks and B. M. Trager. *A Language for Computational Algebra*. In Proc. SYMSAC, pages 6-13. ACM, 1981.

[12] Kredel, H.; Jolly, R. *Generic, Type-Safe and Object Oriented Computer Algebra Software* Computer Algebra in Scientific Computing, Lecture Notes in Computer Science, 2010, Volume 6244/2010, 162-177,

[13] Kredel, H., *Evaluation of a Java Computer Algebra System*, in Lecture Notes in Computer Science, volume 5081/2008, pp. 121-138, Springer Berlin / Heidelberg.

[14] Kredel, H.: *On the Design of a Java Computer Algebra System*. In Gitzel, R., Aleksy, M., Schader, M., Krintz, C., eds.: Proc. PPPJ 2006. ACM International Conference Proceedings Series, Mannheim University Press (2006) pp. 143-152.

[15] Lujn, M., Freeman, T. L., Gurd, J. R., *OoLaLa: an Object Oriented Analysis and Design of Numerical Linear Algebra*, In the Proceedings of Conference on Object-Oriented Programming, Systems, Languages, and Applications – OOPSLA 2000.

[16] Musser, D.R., Scine A., *STL Tutorial and Reference Guide: C++ Programming with Standard Template Library*, Addison-Wesley, 1995.

[17] Niculescu, V.,*A Design Proposal for an Object Oriented Algebraic Library*, Studia Univ. Babes-Bolyai, Informatica, Volume XLVIII, No. 1, 2003, pg. 89-100.

[18] Niculescu, V., Moldovan, G.S. *OOLACA: an object oriented library for abstract and computational algebra*, Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications OOPSLA 2004, Vancouver, BC, CANADA, ACM Press New York, NY, USA, pp. 160-162.

[19] Niculescu, V., Moldovan, G.S. *Building an Object Oriented Computational Algebra System Based on Design Patterns*. Proceedings of International Symposium on Symbolic and Numeric Algorithms for Scientific Computing SYNASC'05, Timisoara, IEEE Computer Society Press, Romania, Sept. 2005, pp. 101-108.

[20] Niculescu, V.,*Teaching about Creational Design Patterns – General Implementations of an Algebraic Structure*, In the Proceedings of ECOOP 2003 Workshop on Pedagogies and Tools for Learning OOP, Darmstadt, Germany.

[21] Poll, E.; Thomson, S.: *The type system of Aldor*. Technical report, Computing Science Institute Nijmegen (1999)

[22] V. Shoup. *NTL: A Library for doing Number Theory*, 2004, http://www.shoup.net/ntl/doc/tour.html.

[23] B. Stroustrup, M. Ellis, *The Annotated Reference C++ Manual*, Addison-Wesley, 1994.

[24] Watt,S. M.; Jenks,R. D. ; Sutor,R. S.; Trager, B. M.. *The Scratchpad II Type System: Domains and Subdomains*. In Proc. of Computing Tools For Scientific Problem Solving, pages 63-82. A. Miola ed. Academic Press, 1990.

[25] *Generic Java*,http://download.oracle.com/javase/1.5.0/docs/guide/language/generics.html

Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca
E-mail address: vniculescu@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 4, 2011

GOD CLASS DESIGN FLAW DETECTION IN OBJECT ORIENTED DESIGN. A CASE STUDY

CAMELIA ŞERBAN

Abstract. In this article we present an experimental evaluation of our proposed methodology for object-oriented design (OOD) assessment introduced in previous work. A comparison with a related approach based on detection strategies is comprised in the proposed case study. It also discusses the limitations of existing approaches found in literature in the context of OOD assessment.

1. Introduction

Object-oriented systems that undergo repeated addition of functionality commonly suffer a loss of quality in their underlying design. A small change in one part of it may have unforeseen effects in completely other parts, leading to potential disasters. In order to prevent this, we aim to maintain a high quality design throughout the system lifecycle. This goal can be attained by repeated evaluation of the system design, with the goal of early identification of those design entities that are not conformable to rules, principles and practices of a good design [2].

As a result in this direction software metrics have been brought to the attention of many researchers. They measure different aspects of software and therefore play an important role in *understandig, controlling and improving* software quality.

Several approaches [2, 4, 5, 6] were found in literature that address the problem of metrics based assessment for OOD. However, these approaches encounter some limitations: i) how to set proper threshold values for metrics is not addressed; ii) they lack a standard terminology and formalism in order to define the contextual background which can also serve for metrics definition. To mitigate these limitations, a *Conceptual Framework for OOD Assessment (CFDA)* has been introduced in our previous work [1]. This paper presents

Received by the editors: October 10, 2011.

2000 *Mathematics Subject Classification.* 68N30, 68T37.

1998 *CR Categories and Descriptors.* code D.2.8 [**Software Enginnering**]: *Metrics – Product metrics*; code D.1.5 [**Pattern recognition**]: *Clustering –Algorithms.*

Key words and phrases. Software metrics, object oriented design, fuzzy clustering.

an experimental evaluation of this methodology for object-oriented design assessment.

The paper is organized as follows: Section 2 briefly describes our previous work. The proposed experimental evaluation is presented in Section 3. To emphasize the advantages of the CFDA, Section 4 presents a comparison with a related approach based on detection strategies [2]. Finally, Section 5 summarizes the contributions of this work and outlines directions for further research.

2. A conceptual framework for OOD Assessment

In our previous work [1] we have proposed a quantitative evaluation methodology for object-oriented design. The proposed methodology, based on static analysis of the source code, is described by a conceptual framework which has four layers of abstraction:

The first layer, *Object-Oriented Design Meta-Model*, formally defines *the domain of the assessment* $D(S) = (E, Prop(E), Rel(E))$ of an OOD corresponding to a software system S; this meta-model describes the design entities that are evaluated E, their properties $Prop(E)$ and the relationships between them $Rel(E)$.

The second layer, *Formal Definitions of OOD Metrics*, consists of a library of OOD metrics definitions. Metrics are formally defined using the context delineated for the meta-model presented by first layer and expressing them in terms of algebraic sets and relations, knowledge assumed as familiar since the first stages of our studies.

The third layer, *Specifications of the Assessment Objectives*, specifies in a formal manner the assessment objectives using a metrics based approach.

The last layer, *Mesurement Results Analysis*, uses the fuzzy clustering analysis method to interpret the measurement results obtained in the assessment process. This method overcome the limitations of the existing approaches which use threshold values for metrics. The selected algorithm, Fuzzy Divisive Hierarchic Clustering (FDHC) [3] produces a binary tree hierarchy that provides an in-depth analysis of the data set, by deciding on the optimal subcluster cardinality and the optimal cluster substructure of the data set.

3. Proposed Case-Study

In this section, we describe the steps needed to be performed to apply the proposed evaluation framework on an open source application, namely *log4net* [7]. It consists of 214 classes grouped in 10 packages.

Domain Assessment Identification. We parse the source code of *log4net* application and produce the domain of the assessment, $D(log4net) =$

$(E, Prop(E), Rel(E))$, i.e the design entities, their properties and the relations between them.

Setting the Assessment Objectives. The objective of the proposed assessment is to identify those classes AE (assessed design entities) from the *log4net* application affected by "God Class" [8] design flaw. In order to attain this goal we proceed as follows: *i*) a set of design principles, heuristics or rules DP are related to "God Class" design flaw, defining a flaws–principles graph $FPG = (DF, DP, G_{DF \mapsto DP} \subseteq DF \times DP)$, $DF = \{God\ Class\}$; *ii*) the elements of DP are than related to a set of software metrics M which quantify these principles, obtaining the principles–metrics graph $PMG = (DP, M, G_{DP \mapsto M} \subseteq DP \times M)$.

Therefore, the 3-tuple $AO(log4net) = (AE, FPG, PMG)$ represents *the assessment objectives specification* regarding the evaluation of the object oriented design $D(log4net) = (E, Prop(E), Rel(E))$ corresponding to *log4net* application.

An instance of a God Class performs most of the operations, delegates only minor details to a set of trivial classes, and uses the data from other classes. This design flaw violates the principle of *manageable complexity*, as god classes tend to capture more than one abstraction. Consequently, such pathological classes tend to be also *non-cohesive*.

The selected heuristics [8] related to God Class design flaw are: *i*) distribute system intelligence horizontally as uniformly as possible (it refers to *class complexity*); *ii*) beware of classes with much non-communicative behavior (it refers to intraclass communication – *low cohesion* principle); *iii*) beware of classes that access directly data from other classes. Thus, the selected metrics to quantify these heuristics are: Weighted Method per Class (WMC) [9], Tight Class Cohesion (TCC) [10], Access to Foreign Data (ATFD) [2].

To add more clarity for the above mentioned statements, Figure 1 presents the specification of the assessment objectives.

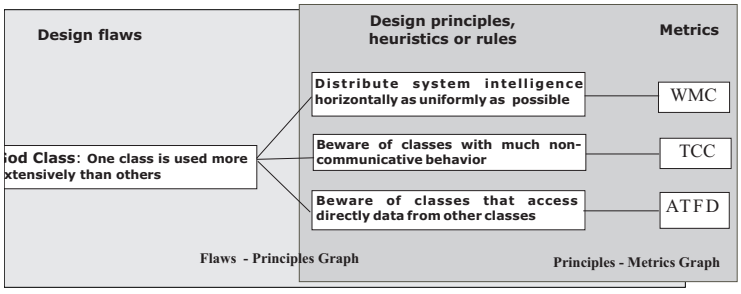


FIGURE 1. Specification of the assessment objectives.

Formal Definition of Selected Metrics. Due to space restriction, we present here only the formal definition of ATFD [2] metric. ATFD represents the number of external classes from which a given class accesses attributes, directly or via accessor-methods. Its formal definition being:

$$ATFD(c) = card(A - B - C), c \in Class(E) \ where :$$

$A = \{d \in Class(E) | \exists m_0 \in Meth(c) : ARefRel(m_0) \cap Attr(d) \neq \emptyset\}$,
$B = \{d \in Class(E) | \exists c_{i_0}, c_{i_1}, ..., c_{i_{l_i}} \in Class(E) : c_{i_0} = c, c_{i_{l_i}} = d, (c_{i_j}, c_{i_{j+1}}) \in IRel_{c_{i_j}}, j = \overline{0, l_i - 1}\}$,
$C = \{d \in Class(E) | d \in Inner(c))\}$.

Measurement Results Analysis. The results of the assessment are a set of design entities that were evaluated AE, in this case the set of classes from *log4net* application, together with their corresponding values of the selected metrics WMC, TCC, $ATFD$. Based on the metrics values, we will select from AE the "*suspect*" entities (those classess affected by "God Class" design flaw).

Following a classical approach we have to set thresholds values for metrics that we use. In order to overcome this limitation, we propose an alternative approach, based on fuzzy clustering analysis. Thus, an entity may be placed in more that one group, having different membership degrees.

The optimal fuzzy partition, $U = \{1.1.1, 1.1.2, 1.2.1, 1.2.2, 2.1, 2.2\}$ obtained by applying the FDHC [3] algorithm contains six clusters. From each of them we have eliminated some entities whose metrics values are very different from the majority of entities of this cluster. We name these entities "isolated data points", they are considered for further analysis.

We consider that clusters with "suspect" entities have to simultaneously meet the following two conditions: *i*) the member's average values for the WMC and ATFD metrics are greater than the average values of the entire set of analysed design entities; *ii*) the member's average value for the TCC metric is lower than the average value computed on the entire set of analysed design entities.

Taking into account the above mentioned criteria, clusters 1.1.1, 2.1, 2.2 have been identified as containing possible *suspect* entities.

Regarding the set of isolated points we can conclude the following: *i*) The entities with id 16, 26, 44, 158, 175, 180 are not considered *suspect* entities. They have low values for the WMC and ATFD metrics and they are cohesive. *ii*) One entity with class id 123 is complex but cohesive and with low values of ATFD metric, thus is not considered *suspect* entity. *iii*) The classes with the id 33, 36, 39, 42, 52, 73, 170 are considered *suspect* entities, some of them being cohesive but very complex and with high values for ATFD and some being less cohesive and lacking complexity but also having high values for ATFD.

4. A Comparative Analysis

Marinescu [2] defined metric-based detection strategies for capturing design flaws of object-oriented design. In the definition of detection strategy he used threshold values for metrics. The current section makes a comparison of our approach based on fuzzy analysis with that proposed by Marinescu regarding the identification of "God Class" design flaw. The detection strategy defined by Marinescu is presented in what follows.
$GodClasses := ((ATFD, TopValues(20\%)) \wedge (ATFD, HigherThan(4))) \wedge ((WMC, HigherThan(20)) \vee (TCC, LowerThan(0.33))$.

We have applied the above mentioned detection strategy on the same data set of measurement results. In this respect, we will make a comparison beetwen the two sets of *suspect* entities: the list of entities identified as *suspect* using the approach proposed by us, and the list of entities identified as *suspect* using the approach proposed by Marinescu. The conclusions are presented in what follows.

From the 30 entities which are identified as suspects by the approach of Marinescu, only two are not also identified using our approach, entities 94 and 151. These entities are classes which are non-cohesive and access external data but have a relative low complexity in comparison to the other suspect entities.

From the 46 entities identified by our approach, 18 were not identified using the detection strategy based approach. The reason why these were not considered as suspects by the approach of Marinescu is because of their low ATFD value (lower than 4). Let us analyse two entities identified as suspects by our approach: One with classId=13 and metric values WMC=19, TCC=0.07 and ATFD=4 and the other with classId=3 and metric values WMC=44, TCC=0.07 and ATFD=3. The first one is considered suspect by the detection strategy based approach and the second one is not. We ask ourselves if the difference of 1 on the ATFD metric is relevant when deciding between the two, which one is suspect and which one isn't. We do observe that the two entities have the same value for TCC but the WMC value is much higher for the second entity. Also, the membership degree of the second entity to the cluster of suspect entities is much higher than the one of the first entity.

All the above mentioned statements argue that our approach overcome the limitations which the approaches based on the establishement of thresholds values for metrics have encountered. We recall here these limitations: they are *inflexible*, the selected entities have metrics values that belong to a strictly defined interval, and they are *opaque*, providing information only about the suspect entities and they do not provide an overview of all entities in the

system, thus we do not know if a suspect entity is not very similar to many more other entities who have not passed the threshold values.

5. Conclusion and Future Work

We have presented in this paper a case-study to experimentally validate our proposed methodology for object-oriented design assessment [1]. The case-study addressed the issue of "God Class" design flaws detection. The approach is based on metrics and on fuzzy clustering analysis method. To highlight the advantages of using our model, a comparative study with a similar approach which uses threshold values for metrics has been made.

As one of our further work we aim to propose other comparisons with similar approaches regarding OOD assessment based on metrics.

References

[1] Camelia Serban, *A Conceptual Framework for Object-oriented Design Assessment*, Computer Modeling and Simulation, UKSim Fourth European Modelling Symposium on Computer Modelling and Simulation, 90–95, 2010.

[2] R. Marinescu, *Measurement and quality in object-oriented design*, Ph.D. thesis in the Faculty of Automatics and Computer Science of the Politehnica University of Timisoara, 2003.

[3] Dumitrescu, D., *Hierarchical pattern classification*, Fuzzy Sets and Systems 28, 145–162, 1988.

[4] S. Mazeiar, Li. Shimin, and T. Ladan. *A Metric-Based Heuristic Framework to Detect Object-Oriented Design Flaws* Proceedings of the 14th IEEE International Conference on Program Comprehension (ICPC06), 2006.

[5] P.F. Mihancea and R.Marinescu. *Towards the optimization of automatic detection of design flaws in object-oriented software systems*, In Proc. of the 9th European Conf. on Software Maintenance and Reengineering, 92-101, 2005.

[6] L. Tahvildari and K. Kontogiannis. *Improving design quality using meta-pattern transformations : A metric-based approach*, Journal of Software Maintenance and Evolution: Research and Practice, 16, 331-361, 2004.

[7] Open source project: log4net, http://logging.apache.org/log4net.

[8] A.J. Riel. *Object-Oriented Design Heuristics*, Addison-Wesley, 1996.

[9] S. Chidamber and C. Kemerer, *A metric suite for object- oriented design*, IEEE Transactions on Software Engineering, 20(6), 476–493, 1994.

[10] J.M. Bieman and B.K. Kang, *Cohesion and Reuse in an Object-Oriented System*, ACM Symposium on Software Reusability, 1995.

Department of Computer Science, Faculty of Mathematics and Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania

E-mail address: camelia@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 4, 2011

LOCATION BASED APPLICATION PERFORMANCE STUDY IN MYSQL

ADRIAN SERGIU DĂRĂBANT, VIORICA VARGA, LEON ŢÂMBULEA, AND BAZIL PÂRV

Abstract. This paper presents a short study and an optimization process based on data organization in mySQL databases. Spatial data is largely used nowadays not only on web but also in mobile and desktop applications. We show that organizing spatial data as spatial structures, as it comes naturally is not always the best approach when dealing with query response times. Although many database engines have strong spatial features, characteristics of the data and applications might fit better sometimes with relational modeling or even, in extreme cases, with deviated relational modeling.

1. Introduction

Location-based services (LBSs) are IT services for providing information that has been created taking into consideration the current locations of the users or mobile objects. They can also appear in conjunction with conventional services like telephony. LBSs' participants do not have to enter location information manually, they are automatically pinpointed and tracked.

The simplest type of LBSs are enquiry and information services, which provide the mobile user with nearby *points of interest* such as restaurants, movies, or filling stations. The user is automatically located by the mobile network. He must specify the points of interest, for example, whether he would like to receive a list of all nearby theaters or concerts, and the desired maximum distance between his current position and the points of interest. The request is then passed to a service provider, which assembles a list of appropriate points of interest and returns it to the user.

When dealing with LBSs it is first important to be clear about the meaning of the term *location*. Basically, the term location is associated with a certain

Received by the editors: October 20, 2011.

2010 *Mathematics Subject Classification.* 68P15.

1998 *CR Categories and Descriptors.* H.2.8 [**Database Management**]: Database Applications – *Spatial databases and GIS*.

Key words and phrases. relational database, query optimization, location based services.

place in the real world. These kinds of locations belong to the class of *physical locations*. There are also virtual locations (see [2]). There are different kind of physical locations, we are interested in spatial locations. A spatial location represents a single point in the Euclidean space. Another, more intuitive term for spatial location is position. A convenient way to express spatial location is to use an ellipsoidal coordinate system that models the Earths surface as an ellipsoid. A reference ellipsoid describes the shape of the Earth by fixing an equatorial and a polar radius. The origin of an ellipsoidal coordinate system is given by two reference planes, both arranged orthogonal to each other and crossing the geocenter. The horizontal plane corresponds to the equatorial plane. The vertical reference plane includes the Earths rotation axis and hence intersects North and South Poles.

A position at the surface of the Earth is then represented by the angles between the reference planes and the line passing from the geocenter to the position (see Figure 1). The latitude ϕ is defined as the angle between the equatorial plane and the line, that is, it represents the NorthSouth direction measured from the Equator, while the longitude λ describes the angle between the vertical plane and the line, that is, it reflects the EastWest direction of a position with regard to the vertical reference plane

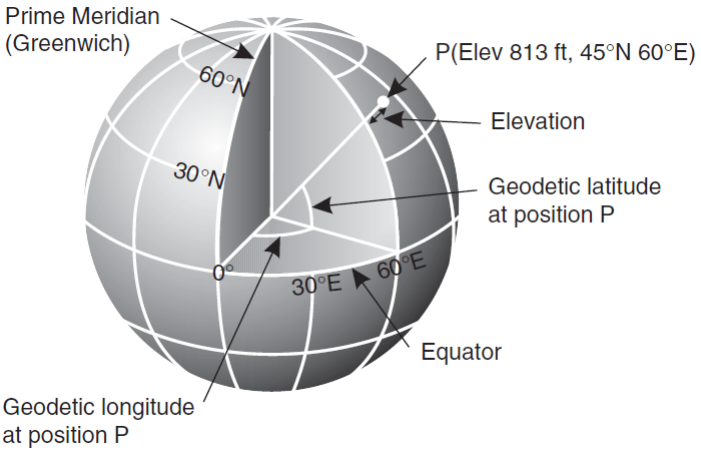


FIGURE 1. Latitude and longitude on Earth

Spatial location or position information represents an appropriate means for exactly pinpointing an object on Earth. One of the positioning methods used in LBSs is the Global Positioning System (GPS), for example, delivering 48°06′37″N 16°34′11″E we know that the target person currently resides at

the airport in Wien, Austria. A well known application is to determine the distance between the LBS user and a selected point of interest and to derive the expected traveling time from that. A simple approach would deliver only the geodesic (straight)line distance between both positions, but it would be more convenient for the user to get the shortest route distance in a road or public transportation network, preferably in combination with the shortest route displayed on a map and additional navigation assistance.

Spatial databases [3] and Geographic Information Systems (GISs) are the essential key technologies which deal with the mapping between spatial and descriptive location information as well with maintaining and deriving relationships between locations in general. Spatial database management systems support queries that involve the space characteristics of the underlying data. For example, a spatial database may contain polygons that represent building footprints from a satellite image, or the representation of lakes, rivers and other natural objects. It is important to be able to query the database by using predicates related to the spatial and geometric characteristics of the objects. To handle such queries, a spatial database system is enhanced by special tools. These tools include new data types, sophisticated indexing mechanisms and algorithms for efficient query processing that differ from their counterparts in a conservative alphanumeric database.

The aim of this paper is the study of query optimization in a mySQL database containing a large dataset with information about points of interest, events and their geographical coordinates. Data models explored in this paper are: normalized relational data, spatial data model and relational data model with materialized views. For query optimization we use B+ tree indexes and stored procedures in case of normalized relational data model, R-trees for the spatial model. The next section is about the used data models and the query performance for these models. The last section presents the experimental results in the proposed models.

2. mySQL relational versus spatial organization performance

In this paper we study and give solutions on optimizing data access to a large mySQL database about points of interest, events and their geographical coordinates. Being a POI database all queries that are running against the database are invariably filtering on the position coordinates. For most of them time is also involved as an interval. The first solution that comes to mind is to model data using the spatial features of the mySQL server in order to let the system perform the computations on the geographical coordinates. The test case database we try to optimize has between 5 and 10 GB of data with millions of non unique records containing geographical coordinates and the

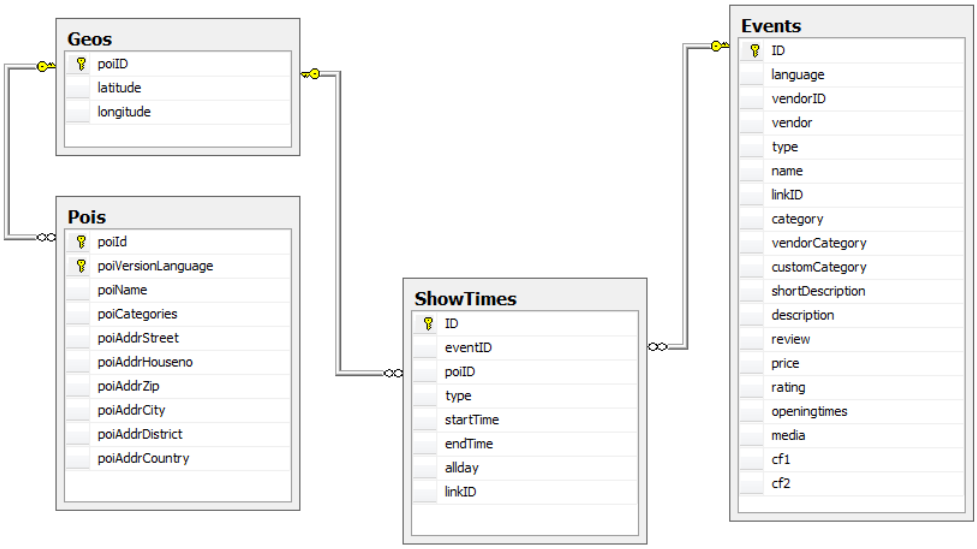


FIGURE 2. Simplified database scheme

main goal is to find a data organization/structure that has the best response times for queries.

In the following we will try to apply a few optimizations on the different data organizations in order to improve query response time while still maintaining mySQL as database engine. Most of the queries variants are searching different kind of events (and show times) and places in an area around the user position and within a specified time interval. The location of the user is given. The search for show times is made in the bounding box of the user's position. In the first approach we store the data in a relational database in normalized tables: `Geos`, `Pois`, `Events` and `ShowTimes`. Every point of interest has different name in every different language, so we store the location of the point of interest in the `Geo` table once and the name, address, etc. in table `Pois`. One event can appear with its show time at different points of interest, so we store the event once in table `Events` and its show times in table `ShowTimes`, where the `poiID` represents the location of the event. We present only a few of tables' columns. The simplified structure of the database is presented in Figure 2.

The problematic queries against this database involve the join of tables `Geos`, `Pois`, `Events` and `ShowTimes`. In this paper we pick five of the slow queries for presentation. The queries select show times of different events from crowded areas, like Paris, Wien, New York, etc. The location is parameter for

every query; we give the bounding box of the selected city. For example the latitude for Paris has to be between 48.6 and 48.93 and the longitude between 2.20 and 2.5. The start time and end time are parameters too; usually we select the show times next week after the current date. Every query returns the point of interest's location, name and address, the name of event and other information.

(1) The first query (QT1) selects the show times of *every* event category from one given vendor in the given location for the given time period.

```
SELECT e.ID, sh.poiID, e.language, e.type, e.vendor, e.vendorID,
   e.name, e.category, e.vendorCategory, p.poiVersionLanguage,
   GROUP_CONCAT(sh.startTime) AS START_TIMES, GROUP_CONCAT(sh.endTime),
   g.latitude, g.longitude, p.poiName, p.poiAddrCity,
  FROM (Events AS e INNER JOIN ShowTimes sh ON e.ID = sh.eventID)
    INNER JOIN Geos AS g ON sh.poiID = g.poiID
    INNER JOIN Pois AS p ON sh.poiID=p.poiId
  WHERE g.latitude BETWEEN ? AND ? AND g.longitude BETWEEN ? AND ?
    AND (sh.startTime >= ? OR sh.endTime >= ? OR
           (sh.startTime = ? AND sh.endTime IS NOT NULL))
    AND sh.startTime < ? AND sh.type = 0 AND e.vendor= ?
  GROUP BY sh.eventID, sh.poiID, e.language, p.poiVersionLanguage
  ORDER BY sh.startTime;
```

(2) The second query (QT2) searches the show times of *theater, movies, dance, etc* event categories from one given vendor in the given location for the given time period.

```
SELECT e.ID, sh.poiID, e.language, e.type, e.vendor, e.vendorID,
   e.name, e.category, e.vendorCategory, p.poiVersionLanguage,
   GROUP_CONCAT(sh.startTime) AS START_TIMES, GROUP_CONCAT(sh.endTime),
   g.latitude, g.longitude, p.poiName, p.poiAddrCity,
  FROM (Events AS e INNER JOIN ShowTimes sh ON e.ID = sh.eventID)
    INNER JOIN Geos AS g ON sh.poiID = g.poiID
    INNER JOIN Pois AS p ON sh.poiID=p.poiId
  WHERE g.latitude BETWEEN ? AND ? AND g.longitude BETWEEN ? AND ?
    AND (sh.startTime >= ? OR sh.endTime >= ? OR (sh.startTime= ? AND
           sh.endTime IS NOT NULL)) AND sh.startTime < ?
    AND e.type = ? AND e.vendor = ? AND  category like ?
  GROUP BY sh.eventID, sh.poiID, e.language, p.poiVersionLanguage
  ORDER BY sh.startTime;
```

(3) The third query (QT3) selects the show times of a *parameter* given event category from the second vendor in the given location for the given time period.

```
SELECT e.ID, sh.poiID, e.language, e.type, e.vendor, e.vendorID,
   e.name, e.category, e.vendorCategory, e.cf1, e.description,
   GROUP_CONCAT(sh.startTime) AS START_TIMES, GROUP_CONCAT(sh.endTime),
   e.media, g.latitude, g.longitude, p.poiName,
```

```
 p.poiAddrCity, p.poiVersionLanguage
FROM (Events AS e INNER JOIN ShowTimes sh ON e.ID = sh.eventID)
  INNER JOIN Geos AS g ON sh.poiID = g.poiID
  INNER JOIN Pois AS p ON sh.poiID = p.poiId
WHERE g.latitude BETWEEN ? AND ? AND g.longitude BETWEEN ? AND ?
  AND (sh.startTime >= ? OR sh.endTime >= ? OR (sh.startTime= ? AND
          sh.endTime IS NOT NULL)) AND sh.startTime < ?
  AND e.type = ? AND e.vendor = ? AND e.customCategory LIKE ?
GROUP BY sh.eventID, sh.poiID, e.language, p.poiVersionLanguage;
```

(4) The forth query (QT4) selects the show times of *cinema* point of interest category from the second vendor in the given location for the given time period.

```
SELECT e.ID, sh.poiID, e.language, e.linkID, e.vendorID, e.name,
   e.customCategory, e.rating, e.cf1, e.media,
   GROUP_CONCAT(sh.startTime) AS START_TIMES,
   g.latitude, g.longitude, p.poiName, p.poiVersionLanguage
 FROM (ShowTimes AS sh INNER JOIN Events AS e ON sh.linkID =e.linkID)
    INNER JOIN Geos AS g ON sh.poiID = g.poiID
    INNER JOIN Pois AS p ON sh.poiID = p.poiId
 WHERE p.poiCategories LIKE '%Cinema%'
   AND (sh.startTime >= ? OR sh.endTime >= ? OR (sh.startTime = ?
   AND sh.endTime IS NOT NULL)) AND sh.startTime < ?
   AND g.latitude BETWEEN ? AND ? AND g.longitude BETWEEN ? AND ?
   AND e.vendor=? AND sh.type = 1 AND e.cf2= ?
 GROUP BY e.id, sh.poiID, e.language, p.poiVersionLanguage;
```

(5) The fifth query (QT5) searches the show times of *concert* event category from one given vendor in the given location for the given time period.

```
SELECT e.ID, sh.poiID, e.language, e.name, e.category, e.vendorID,
  GROUP_CONCAT(sh.startTime) AS START_TIMES, GROUP_CONCAT(sh.endTime),
  g.latitude, g.longitude
 FROM (Events AS e INNER JOIN ShowTimes sh ON e.ID = sh.eventID)
    INNER JOIN Geos AS g ON sh.poiID = g.poiID
 WHERE g.latitude BETWEEN ? AND ? AND g.longitude BETWEEN ? AND ?
   AND (sh.startTime >= ? OR sh.endTime >= ? OR
      (sh.startTime = ? AND sh.endTime IS NOT NULL)) AND sh.startTime < ?
   AND (e.type = 1 OR (e.type = 0 AND e.category LIKE '%concerts%'))
   AND e.vendor = ?
 GROUP BY sh.eventID, sh.poiID, e.language
 ORDER BY sh.startTime
```

As we can see every query needs the join of the tables: `Geos, Pois, Events` and `ShowTimes.` We test these five queries as single SELECT statements against the MySQL database. The dates and location data were generated randomly from highly agglomerated cities around he world. The question marks are replaced with the actual query arguments.

In order to optimize the execution of the queries, first we study the execution plan of them, than we build additional index files for attributes from filter conditions or join conditions. The most widely used of several index structures in the relational approach is the B+ tree [6] that maintains efficiency despite insertion and deletion. It takes the form of a balanced tree in which every path from the root of the tree to a leaf of the tree is of the same length. In order to find a key in a B+ tree, the database engine has to read a number of blocks equal with the height of the tree. A leaf node stores the key together with the pointer (rid) to the data file. Using the rid of the record the whole record can be read with one I/O operation. An important property of the B+-tree index is its *fan-out* [4], which is the number of entries in a page. The height of the tree is proportional to $log_{fan-out}$(nr.data entries). In order to decrease the search time the fan-out has to be increased, so the height will decrease. We vary thus in our test the size of the block, in order to allow it to store more keys.

In the second approach we formulate the queries as stored procedures to benefit of already compiled execution plans. We try to force the execution of queries in different orders, processing the selections first and store the partial results in temporary tables.

In the third step we test the GIS functionality of mySQL. The `point` data type is introduced [5] for latitude and longitude, we have the next table:

```
CREATE TABLE ‘geoSpatial‘ (
  ‘poiID‘ varchar(64) NOT NULL,
  ‘location‘ point NOT NULL,
  PRIMARY KEY (‘poiID‘)
  )
```

Spatial data may be indexed too. R-tree ("R" from Region) is used for spatial data indexing ([1]). It involves organizing the minimum bounding rectangle (MBR) of the spatial objects in a tree structure. There are a few variations of R-tree indexing, MySQL uses R-trees with quadratic splitting [5], which is one of the standard methods of building an R-tree index. An R-tree index is similar to a B+-tree in many ways, and organizes the indexed nodes in a hierarchy where the nodes in the index represent the MBR of the objects in the node. The leaf nodes in the index contain references to the row that contain the data, just like a B+-tree index. An R-tree for the location column can be constructed with the following command:

```
CREATE SPATIAL INDEX sp_index
ON geoSpatial (location);
```

Query QT1 for point of interests near Wien using table `geoSpatial` is:

```
SET @g1 = GeomFromText(
’Polygon((48.1 16.2, 48.1 16.5, 48.3 16.5, 48.3 16.2, 48.1 16.2))’);
```

```
SELECT e.ID, sh.poiID, e.language, e.type, e.vendor, e.vendorID,
   e.name, e.category, e.vendorCategory, p.poiVersionLanguage,
   GROUP_CONCAT(sh.startTime) AS START_TIMES, GROUP_CONCAT(sh.endTime),
   X(g.location), Y(g.location), p.poiName, p.poiAddrCity,
  FROM (Events AS e INNER JOIN ShowTimes sh ON e.ID = sh.eventID)
    INNER JOIN geoSpatial AS g ON sh.poiID = g.poiID
    INNER JOIN Pois AS p ON sh.poiID=p.poiId
  WHERE MBRContains(@g1,g.location)
    AND (sh.startTime >= ? OR sh.endTime >= ? OR
           (sh.startTime = ? AND sh.endTime IS NOT NULL))
    AND sh.startTime < ? AND sh.type = 0 AND e.vendor= ?
  GROUP BY sh.eventID, sh.poiID, e.language, p.poiVersionLanguage
  ORDER BY sh.startTime;
```

In the fourth approach we use materialized views on the three main tables of the database and perform the same queries. We keep data fully organized in a relational manner. By default mySQL does not have support for materialized views but they can be fairly easy implemented with the help of the database engine primitives. There are also external implementations to mySQL (plug-ins) that export this functionality. One of these implementations: *flexviews*[7] provides even for incremental view update in order to optimize the insertions and table updates.

Finally there is a fifth approach where we try to load the entire database in a memory simulated disk, but with a proprietary specific file system in order to analyze the impact disk reading and seeking operations have to the overall processing.

3. Experiments and results

In the following paragraphs we show a comparative study of the five different scenarios proposed above. The approach with stored procedures exhibits exactly the same response time as executing the actual queries and is not represented in the following figures.

In the RAM backed database comparison we show the worst response time in order to pinpoint the impact of disk operations. According to Figure 3 it seems that the disk operations do not have a great impact on the overall query processing time. However, we note here that the only RAM file system available that could store a 5 GB to 10 GB database was a *linux tmpfs* which is different than a fully fledged disk based file system as ext3. Tests performed on smaller databases that are loaded in file systems based ram disks do show a better response time. Creating a large file system based RAM disk was impossible with the operating system configuration we used to run our tests. Although the physical test machine has 24 GB of RAM that could be used, the kernel limitations of the underneath operating system did not allow creating

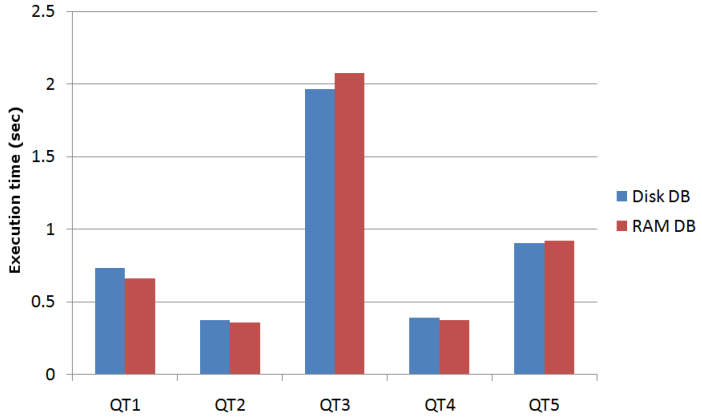


FIGURE 3. RAM loaded database versus on disk database.

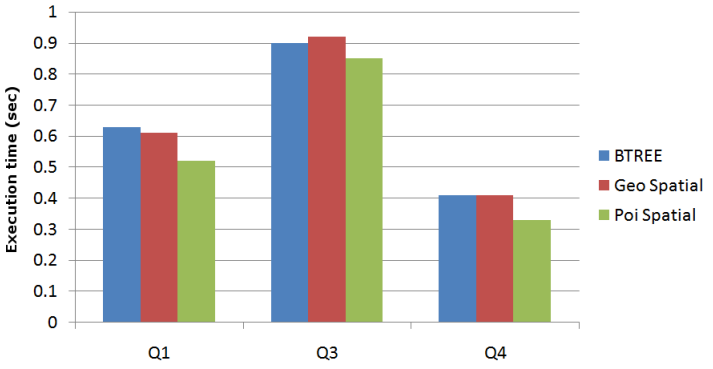


FIGURE 4. Spatial versus relational comparison

large usable file system based RAM disks. We did not focused our research on this variant because it is highly improbable for such a hardware dependent solution to be accepted in a practical production database system. We produced the RAM based experiments to merely show that the improvement of the response time is not that important to consider this approach as a potential solution. MySQL already caches data and can be instructed to keep indexes in system memory, fact that explains the insignificant improvements (where any are present) of the response time.

Figure 4 shows the average response times for the relational normalized approach and two spatial organizations with spatial data either in *Pois* or in

Geos tables. As observed the spatial approach improves only marginally the response time. Our initial approach was to model the entire dataset spatially. Searching in a well defined area often reduces to a few operations that could be easily implemented in pure SQL using the *between* and $<, >$ operators. When comparing the performance of the relational *between* operator, for range queries, with that of location aware primitives we found no major difference. Searching for points into rectangular or polygonal shapes is not faster than real numbers range checking using *between*. The entire approach using spatial features, as shown in figure 4, does not change the average response time in a significant way.

According to figure 5, a materialized view structure highly improves the average response time. The main bottleneck in the query evaluations (also according to the query execution plans) proves to be the execution of join operations. This is usually a high cost operation, but the fact that MySQL only implements nested loops [5] as join strategy penalizes the average response times. The materialized view schema helps cutting by a factor of two some of the average response times. As the main purpose was to find a solution that would allow executing between 2-5 queries per second this is the only approach that seems to help. During the experiments we also played with the database parameters, but the materialized view approach was the one that kept the best performance. For databases with high and very high frequency update and insert operations it is probably not the best approach as one needs a very good implementation, with incremental updates, of the materialized view functionality. The materialized view reduces the response time by a factor of two for most of the applications. The problem we tried to solve implies a moderate to very low frequency update which proves to be the ideal candidate for our last approach.

4. Conclusions and further research

This paper shortly presents a location aware application and its data stored in a mySQL database under various organization (relational, spatial structures, relational with materialized views). Our initial database modeling approach was a fully normalized relational. The main drawback of this model is that it could not stand a high concurrency degree because of the large response times required to evaluate the user queries. We investigated alternative data organizations in order to allow for concurrency factors of 2-5 queries/sec. Given the fact that the database is mostly static we discovered that for this case the best scenario is not a spatial organization with spatial indexes but a relational model with materialized views. Of course this approach is best suited to mostly static databases but we used the study to discover the major problems of the

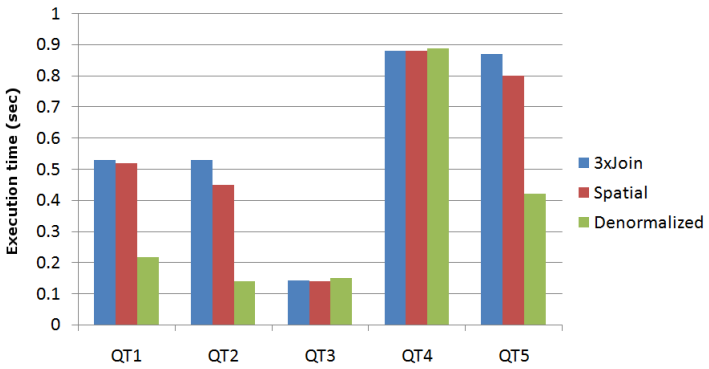


FIGURE 5. Relational, spatial and materialized view comparison.

mySQL engine in the context of spatial information and large databases in the context of simple spatial operations. We plan to adapt the current results in order to derive and implement a custom R-Tree mySQL indexing technique allowing loading large portions of the index in memory for fast geographical lookups, combined with a hash join execution strategy.

5. ACKNOWLEDGEMENT.

The author Viorica Varga has been fully supported by Romanian Ministry of Education in the frame of Research Grant CNCSIS PCCE-55/2008.

This work is also supported by Nokia Romania - Methods (Techniques) for Efficiently Searching in Spatial Data.

REFERENCES

[1] H. Garcia-Molina, J. D. Ullman, J. Widom: Database Systems: The Complete Book, Prentice Hall, 2009

[2] A. Küpper: Location-Based Services: Fundamentals and Operation. JOHN WILEY & SONS, LTD. (2005)

[3] Y. Manalopoulos; A. Papadopoulos; M. Gr. Vassilakopoulos: Spatial Databases: Technologies, Techniques and Trends, 2005.

[4] R. Ramakrishnan, J. Gehrke: Database Management Systems, Third Edition, WCB McGraw-Hill, 2003.

[5] Oracle Corporation: MySQL 5.1 Reference Manual, 2010.

[6] A. Silberschatz, H. F. Korth, S. Sudarshan: Database System Concepts, McGraw-Hill, Fifth Edition, (2005)

[7] * * *, FLEXVIEWS - Incrementally refreshable materialized views for MySQL, http://code.google.com/p/flexviews, consulted oct 2011.

Babes-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogălniceanu str., Cluj-Napoca 400084, Romania

E-mail address: dadi@cs.ubbcluj.ro

E-mail address: ivarga@cs.ubbcluj.ro

E-mail address: leon@cs.ubbcluj.ro

E-mail address: bparv@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 4, 2011

STORING LOCATION-BASED SERVICES' DATA IN KEY-VALUE STORE

VIORICA VARGA, ADRIAN SERGIU DĂRĂBANT, LEON ŢÂMBULEA, AND BAZIL PÂRV

Abstract. This paper proposes some solutions and discusses some issues related to the mapping of relational data and queries to a Key-Value store. The problem is stated for Location-based services (LBS), which deal with millions of users and thus millions of database records. The mapping process of a relational table consists in a transformation that assigns a part of the tables's row to the `Key` part and another part to the `Value`. We discuss two approaches to the mapping: generic and a LBS specific. The generic approach is a universal solution that can be applied to any relational to Key-Value mapping process.

1. Introduction

The data management research community is confronted with building consistent, available, and scalable data management systems capable of serving petabytes of data for millions of users especially for web application. Distributed database systems [13], [9] were the first generic solution that dealt with data not bounded to the limits of a single machine. These systems are not used frequently in industry because of their high complexity and due to the crippling effect on performance caused by partial failures and synchronization overhead.

Recent years different classes of scalable data management systems have been appeared such Google′s Bigtable [5], Amazon′s Dynamo [8] and PNUTS [6] from Yahoo! and others. All of these systems deal with petabytes of data, serve millions of requests with high availability requirements and run on cluster computing architectures. With the growing popularity of the cloud computing paradigm, many applications are moving to the cloud.

Received by the editors: October 25, 2011.

2010 *Mathematics Subject Classification.* 68P15, 68P20.

1998 *CR Categories and Descriptors.* H.3.2 [**Information Storage and Retrieval**]: Information Storage – *File organization*.

Key words and phrases. Key-Value store, relational model, data design, query mapping.

Nowadays mobile network operators create differentiation through the delivery of highly personalized services. One of the most powerful ways to personalize mobile services is based on location. These systems have millions of users. Location-based services (LBSs) are IT services for providing information that has been created, selected, or filtered taking into consideration the current locations of the users or mobile objects. They can also appear in conjunction with conventional services like telephony. The attractiveness of LBSs results from the fact that their participants do not have to enter location information manually, but that they are automatically pinpointed and tracked.

The simplest type of LBSs provide the mobile user with nearby *points of interest*s such as hospitals, parking places, restaurants, movies, or filling stations. The user is automatically located by the mobile network. He must specify the points of interest, for example, whether he would like to receive a list of all nearby theaters or concerts, and the desired maximum distance between his current position and the points of interest. The request is then passed to a service provider, which assembles a list of appropriate points of interest and returns it to the user. Location-based services in mobile network store usually the required data in distributed database systems. The response time of these systems for a large number a users is not adequate. In a current work [7] we study the performance of *location based applications*. In the previous approach [7] the problem of finding events, the events' show times in the current week for a user was studied. The search for show times was made around the user's position. The data has been stored in a relational database in normalized tables and the response time was not acceptable.

[1] analyzes the design choices that allowed modern scalable data management systems to achieve orders of magnitude higher levels of scalability compared to traditional databases. The authors give some possible design solutions for data management in the cloud.

Database research recognizes today [4] that different data models and database technologies should be used in different application domains. As an example, many companies in the Web industry have abandoned traditional relational DBMSs for so-called "No-SQL" data stores. This paper refers to *key-value* stores. Our goal is to analyze the pros and cons of transferring huge amounts of data of a location-base application from a distributed relational DBMS to a *key-value* store in the cloud in order to exploit the performance of the latter and, in the same time, to provide good response times for queries. The goal of this paper is to lay some design principles for the data management systems serving the next generation of applications in the cloud.

The structure of the paper is as follows. After this introductory section, the following section introduces the technical details of a *No-SQL* store. The

original part of the paper follows. The next two sections discuss in some detail the options regarding data design and storing, and relational query mapping to *key-value* stores. Last section draws some conclusions and sets future research directions.

2. No-SQL store

The data model for "No-SQL" store it is not a unique one. It can be build by basic data elements called "documents", "objects", or "records". According to [4] its essence is not the lack of SQL, but the presence of the following features:

- a little or no pre-defined schema; objects, records, or documents can have any number of attributes of any type;
- a simple query interface, and not a SQL processor;
- high scalability over dozens or hundreds of nodes, with the price of giving up 100% ACID semantics;
- eventual consistency - guaranteed consistency only within a single object, record, or document;
- high availability, necessary to make scalability across many machines useful.

Considering their data model and functionality, "No-SQL" data stores can be split into three groups: *key-value stores*, *document stores* and *extensible record stores*. We will use *key-value* stores, which have a distributed index for object storage. The stored objects are not interpreted by the system; they are stored and handled back to the client application as BLOBs. Basic functionality of *key-value* stores usually include object replication, partitioning the data over many machines, and a sort of object persistence;

A *key-value* store record has two parts: `Key` and `Value` part.

Different client API's were elaborated for different systems, see for example [2], [3]. We present the interface of [2] which includes the following operations:

- `STORE`: stores (*key, value*) in the file;
- `ADD`: adds (*key, value*) to the file iff the lookup for *key* fails;
- `REPLACE`: replaces (*key*, $value_1$) with (*key*, $value_2$) based on (*key*, $value_2$);
- `GET`: retrieves either (*key, value*) or a set of (key_i, $value_i$) pairs based on *key* or $key_i, i = 1 \dots k$.
- `DELETE`: deletes (*key, value*) from the file based on *key*

Taking into account these facts, the process of mapping a relational database to a *key-value* store is split into two sub-processes: migrating the data and transforming the queries.

3. Data design and storing in key-value store

The structure of one record in a key-value file being simple, the `Key` part and the `Value` part have to be determined. In order to know the structure of the record in a key-value file, we have to store its metadata. It can be represented using well-known formats like XML or JSON. The structure of the *key-value* record for LBS applications after the mapping can be described using (a) a generic style or (b) a custom structure for spatial data. The first solution is called generic because it can be applied to any relational database.

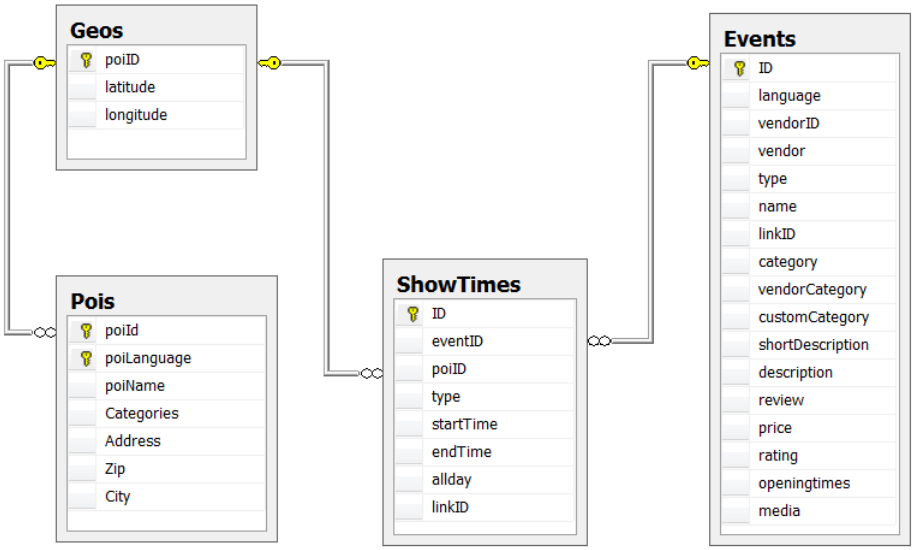


Figure 1. Simplified database scheme

3.1. **Generic style.** In this article the running example is about finding events, the events' show times in the current week for a user in a LBS application. The location of the user is known, the search for show times is made in the user's bounding box, which is a rectangle. In the first approach the structure of the data is presented in a relational database in normalized tables: `Geos`, `Pois`, `Events` and `ShowTimes`. Every point of interest has a different name for each different language, so we store the location of the point of interest in the `Geo` table once and the name, address, etc. in table `Pois`. One event can appear with its show time at different points of interest, so we store the event once in table `Events` and its show times in table `ShowTimes`,

where the `poiID` represents the location of the event. The simplified structure of the database is presented in Figure 1.

Now the proposed structure of the *key-value* record is presented. The metadata describes for each table of a database: the key part in `primaryKey` tag, the structure of the record in `Attribute` tags, index files in `IndexFile` and `IndexAttributes` tags, and the relationships in `foreignKey` tags.

```
− <Databases>
  − <DataBase dataBaseName="SpatialPOIs">
    − <Tables>
      − <Table tableName="geo" fileName="geo.kv" rowLength="114">
        − <Structure>
            <Attribute attributeName="poiID" type="char" length="64" isnull="0"/>
            <Attribute attributeName="latitude" type="double" isnull="0"/>
            <Attribute attributeName="latitude" type="double" isnull="0"/>
          </Structure>
        − <primaryKey>
            <pkAttribute>poiID</pkAttribute>
          </primaryKey>
        − <IndexFiles>
          − <IndexFile indexName="geoLat.ind" keyLength="25" isUnique="0" indexType="BTree">
            − <IndexAttributes>
                <IAttribute>latitude</IAttribute>
              </IndexAttributes>
            </IndexFile>
          − <IndexFile indexName="geoLong.ind" keyLength="25" isUnique="0" indexType="BTree">
            − <IndexAttributes>
                <IAttribute>longitude</IAttribute>
              </IndexAttributes>
            </IndexFile>
          </IndexFiles>
        </Table>
      + <Table tableName="poi" fileName="poi.kv" rowLength="626"></Table>
      </Tables>
    </DataBase>
  </Databases>
```

FIGURE 2. The structure of the data

Example 1. For the running example the structure of `geo` *key-value* file is on Figure 2. The primary key and index files of the table can be seen too. In Figure 3 the metadata for `poi` *key-value* file is presented with the relationship (`foreignKey` tag) to `geo` *key-value* file.

Transforming a table `T` from the relational DB to the *key-value* store yields a set *key-value* files (usually more than one). The data from one table is stored in a *key-value* file, which we will name `master` file, where the `Key` part is the primary key of the table (`primaryKey` tag) and the `Value` part

```
- <DataBase dataBaseName="SpatialPOIs">
  - <Tables>
    + <Table tableName="geo" fileName="geo.kv" rowLength="114"></Table>
    - <Table tableName="poi" fileName="poi.kv" rowLength="676">
      - <Structure>
          <Attribute attributeName="poiID" type="char" length="64" isnull="0"/>
          <Attribute attributeName="poiLanguage" type="int" isnull="0"/>
          <Attribute attributeName="poiName" type="varchar" length="100" isnull="0"/>
          <Attribute attributeName="Categories" type="varchar" length="255" isnull="0"/>
          <Attribute attributeName="Address" type="varchar" length="100" isnull="0"/>
          <Attribute attributeName="Zip" type="varchar" length="25" isnull="0"/>
          <Attribute attributeName="City" type="varchar" length="100" isnull="0"/>
        </Structure>
      - <primaryKey>
          <pkAttribute>poiID</pkAttribute>
          <pkAttribute>language</pkAttribute>
        </primaryKey>
      - <foreignKeys>
        - <foreignKey>
            <fkAttribute>poiID</fkAttribute>
          - <references>
              <refTable>geo</refTable>
              <refAttribute>poiID</refAttribute>
            </references>
          </foreignKey>
        </foreignKeys>
      - <IndexFiles>
        - <IndexFile indexName="poiPoID.ind" keyLength="64" isUnique="1" indexType="BTree">
          - <IndexAttributes>
              <IAttribute>poiID</IAttribute>
            </IndexAttributes>
          </IndexFile>
        - <IndexFile indexName="poiName.ind" keyLength="100" isUnique="1" indexType="BTree">
          - <IndexAttributes>
              <IAttribute>poiName</IAttribute>
            </IndexAttributes>
          </IndexFile>
        </IndexFiles>
      </Table>
```

FIGURE 3. The structure of the data with relationships

is the concatenation of non-primary key attributes, so every attribute from `Structure` tag except the primary key of the table.

One table may have one or more unique index files. For every unique index file a *key-value* file has to be created: the `Key` part is the search key of the

index file, namely the concatenation of `IAttribute` values. The `Value` part of the unique index file is the corresponding `Key` part of the master file, which is the primary key of the `master` file. The search process on a unique index key requires two `GET(Key)` methods, the first on the unique index and the second on the master file, using as a key the value obtained in the first `GET(Key)` method. Where the underneath implementation allows access to the address of the `Value` part of the master file one can alternatively store this address as the `Value` part in the unique index. This approach avoids an additional data access when the index is used.

For every non-unique index file a new *key-value* file is created. More solutions can be given. If the *key-value* store accepts duplicate key values, then the mapping is immediate. Otherwise one can use solutions based on inverted indexes or *making search key unique* as proposed in [11] pp. 356-358.

Summarizing the above, for each table `T` the following *key-value* files have to be produced:

- the data from `T` is stored in a `master` *key-value* file `T.key-value` with `Key = T.PrimaryKey` and `Value` = concatenation of non-primary key attributes of `T`;
- for every unique index file `T.I` of `T` a new `T.I.key-value` file is created with `Key` = concatenation of `IAttribute` values and `Value = T.Key` (master);
- for every non-unique index file `T.I` of `T` a new `T.I.key-value` file is created. Different solutions: inverted index or making search key unique or duplicate key values.

Example 2. For relational table `poi` the following *key-value* files have to be produced:

- the data from the table is stored in a `masterPOI` *key-value* file with `Key = (poiID+poiLanguage)` and `Value` = `poiName+Categories+Address+Zip+City`;
- one *key-value* file for the `poiName` unique index key with `Key = poiName` and `Value = (poiID+poiLanguage)` ;

3.1.1. *Inverted index for non-unique index.* A single *key-value* record for each search key is built, with `Key` = search key value and `Value` = list of tuple pointers of the records (primary keys from master), where the corresponding search key is in the value part.

Example 3. Latitude is a search key for `geo` table (see geoLat.ind) and let be 3 poiID: 23456XX, 45678YY, 67890ZZ for which the latitude is 48.3. Value

48.3 will be stored once in the Key part and Value part will store the list of poiID-s: 23456XX;45678YY;67890ZZ, see Figure 4.

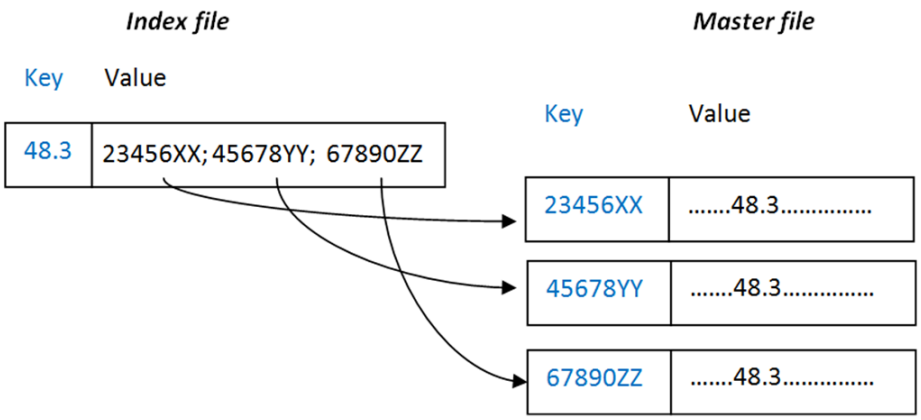


FIGURE 4. Inverted index example

3.1.2. *Make search key unique.* For each search key many *key-value* records are built, one for each record in the master with the same search key value. This is done by making search key unique, adding a record-identifier. In other words, `Key` = (search key + record identifier) and `Value` = NULL.

Example 4. For the example above, 3 different *key-value* records are built: (48.3#23456XX, NULL); (48.3#45678YY, NULL); (48.3#67890ZZ, NULL); See solution on Figure 5.

The "#" is used to separate the key part from unique identifier. The use of a separator can be avoided if constant (maximum) key-length is implemented. Shorter than maximum length keys are padded with spaces/blanks.

The option of making the search key unique, which is widely used for indexing relational databases, adds extra storage overhead for keys and extra custom code for insertion/deletion operations in the master file.

Inverted index option compared to *making the search key unique* adds low space overhead and no extra cost for queries, but may need extra code for handling long lists; also, deletion of a record in the master file may be expensive.

Compared to the relational model, finding a record in a *key-value* store using an index needs an additional read (`GET`) operation.

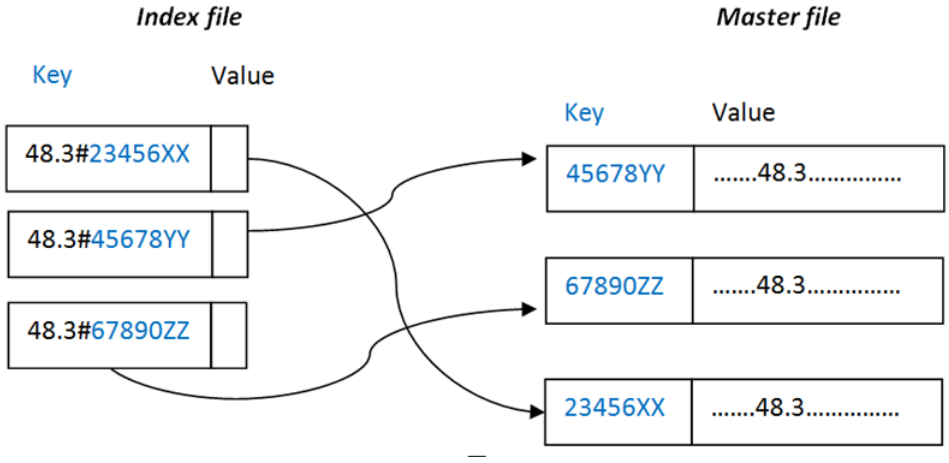


FIGURE 5. Making key unique example

3.2. **Custom solution for spatial data.** In this section we consider the design of the same environment in a *key-value* store. The divisions of the Earth in squares or rectangles called `Regions`, have to be stored in a `Regions` *key-value* file. Each region will have a unique key: the `RegionID`, so `Key` = `RegionID`. The `Value` part will have the following structure:

`UpperLeftLat,UpperLeftLong,LowerRightLat,LowerRightLong`

The poi-s, events, show times, can be stored in `Pois` *key-value* file. The aggregation of objects can be designed in different ways. One solution is a hierarchical style which can be described using regular expressions as follows:

```
geo(poi*, latitude, longitude);
poi(poiName,poiLanguage,Categories,Address,Zip,City,event*);
event(eventName, language, category, showtime*);
showtime(startTime, endTime);
```

`Pois.Key` = Region ID, `Pois.Value` = concatenation of `geo` composite objects, including poi-s, events, show times. By storing objects in this manner, information for one region is contained in a single object, so join operations are not necessary. The solution is usable, if such a `geo` object fits in memory. If it does not fit in memory, the region can be divided into more regions. One can use multiple division dimensions: regions, time, regions + time, etc.

Usually queries on spatial data involve not just a point (latitude, longitude), but a search area around the given point. This implies an additional radius. According to the location of the point in the region, multiple cases arise.

- the point and the search radius are completely contained in a single region
- the point and the search radius are contained in multiple regions

In a real system the search radius should be smaller than the region dimensions $r \leq l \leq L$, where r is the search radius, l and L are the 2 dimensions of the region's bounding box. As a result a search area will not overlap more than four neighbourhooding regions.

4. Querying a key-value store

The mapping of queries needs to take into account the structure of the metadata, described in the previous section.

4.1. **Generic style.** In this case, there is a general solution: implement the `SELECT` statement over a *key-value* store (for more details see [15, 11]). This general solution is not discussed here.

The remaining part of this section discusses separately how to transform one-table queries and multiple-table queries, respectively. In the case of one-table queries, search by unique key is mapped straightforward, using `GET` on master and index files.

Search by non-unique key may use one of the following:

- *Use of inverted indexes.*
- *Make search key unique by adding a record-identifier.*

4.1.1. *Inverted index for non-unique index.* The search process is straightforward; the programmer has to handle the list of primary keys from *key-value* index files and retrieve every record from *key-value* master with the searched key, using `GET`.

4.1.2. *Make search key unique by adding a record-identifier.* The search is much difficult, because the search for one key value is transformed to a range query and range queries are not implemented yet in *key-value* stores [16, 12]. In our opinion, B+ Tree index is suitable for range queries, if the *key-value* store allows direct access to the tree.

Example 5. The search of all poiID-s for latitude 48.3 is transformed to the following range query:

```
minKey=48.3# + possible min value
maxKey=48.3# + possible max value
minKey <= Key <= maxKey
```

If the index file on Key is clustered, the minKey is found using the B+ tree and the data file is scanned sequentially from there until the record with $Key > maxKey$ is encountered.

In the case of multiple-table queries, each specific query is mapped in a custom way. However, some general guidelines can be given. Generally speaking, such a complex query is executed in a five-step process, as follows:

(1) apply the filter for each table, which reduces to a one-table query;
(2) establish a join order;
(3) compute the join in memory if possible; if not, apply hash join or other join types;
(4) compute `GROUP BY` operation of the query;
(5) compute `SORT` operation of the query.

For the implementation of these operation see [15, 11].

4.2. **Custom solution for spatial queries.** If the division in regions is adequate, the search is restricted to locate the location point of the current user in a region (this will need an index file on longitude and/or latitude), then read from `Regions` *key-value* file by regionID, and the complex object will be in the corresponding `Value`.

5. Conclusions and further work

In this paper we investigated the opportunity of using *key-value* stores as a replacement for traditional relational DBMSs. Two issues were discussed: data design and query mapping, together with the additional costs incurred. Future work on this topic will include implementation and testing issues on concrete spatial data models.

6. Acknowledgement.

The author Viorica Varga has been fully supported by Romanian Ministry of Education in the frame of Research Grant CNCSIS PCCE-55/2008.

This work is also supported by Nokia Romania - Methods (Techniques) for Efficiently Searching in Spatial Data.

References

[1] D. Agrawal, A. E. Abbadi, S. Antony, S. Das: Data Management Challenges in Cloud Computing Infrastructures, *Databases in Networked Information Systems, 6th International Workshop*, DNIS 2010, Aizu-Wakamatsu, Japan, March 29-31, pp. 1-10

[2] M. Berezecki, E. Frachtenberg, M. Paleczny, K. Steele: Many-core key-value store, in *Green Computing Conference and Workshops (IGCC), 2011 International*, Orlando, July, pp. 1-8

[3] C. Bunch, J. Kupferman, C. Krintz Active Cloud DB: A. Database-Agnostic HTTP API to Key-Value Datastores. In UCSB. CS Technical Report 2010-07

[4] R. Cattell: Relational Databases, Object Databases, Key-Value Stores, Document Stores, and Extensible Record Stores: A Comparison, http://www.odbms.org/download/Cattell.Dec10.pdf

[5] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, R.E. Gruber: Bigtable: A Distributed Storage System for Structured Data. In: *OSDI*. 2006, pp. 205218

[6] B.F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.A. Jacobsen, N. Puz, D. Weaver, R. Yerneni: PNUTS: Yahoo!s hosted data serving platform. *Proc. VLDB Endow. 1(2)*, 2008, pp. 12771288

[7] A.S. Dărăbant, V. Varga, L. Ţâmbulea, B. Pârv: Location Based Application Performance Study in mySQL, to be appeared in Studia Universitatis "Babes-Bolyai" Cluj-Napoca, Informatica.

[8] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, W. Vogels: Dynamo: amazons highly available key-value store. *SOSP*. 2007, pp. 205220

[9] B.G. Lindsay, L.M. Haas, C. Mohan, P.F. Wilms, R.A. Yost: Computation and communication in R*: a distributed database manager. *ACM Trans. Comput. Syst. 2(1)*, 1984, pp. 2438

[10] H. Garcia-Molina, J.D. Ullman, J. Widom *Database Systems: The Complete Book*, Prentice Hall, 2008.

[11] R. Ramakrishnan, J. Gehrke, *Database Management Systems*, 3rd Edition, WCB McGraw-Hill, 2003.

[12] S. Ramabhadran, S. Ratnasamy, J.M. Hellerstein, S. Shenker Prefix Hash Tree: An Indexing Data Structure over Distributed Hash Tables, Technical Report, 2004 http://berkeley.intel-research.net/sylvia/pht.pdf

[13] J.B. Rothnie Jr., P.A. Bernstein, S. Fox, N. Goodman, M. Hammer, T.A. Landers, C.L. Reeve, D.W. Shipman, E. Wong: Introduction to a System for Distributed Databases (SDD- 1). *ACM Trans. Database Syst. 5(1)*, 1980, pp. 117

[14] T. Shimizu, M. Yoshikawa Full-Text and Structural Indexing of XML Documents on B+-Tree, *IEICE TRANSACTIONS on Information and Systems*, Vol. E89-D(2006), No.1, pp. 237-247

[15] A. Silberschatz, H. Korth, S. Sudarshan *Database System Concepts*, McGraw-Hill, New York, 2006.

[16] Range queries in *key-value* systems, http://groups.google.com/group/project-voldemort/browse_thread/thread/cad4888b492d897f

Babes-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogălniceanu str., Cluj-Napoca 400084, Romania

E-mail address: ivarga@cs.ubbcluj.ro

E-mail address: dadi@cs.ubbcluj.ro

E-mail address: leon@cs.ubbcluj.ro

E-mail address: bparv@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 4, 2011

MIRA –UPPER LIMB REHABILITATION SYSTEM USING MICROSOFT KINECT

ALINA CĂLIN, ANDREI CANTEA, ANDREI DASCĂLU, COSMIN MIHAIU, AND DAN SUCIU

Abstract. In the current context of profound involvement of technology in different areas of science, our research approaches the medical field, aiming to help the physical recovery process of people having a temporary handicap at the upper limb.

The result of our research is a software application called MIRA (Medical Interactive Recovery Assistant) that uses the Microsoft Kinect sensor, which, having an RGB camera, an IR camera and a depth sensor, makes possible the detection of the arm and its movement relative to the body. The application comes in the help of doctors, who receive important statistics and data about the performance of the patients during the exercises and games. Furthermore, receiving accurate data about the moves, the patients do not need to be supervised by a medical professional at all times, as the correctness of the exercises are verified automatically.

1. Introduction

In the last decades, medicine has improved its therapeutic procedures and techniques, making use of numerous modern approaches that are given by the exponential development in technology and other sciences. Being well known that the physical health is strongly related to the mental condition of the patients, to their motivation and engagement in fighting the illness they have to face with, a lot of studies have focused on how to improve their moral, by bringing fun or interesting elements in the recovery treatment.

Received by the editors: November 11, 2011.

2000 *Mathematics Subject Classification.* 68N01, 68U20.

1998 *CR Categories and Descriptors.* code [**J. Computer Applications**]: J.3 LIFE AND MEDICAL SCIENCES – *Health, Medical information systems*; code [**H.5 INFORMATION INTERFACES AND PRESENTATION**]: H.5.2 User Interfaces – *User-centered design, Input devices and strategies*.

Key words and phrases. medical software, physical recovery, games, Kinect.

In this context, occupational therapy is known to promote health by enabling people to perform meaningful and purposeful occupations like enjoying life, being socially and economically productive, being involved in community or group activities [2]. In this area, the use of Video-Games together with Virtual and Augmented Reality is more and more acknowledged by recent studies, not only as an occupational therapy framework basis, but also in physical therapy and rehabilitation, for both adults and children with movement problems having neurological or trauma causes.

Regarding these, our solution MIRA (Medical Interactive Recovery Assistant) proposes to assist and augment the physical recovery process of patients with movement disabilities (at the upper limb) caused by neurological, orthopedic and rheumatoid problems, by the use of interactive applications and games that monitor the patients' movement and engage them in performing the exercises recommended for recovery. This is something that many researchers have recently tried to achieve using different devices and sensors for patients' movements detection, like the Wii-mote, but the innovation we bring is given by the use of a latest generation sensor , the Microsoft Kinect, which tracks the human motion without requiring any devices attached to the body. Even more, the application provides meaningfull feedback, like statistical data about the patients' movement, which can be used by the doctor as an evaluation of the patients' state and improvement.

This way, it is supposed to improve the recovery process, shortening the period (it is known that physical exercises reduce the recovery time with 50% to 70%) and saving time and money for the medical institutions, while being attractive, engaging and adaptive to the patients' needs.

The next section describes comparatively the results obtained by similar studies and applications, such as Wii-hab. Section three contains a list of used technologies and a detailed presentation of Kinect: its capabilities, possible improvements and extensions. Also, the features provided by the open SDK used for programming Kinect will be presented here. The solution and its functionality will be explained in the fourth section: what does MIRA offer, how does it work in practice, the exercises mapped till now for the upper limb and the games created on this basis, how the data about the movements are collected and interpreted, which were the main challenges and which are the possibilities for future extension. Last, will be presented the evaluation results made on the patients at the Rehabilitation Hospital of Cluj-Napoca and the conclusions and future perspectives regarding the development of MIRA.

2. Related work

The use of virtual reality in medicine is under a continuous study for the past years, starting with simulation environments for practicing surgery and other medical interventions, continuing with diagnosing from 3D data mapping of human body and ending with psychological and physical rehabilitation therapies. Therefore, MIRA was inspired by many ideas already seen in existing application and studies.

Regarding this area of study, we know that the recovery of patients is usually a lengthy process that requires professional supervision. From our perspective, the best improvement would be a device that monitors the patients' movement and progress, while providing visual feedback during the rehabilitation exercises.

This was achieved by other studies, through games that use sensors for collecting data about the patients movement, having an adaptive difficulty for every patient and a winning rate of 80%, in order to keep the patient engaged in making exercises and avoid getting him or her bored because of always winning or frustrated because of low performance. Some good examples of such systems and studies, mainly as an aid to recovery therapies of stroke patients, are the ones designed by RIVERS Lab, using the Glove and other external devices for input in Virtual Reality applications [4], by VividGroups Gesture Xtreme Virtual Reality using the PlayStation II EyeToy [5] and a very interesting system using Webcam based Augmented Reality applications [7]. The clinic Northern Arm & Hand Center for Upper Limb Rehabilitation in Duluth, Minessota, uses VR games in this kind of therapy as they wanted to create a relaxed and family-oriented environment for the patients, and noticed that many patients over-exercise because they enjoy so much playing, and forget about their condition, fastening the recovery [9]. A Canadian study managed to measure the percentage of improvement in case of a VR and Video Games based therapy in Stroke Rehabilitation in comparison with classical recreational routine therapy, on a number of twenty stroke survivors; after two weeks the results in terms of improvement measured on different tasks were with 30% more advanced in the case of th patients that used games during the therapy [8].

A very similar application to ours is Wii-hab, based on the Wii-fit system, that uses the Wii-remote for movement detection to implement a system on

exercises for limb recovery in different types of injuries (the system is functioning, but still in development, exploring its potential) [**?**]. The difference is in that they use existent games which are likely to require adaptations to the limitations of the patients with disabilities, and their target is much wider, including a large spectrum of therapeutic procedure, while MIRA is specialized on physical rehabilitation and adapts to the patient's limited range of motion (of the upper limb).

In this context, this system would be the first one that is created specifically to assist the recovery process of patients in an interactive and fun way, not just using existing games, but creating them so that any patient would enjoy playing while recovering, making use of a last hour wireless device, the Kinect.

3. Microsoft Kinect

3.1. **Motion Sensor.** "Microsoft Kinect (Figure 1) is based on a software technology developed internally by Rare, a subsidiary of Microsoft Game Studios owned by Microsoft, and on range camera technology by Israeli developer PrimeSense, which interprets 3D scene information from a continuously-projected infrared structured light." [1] This system employs a variant of image-based 3D reconstruction, which actually means that it is very convenient for every person, as it is wireless, requires no devices attached to the body, and permits total freedom in movement.

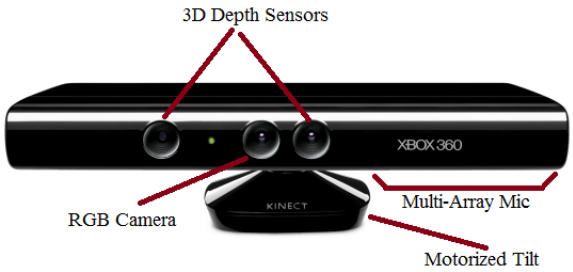


Figure 1. Microsoft Kinect for Xbox 360 [1]

Physically, the Kinect sensor "is a horizontal bar connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. The device features an RGB camera, depth sensor and multi-array microphone running proprietary software,which provide

full-body 3D motion capture, facial recognition and voice recognition capabilities.[...] The depth sensor consists of an infrared laser projector combined with a monochrome CMOS sensor, which captures video data in 3D under any ambient light conditions. The sensing range of the depth sensor is adjustable, and the Kinect software is capable of automatically calibrating the sensor based on game-play and the player's physical environment, accommodating for the presence of furniture or other obstacles." [**?**]

The most interesting thing is that Kinect offers the possibility of "simultaneously tracking up to six people, including two active players for motion analysis with a feature extraction of 20 joints per player" [1], the number of people the device can detect being limited only by how many will fit in the field-of-view of the camera.

3.2. **Programming Kinect using the OpenNI library.** MIRA was initially created using the OpenNI SDK with the NITE library and the PrimeSense drivers that provide the possibility of programming in many languages, and on different platforms. The one used here, on Windows OS, in C# (Visual Studio 2010), provides us with the possibility of detecting gestures (hand wave, hand click), the entire human body, or partially, after recognizing a pose body position similar in shape to the Greek letter "ψ"(psi) as presented in Figure 2. The wave gesture detection is used also for the hand position detection, that can be tracked during the application or game, after this pose position was detected. If the hand tracking is lost for a short period, the simple gesture of raising the hand will help quickly recognizing it, and it will start being tracked again. [3]

The human body recognition and body parts tracking is done in several steps. First, a movement with the whole body in front of the sensor will signal that a person is there and it will wait for the pose detection. When the "ψ" position is identified (standing straight, with hands up, forming a right angle at the elbow and between the trunk and forearm) it starts the calibration process: the joints positions are calculated. When this is completed, all the user's motions and body parts are tracked: head, neck, hand, elbow, shoulder, knee, etc. It is possible to make the calibration detect the entire body (which requires standing up), or only for certain regions: the upper part (which can be done even sitting down), or only head and hands. In case the tracking of one or more joints is lost, reassuming the "ψ" position will quickly solve the

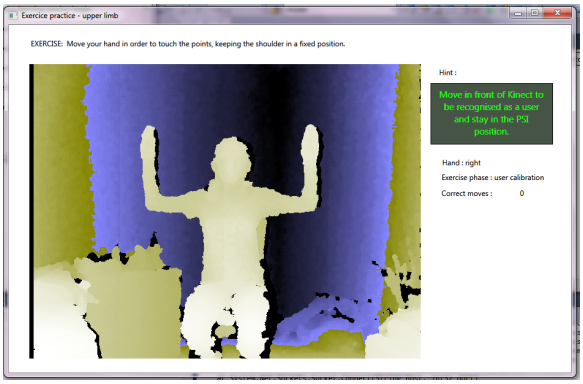


FIGURE 2. The "ψ" position of the body, while sitting on a chair. The depth map is visualized from white to yellow (near) and from light to dark blue (far).

problem. The best part is that many users can be tracked at once, at least two or three, even if they overlap and move in front of the camera.

4. The idea of MIRA

4.1. **What is MIRA.** Knowing these new tendencies in medicine, MIRA, as we designed it, should build an appropriate framework for the fulfillment of the medicine's new objectives. It is a system composed of interactive applications and games based on external sensors, created specifically to assist the patients during rehabilitation therapy.

The applications and games are created such that they require the same moves as in the case of a medical exercise, while monitoring the movements of the patient, in order to evaluate the correctness of the exercises (example in Figure 3), to adapt to the patients possibility of movement and to provide a statistical feedback about its performance and improvement. This way, the presence of the specialized medical staff is not required at all times, as the movements are correctly supervised, and this kind of therapy could replace a large amount of classical exercises.

By providing a good visual and audio feedback, these applications and games are very engaging and motivating for patients, being powerful stimuli to overcome their limits and to regain reflexes and dexterity. Moreover, the games are conceived such that the winning rate is about 80%, in order to keep the patient motivated to play and not frustrated by too much losing or bored

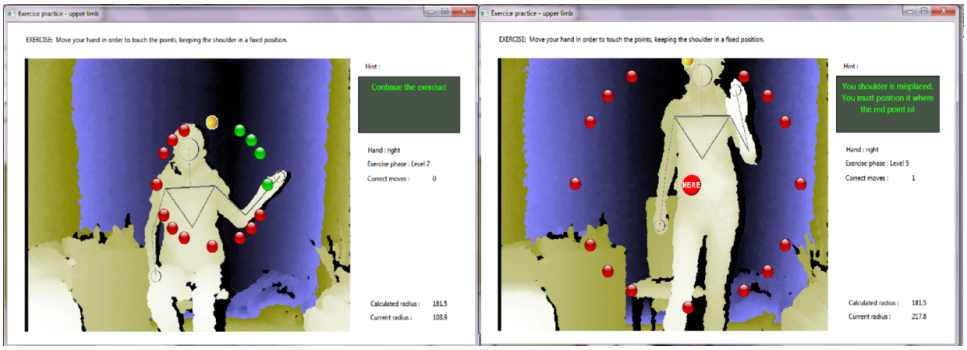


FIGURE 3. Patient performing rotation exercise for the shoulder in a correct manner (left) and trying to cheat, raised up from the chair (right the red point specifies that the shoulder should be placed there).

by too much winning. The applications perform the calibration, registering the range movement (the basis, calculated for each patient) and then sets the target range 10% higher during the exercises, in order to push the patient towards improvement.

Initially, for monitoring the patients' movement, we used an accelerometer, because it was very good for detecting the arm moves in four directions, and approximating the acceleration of the movement , while being attached to the patients' hand. The idea was to create a system that could be used with any device that contained an accelerometer, like a Wii-mote, a smart phone and so on, in order to keep extra costs low and increase its range of applicability for any kind of person. Despite this, it had different flows: for more complex motions, something more sophisticated was needed to monitor the movement, to prevent cheating and to be able to monitor even patients that are able to do only small limited movement, with a very small acceleration. This required the integration with some other sensors that would have increased the costs very much and lead us far from the main idea of making it universal and adaptive to other existing devices.

But looking at the latest technology products, the Microsoft Kinect was a single sensor providing a complete feedback with respect to what was needed: detects the human body with its components and joints, providing feedback about its position in a 3D space. It is wireless, requires no other devices attached on the human body, does not depend on light, magnetism or other

sensible environment factors, because it is based on a web-cam, an infrared-cam and a depth sensor that complete each other in detecting 3D objects.

Another feature that brings efficiency is that it can be used at the hospital, with two patients at once, or at home for those who have a Kinect sensor. The feedback obtained is relevant in all cases and can serve the medical staff to decide the next step of the therapy.

4.2. **The system architecture.** MIRA consists of a Desktop interface application for the therapist, that manages the list of patients and their data, and a series of applications and games mapped on different types of exercises. What we are interested in here is the second part, mainly the interaction between the application and patient during the exercises, provided by the Kinect sensor.

The existent applications can be split into several categories, considering their functionality : applications for user calibration, applications for learning, performing and correcting typical exercises and moves, and games mapped on the same type of exercises. The first two types of applications consist of gradual exercises and moves, destined for beginners, to get used to the system and to performing exercises with MIRA. The last category brings a large diversity of games that keep the patient engaged in a range of moves adapted to the patient's possibility of movement, using the values calculated in the previous exercises.

The applications can also be categorized by the recognition gesture that needs to be done by the patient : the "ψ" pose position, for advanced patients that can raise their hands up, or the wave hand gesture for the patients that are able to perform only limited movements.

From a technical point of view, the applications' visual feedback is based on either virtual or augmented reality, in all types of applications and games developed in C# using WPF, Silverlight and OpenTK.

The types of exercises and games currently implemented by MRA include the basic movements of the arm: extension and flexion of the shoulder and elbow, rotation of the shoulder, dexterity and reflexes of the entire hand, detailed finger motion. Every application has an option for remote monitoring, that would be ready to be activated and used after setting it up in a large medical system.

4.3. **The conceptual model on which exercises are based.** The current implemented exercises were recommanded and validated by medical specialists and constitute the basics of the rehabilitation therapy of the upper limb. For

typical exercises, the patients have to perform the calibration phase, after which they have to touch with their hand (more eaxctly with the hand of their body represented virtually in the application, as tracked by the sensor) some points on the screen, according to the type of exercises (shoulder flexion-extension, rotation and left-right movement, elbow flexion-extension). In the calibration phase, the system computes the user's body dimentions and adjusts the position of the points that mark the wanted movement according to his or her particularities. The exercises are progressively; this means that they start in a simple way and as the level of the exercises increases, the points are placed such that the movement of the hand that should touch them becomes wider and more complex (for example, in a rotation movement, the points are situated in a small circle, and as the level increases, the circle becomes wider till it requires a completely stretched hand). In case of the games (Figure 4 presents some examples), the same principle is valid, but instead of points there are different funny elements, as butterflies, and more complex movements (for example use of fingers in the puzzle game). Also, the system verifies the correct position of the body relative to the initial calibration position, in terms of distance, correct position of the shoulder or elbow relative to the Kinect sensor and to the hand during the exercise. If such a mistake is detected, the user is announced and can not continue the exercise, unless the exercise is done correctly. Of course, a small percentage of error is admitted, easily adjustable according to the disability of the patients, knowing that they might not be capable of a perfect performance at the beginning.

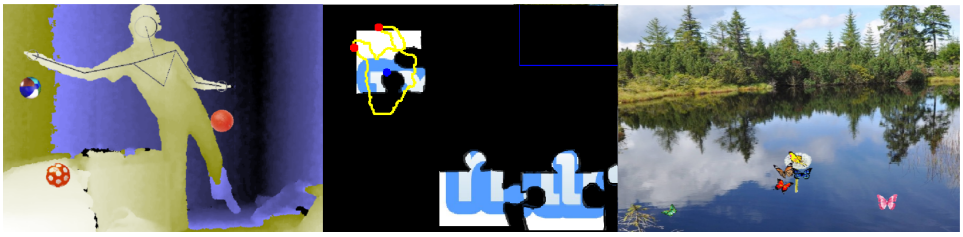


FIGURE 4. Other games and exercises: BeatBalls, Puzzle, Butterfly

The importance of this system is that it implements specific exercises and games, that match the specific requirements of the patients that need rehabilitation therapy. As it is very flexible at the software level, it is the most convenient for rehabilitation centers, as it requires a unique device at a low price with a lot of utilities, always extensible in a short time. This way, when a

particular new case appears, it is easy to implement something adapted to it, without requiring the construction of a new device that would suit the specific need of the patient. As a result, the patients can perform the correct exercises in a fun way, fastening their recovery also because of the increased moral, the exercises adapted perfectly to them in matters of difficulty and fun factor, and because they can perform the exercises any time, regardless of the availability of a therapist.

5. Evaluation of MIRA

The application had great success among the patients that can recover even at home, by playing, and also among medical professionals that gain more time. Testing done on patients at the Rehabilitation Hospital in Cluj-Napoca proved that the applications and games engage the patients in the physical rehabilitation therapy very intensely, and motivates them to perform the exercises while fully capturing their attention. As a result, the improvements were very soon noticed in terms of physical mobility and reflexes, but also in what concerns the mental state of the patients.

The main study subject was a 30 year old male, recovering from shoulder fracture. At first, he was not able to make the "ψ" position, so we skipped to the simpler games that require a wave gesture for the detection of the hand. These games made him very interested, and being concentrated on winning, his hand was making wider moves than usually, pushing himself to something more. He is still recovering, but he seems motivated by MIRA and quickly improving. He is still not yet able to make the "ψ" position, which keeps him from performing many of exercises that would fit him better and help him more, so it would be necessary to create an improved calibration that permits the pose detection in some other position that is easier to perform by a person with a shoulder movement disability.

6. Conclusions and future perspectives

MIRA seems to have achieved its purpose in making the recovery process of the upper limb more efficient and pleasant by using applications and games based on external sensors (mainly the Microsoft Kinect), but future improvements are required in order to make it usable by rehabilitation centers and even at home, personally. The medical advisors in the project suggest that it should be extended to support other disabilities as well, a wider scale of exercises and games, a more complex interpretation of the incoming data

about the pacients' motions, and to offer the possibility for phyicians to create easily new exercises and games using a specially designed framework. Also a different pose detection needs to be implemented, to increase the adaptability of the system to any kind of disability that could benefit from it, but this is easity implemented with the official Microsoft Kinect SDK that has been released in the meantime and, of course, now it is considered for the future implementation of MIRA applications.

But the most important scientific contribution of this system is the fact that it manages to offer a new way of using technology in the benefit of medical health. Moreover, based on similar existing system that proved to be useful, it takes the best of each, creating something with a great potential that is recognised both by specialised medical therapists and by researchers in this field of technology, which agree that even if the classical therapies and the physicist's guidance during the exercises is very important, in many cases when they are not possible or productive, this system would be able supplement or even replace these in a very efficient manner.

7. Acknowledgements

We would like to acknowledge and thank Kinesiotherapist Octavian Olariu, Lect. Dr. Rares Boian, and Prof. Dr. Alexandru V. Georgescu M.D. for their assistance in creating and validating the MIRA project.

References

[1] Ivan Tashev, "Recent Advances in Human-Machine Interfaces for Gaming and Entertainment",International Journal on Information Technology and Security, vol III-3, Union of Scientists in Bulgaria, September 2011, pp. 69–76

[2] M. Law, A. Majnemer, MA. McColl, J. Bosch, S. Hanna, S. Wilkins, S. Birch, S. Telford, D. Stewart, "Home and community occupational therapy for children and youth: A before and after study", Canadian Journal of Occupational Therapy 72(5), 2005, pp. 289–297

[3] Norman Villaroman, Dale Rowe, Bret Swan, "Teaching natural user interaction using OpenNI and the Microsoft Kinect sensor", SIGITE '11 Proceedings of the 2011 conference on Information technology education, New York, USA, 2011, pp. 224–231

[4] Jilyan Decker, Harmony Li, Dan Losowyj, Vivek Prakash, "Wiihabilitation: Rehabilitation of Wrist Flexion and Extension Using a Wiimote-Based Game System", Governor's School of Engineering and Technology Research Journal, 2009, pp. 92–98

[5] D. Rand, R. Kizony, P.L. Weiss, "Virtual reality rehabilitation for all: Vivid GX versus Sony PlayStation II EyeToy", Proc. 5th Intl Conf. Disability, Virtual Reality & Assoc. Tech., Oxford, UK, 2004, pp. 88–94

[6] David Jack, Rares Boian, Alma S. Merians, Marilyn Tremaine, Grigore C. Burdea, Sergei V. Adamovich, Michael Recce, Howard Poizner, Virtual Reality-Enhanced Stroke Rehabilitation, IEEE Transactions on Neural Systems and Rehabilitation Engineering, 9 (3), Sept 2001, pp. 308–318

[7] James Burke, Michael McNeill, Darryl Charles, Philip Morrow, Jacqui Crosbie, Suzanne McDonough, Augmented Reality Game Design for Upper-Limb Stroke Rehabilitation, 2nd International Conference on Games and Virtual Worlds for Serious Applications, March 2010, pp. 75–78

[8] Alan Mozes, Wii-gaming could aid stroke rehab. physical therapy centered around the high-tech games surpassed standard exercises, study finds, HealthDay News, HealthDay – a division of ScoutNews, LLC, February 2010

[9] Peter J. Groen, Douglas Goldstein, Suniti Ponkshe, Marc Wine, "Medical Informatics 20/20: Quality and Electronic Health Records through Collaboration, Open Solutions, and Innovation", Jones and Bartlett Publishers Inc, 2007

Department of Computer Science, Babeş-Bolyai University, 1 M. Kogălniceanu St., RO-3400 Cluj-Napoca, Romania

E-mail address: alina_calin16@yahoo.com

E-mail address: andreicantea@yahoo.com

E-mail address: andrei66_das@yahoo.com

E-mail address: mihaiu_cosmin@yahoo.com

E-mail address: tzutzu@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 4, 2011

REAL TIME SIGN LANGUAGE RECOGNITION USING ARTIFICIAL NEURAL NETWORKS

CORNELIU LUNGOCIU

Abstract. This paper focuses on the problem of recognizing in real time the sign language used by the community of deaf people. The problem is addressed using digital image processing methods in combination with a surpervised learning approach to recognize the signs made by a deaf person. For the recognition step, an artificial neural network will be used. The main goal of the paper is to show that a good performance can be achieved without using any special hardware equipment, so that such a system can be implemented and easily used in real life. An experiment is provided and directions to further improve our work are also emphasized.

1. INTRODUCTION

The goal of our work is to make possible the communication between deaf people and the rest of the world in daily life. According to some unofficial statistics [1], the sign language is known and used in daily communication by deaf people, and persons leaving close to them. This is why people with hearing disabilities are unable to communicate with other hearing people without a translator. For this reason, the implementation of a system that recognize the sign language on daily live (public spaces, banks, post offices, schools, etc) or video chat applications, would have a significant benefic impact on deaf people social live.

In this paper we propose a system that is supervised for recognizing one component of the sign language communication: finger spelling in English. For the supervised learning scenario, an *artificial neural network* will be used.

The rest of the paper is structured as folows. The problem statement, its relevance as well as the difficulties that arise in the study of this field are

Received by the editors: 09-11-2011.

2010 *Mathematics Subject Classification.* 68T05, 65K05.

1998 *CR Categories and Descriptors.* I.2.10 [**Vision and Scene Understanding**]: Shape; I.4.8 [**Image Processing and computer vision**]: Shape; I.5.1 [**Pattern recognition**]: Neural nets .

Key words and phrases. Sign Language Recognition, Artificial Neural Networks, Image Processing.

presented in Section 2. Section 3 briefly reviews the fundamentals of artificial neural networks, also presenting existing approaches to the sign language recognition problem. Our supervised learning based approach, as well as details of our implementation are introduced in Section 4. An analysis of the obtained experimental results is also provided. Section 5 presents the advantages and drawbacks of our system, emphasizing a set of possible further improvements.

2. PROBLEM DESCRIPTION AND RELEVANCE

To better understand the problem, we start by defining the signs in the context of gesture communication and sign language.

A sign is a form of non verbal communication done with body parts and used instead of oral communication, or in combination with it. Most people use both words and signs during communication. A sign language is a language that uses signs to communicate instead of sounds, where the signs are the hand shapes, positions and movements of the hands, arms or body, facial expressions or movements of the lips.

According to the above definition, a sign language has three major components [11]:

(1) *Finger-spelling*: for each letter of the alphabet there is a corresponding sign. This type of communication is used mainly for spelling names.
(2) *Word level sign vocabulary*: for each word of the vocabulary there is a corresponding sign in the sign language. This is the most commonly used type of communication between people with hearing disabilities in combination with the third type.
(3) *Non-manual-features*: facial expressions and tongue, mouth and body position.

As defined above, it is obvious the fact that a sign language is not international. Each language has its particularities regarding the alphabet and vocabulary. In different languages there are different words with different meanings, or words for which there is no equivalent word in other language, etc.

Like the spoken languages and dialects currently used, the sign language has developed differently depending on the region and culture. That is why it would be very useful to have an automatic sign language recognition system.

3. BACKGROUND

In this section we briefly review the fundamentals of *artificial neural networks*, and present several existing approaches to the problem considered in this paper.

3.1 Artificial Neural Networks

Artificial neural networks are emerging as the technology of choice for many applications, such as pattern recognition, speech recognition[9], prediction [8], system identification and control. An *artificial neural network* [13] is a system based on the operation of biological neural networks, in other words, is an emulation of biological neural system. An Artificial Neural Network is an adaptive system that learns to perform a function (an input/output map) from data. Adaptive means that the system parameters are changed during operation, normally called the *training phase*. After the training phase the Artificial Neural Network parameters are fixed and the system is deployed to solve the problem at hand (the *testing phase*).

In a supervised learning scenario, an input is presented to the neural network and a corresponding desired or target response set at the output. These input-output pairs are often provided by an external teacher (supervisor). An error is composed from the difference between the desired response and the system output. This error information is fed back to the system and adjusts the system parameters in a systematic fashion (the learning rule). The process is repeated until the performance is acceptable.

3.2 Sign Language Recognition Systems

The problem of recognizing the sign language in real time was intensively studied in the past in prestigious universities like MIT, University of Milan and the Royal Melbourne Institute of Technology, as well as in private companies such as Fujitsu.

To be able to recognize the signs, a set of measurable features of the body that make difference between signs is needed. The body characteristics that make the difference between the signs are the shape of the hand, the angle from each joint of the fingers and wrist, or arm position and trajectory.

To implement such a system, researches have been conducted in two main directions [10]:

(1) systems that use specialized hardware devices for data acquisition: robotic glove to measure finger and hand joint angles, and various mechanical, optical, magnetic and acoustic devices to detect hand position and trajectory;
(2) systems that use image processing and computer vision techniques to detect the characteristics of the hand in images taken with a video or web camera.

If using imaging systems for detecting body characteristics, data preprocessing is required to obtain numerical features describing the characteristics of the body. This preprocessing step is quite difficult, so the best results were obtained using specialized hardware for data acquisition.

Several existing sign language recognition systems are:

(1) *SLARTI* [10]: for data acquisition it uses a robotic glove and a system based on magnetic fields. For classifying data, a set of neural networks is used, each one trained to classify the sign according to a set of features.
(2) *Glove-Talk* [14]: for data acquisition it uses a robotic glove and for classification uses multiple artificial neural networks.
(3) *Talking Glove* [6]: for data acquisition uses a robotic glove and for classification uses a neural network. This system achieved only finger spelling recognition.
(4) *University of Central Florida gesture recognition system* [3]: uses a webcam and computer vision techniques to collect the data and a neural network to classify shapes. For recognition, this system requires wearing a specially colored glove to facilitate the imaging process.
(5) *ASLR* [12]: uses a webcam and computer vision techniques to collect field data and for classification of signs uses Bayesian learning [15].

The results obtained by each of the above described systems are presented in Table 1.

System	Vocabulary size	Accuracy
SLARTI	52	94%
Glove-talk	203	94%
Talking Glove	28	-
PlaceTypeCentral Florida	8	94%
ASLR	201	17%

Table 1. Comparative results.

Still a comparison between all these systems can not be made, because not all of them have the same purpose, and there is no benchmark database to test all the applications. Each tries to recognize a particular component of sign language. Some intend only to recognize the finger-spelling; others recognize a vocabulary of signs (SLARTI, Glove-talk) or take into account other features of the body such as head position (ASLR).

4. OUR APPROACH

The purpose of this paper is to implement a system that recognizes a component of a sign language, namely: finger-spelling in English. We also aim to develop a simply to use project, that is why, in addition to other implementations, we are trying to achieve this without using any specialized hardware, and without requiring the user to use gloves or other clothing of certain colors.

To make this possible, the project will use a web camera which takes images of the signs made by the user. The images are processed to extract the characteristics necessary for recognition, which are then used as inputs for an artificial neural network that will recognize the sign. The project steps are described in Figure 1.

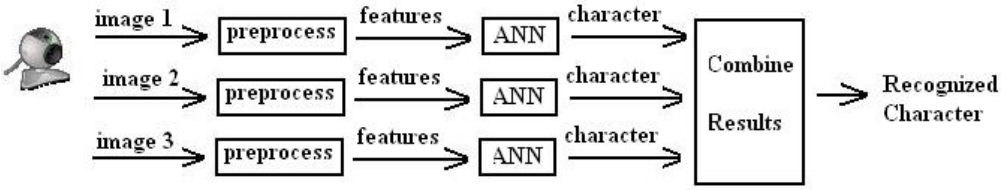


Figure 1. The recognition steps.

To overcome the errors that may occur because of noise from the images, we are using three consecutive images taken from the webcam to recognize a single character. Each image is individually processed and recognized, and then the results are combined using a heuristic method to obtain the final result.

As shown in Figure 1, after taking the image from web camera, three steps are required for recognition of one sign:

(1) *Preprocessing*: uses digital image processing techniques to extract important features from the image, that will be used for recognition.
(2) *Sign recognition*: the data obtained at the first step are used as inputs for a neural network that recognizes the sign.
(3) *Results combination*: the results obtained from these three images are then combined using a heuristic to produce the final result.

The first two steps are done for each of the three images separately. Then the last step is done only one time for all three images.

In the following subsections we will detail each phase of the proposed recognition system.

4.1 Preprocessing

The purpose of this step is to obtain, after the initial image processing, a set of numerical features that uniquely describe a sign. Depending on the goal and available devices, these values may relate to each finger joint angle, hand position and orientation, body position, facial expressions, etc.

Considering the purpose of this work, namely to recognize the fingerspelling in English, detecting the hand shape from the image is sufficient for the recognition of a sign. For this reason, the problem becomes a matter of recognizing objects in an image based on their shape.

The preprocessing is done according to the following diagram:

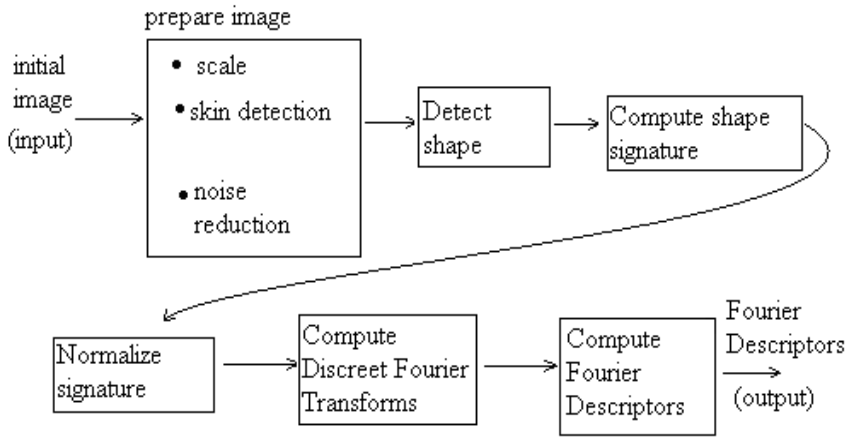


Figure 2. The preprocessing steps.

Image preparation. For this step we propose the following substeps:

(1) *Image scaling* is done to reduce the computational effort needed for image processing.
(2) *Skin detection.* The result of this processing is a binary image in which those pixels that define the hand are colored with white and all the others are black. This processing involves classification of each pixel of the image as part of a human skin or not. There are several techniques developed for skin detection in images, as described in [16]. In this paper, pixels are categorized based on an explicit relationship between the color components red, green and blue. Thus, a pixel is categorized as belonging to the skin if the color components (R, G, and B) meet the following relation:

$R > 95$ and $G > 40$ and $B > 20$ and $\max\{R, G, B\} - \min\{R, G, B\} > 15$ and $|R - G| > 15$ and $R > G$ and $R > B$

(3) *Noise reduction.* After the skin detection, not all pixels are correctly categorized. Noise reduction is meant to overcome these errors, coloring the pixels according to the colors of neighboring pixels [7].

Shape detection. After image processing, the outline of a hand is detected, as a vector of (x, y) coordinates, being the coordinates of the pixels that define the contour of the hand. For this we have used an adaptation of the algorithm described in [17].

Shape signature. Shape signature is a one dimensional vector characterizing the outline of a two-dimensional shape. Four types of shape signatures are described in [4]:

(1) *Complex coordinates*: the vector components are complex numbers. For each pixel of coordinates (x, y) that defines the outline of the shape, the complex number c = x + i*y is saved in the vector.
(2) *Centroid distance*: first, the centroid of the shape is computed, which is defined as the mean of all coordinates of the pixels from the outline. For each pixel from the outline, the distance from its position to the shape centroid is saved in the vector.
(3) *Curvature signature*: for each pixel that defines the outline of the shape the second derivative of the outline is saved.
(4) *Cumulative angular function*: the elements of the vector are the values of the angels between consecutive pixels from the outline.

Each of these variants were analyzed according to the rotation, translation and scaling invariance, and the best results were obtained using the centroid distance, which provides translation and scaling invariance. For this reason, this form signature was used in this paper.

Shape signature is normalized to reduce the size of the signature to a fixed number of values. This is done to reduce the size of the data, thus reducing the computational effort, to allow the uniform treatment of various forms of signatures, and to remove insignificant details form the outline of the shape. There are several ways of normalizing the signature [4]. Here we have used the "*equal point sampling*" method. The equal points sampling method selects candidate points spaced at equal number of points along the shape boundary.

Fourier Descriptors. For a normalized shape signature obtained as described in the previous paragraph, *s(t)*, $t = 0, 1, \ldots N$, the Discreet Fourier Transformations are calculated as indicated in Formula (1).

$$FD_n = \frac{1}{N}\sum_{t=0}^{N-1} s(t)\cdot e^{\frac{-j\cdot 2\cdot\pi\cdot n\cdot t}{N}}, n = 0,1,\ldots, N-1 \qquad (1)$$

The FD coefficients are called Fourier descriptors of the form. The calculation of these descriptors given by the above formula requires a large computational effort. For this implementation we have used the FFD algorithm (Fast Fourier Descriptors), which can reduce the complexity of the calculations if the input vector dimension is power of 2. This is why we chose N = 32.

Fourier transformation concentrates the information from the shape signature in the first terms of the result vector. To normalize the values of descriptors, each term is divided by the first term (called the DC component, which depends only on the position form the image). Since the signature form contains real values, only N / 2 distinct values will be obtained. For this reason, the Fourier descriptors are finally obtained as follows:

$$f = \left[\frac{|FD_1|}{|FD_0|}, \frac{|FD_2|}{|FD_0|}, \ldots, \frac{|FD_{(N-1)/2}|}{|FD_0|} \right] \quad (2)$$

Figure 3 illustrates the preprocessing steps for a sample image.

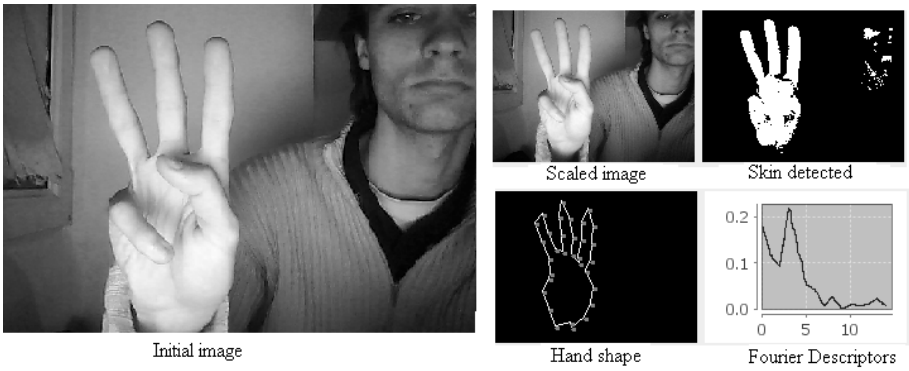


Figure 3. The preprocessing steps

4.2 The Neural Network Model

The Fourier descriptors obtained after the preprocessing step (Subsection 4.1) will be used as input for a neural network that will recognize the shape based on these features. The response of the network is a vector V, with $xv_i \in [0,1]$, $i = \overline{1,24}$, representing the probability for the sign made by the user to be the i^{th} letter from the English alphabet. For one sign made by the user there are three vectors of this form obtained (see Section 4): V^1, V^2, and V3. Combining these vectors we obtain the final result, in the form of a vector V^f, with $v_i^f = v_i^1 + v_i^2 + v_i^3$.

The neural network used is a feed-forward network [10] with layered architecture. It has one input layer, one output layer and one hidden layer, with each layer fully connected with the following layer. The number of the neurons from the hidden layer is the mean between the number of neurons from the input layer and the number of neurons from the output layer. The aggregation function for each neuron is the weighted sum of its inputs and the transfer function is the sigmoid function [15][2].

The network is trained in a supervised learning scenario. The most commonly used learning algorithm for neural networks with layered architectures is the *backpropagation algorithm* [15]. In the current implementation we have used an adaptation of this algorithm, known as *stochastic backpropagation*, which updates the weights after each backward propagation of the error for one training element, not only once for all training set. We have also used the *momentum technique* [15] to avoid local minima in the solution space, and we decreased the *learning rate* during the learning process. The following values

for the parameters were chosen in our implementation: 0.3 for learning rate that linearly decreases to 0.01, and 0.1 for the momentum.

For training a set of 84 elements was used, representing the following letters of the English alphabet: A, B, C, D, E, F, I, K, L, R, U, V, W, X. For each letter there are 6 training instances. To select the training elements, we use the *cross validation* method, thus the data set is randomly partitioned, obtaining a *training* set, and a *validation* set. To avoid the *overfitting* of the network on the training set, every 1000 iterations the initial set is repartitioned. The learning process stops when the network error computed on the validation set, becomes less than a certain value.

4.3 Experimental results

The system was trained to recognize a set of 14 letters from the English alphabet: A, B, C, D, E, F, I, K, L, R, U, V, W, X, and the current results are promising, showing the fact that an approach like this, with a few improvements, could produce good results. A statistic regarding the performance of this system can be found below:

Recognized characters	Recognition accuracy
A, B, C, D, E, F, I, K, L, R, U, V, W, X	80 %

Our approach could be significantly improved if a skin detection algorithm to classify pixels according to the image histogram would be used. This would increase the robustness of the algorithm at the light intensity and color. Also, taking into consideration both the inner and outer contour of shapes would make a better differentiation between the signs, and therefore make the learning process easier.

5. CONCLUSIONS AND FUTURE WORK

We have proposed in this paper a neural network based approach for the sign language recognition. As shown by the experimental results, the solution we have proposed in this paper can be efficiently used in real life situations.

The main advantage of our approach over the other attempts is the fact that it requires no additional hardware equipment or special clothes to recognize the signs. As it happens in most applications that use computer vision techniques to collect data, the main drawback of our solution is the image processing part. The image processing phase of sign recognition process requires a large amount of calculations, which introduces latency in the video stream. The accuracy of the system could be also increased if a more robust skin detection algorithm will be used.

The approach proposed in this paper can be easily extended to recognize the hand trajectory in the image, by this being able to recognize the word level sign vocabulary. Future work may also be done in order to use another

classification model in the supervised learning scenario, such as *support vector machines* [5] or *Bayesian Learning* [15].

References

[1] A.N.S.R, *Manual de Limbaj Mimico-Gestual Românesc*, Editura Mega, 2008.

[2] Dave Andersoan, George McNeill, *Artificial Neural Networks Technology*, Rome Laboratory, 1992

[3] Davis, J and Shas M., *Gesture Recognition,* Technical Report CS-TR-93-11,University of Central Florida, 1993

[4] Dengsheng Zhang, Guojun Lu - *Content-Based Shape Retrieval Using Different Shape Descriptors: A Comparative Study*, ICME, IEEE International Conference on Multimedia and Expo (ICME'01), 2001, pp. 1139–1142.

[5] I. Steinwart, A. Christmann, *Support Vector Machines*, Springer Publishing Company, Incorporated, 2008.

[6] J Kramer and L Leifer. *The Talking Glove: A Speaking Aid for Nonvocal Deaf and Deaf-Blind Individuals*, Proc. of the RESNA 12th annual Conf. (1993) pp. 471-472.

[7] Jerry Huxtable, Java Image Processing Pages: http://www.jhlabs.com/ip/filters/index.html

[8] K. R. Linstrom and A.J. Boye. *A neural network prediction model for a psychiatric application*, International Conference on Computational Intelligence and Multimedia Applications, pp. 36-40, 2005.

[9] M. D. Skowronski and J.G. Harris, *Automatic speech recognition using a predictive echo state network classifier*, Neural Networks, Volume 20, Issue 3, pp:414-423, April 2007.

[10] Peter Wray Vamplew , PhD Thesis: *Recognition of Sign Language Using Neural Networks*, Flinders University of South Australia, 1990

[11] Philippe Drew, Research on Sign Language Recognition: http://www-i6.informatik.rwth-aachen.de/∼dreuw/database.php

[12] Philippe Drew, David Rybach, Thomas Deselaers, Morteza Zahedi, Herman Ney, *Speech recognition techniques for sign language recognition system,* In Interspeech, pages 2513-2516, Antwerp, Belgium, August 2007.

[13] R. Rojas, *Neural Networks: A Systematic Introduction*, Springer, 1996.

[14] S. Sidney Fels and Geo_rey E. Hinton, *Glove-TalkII: A neural network interface which maps gestures to parallel formant speech synthesizer controls*, Transactions on Neural Networks, 9 (1), 205–212. 1998.

[15] Tom M. Mitchell, *Machine Learning*, McGraw-Hill, 1997

[16] Vladimir Vezhnevets, Vassili Sazonov, Alla Andreeva, *A Survey on Pixel-Based Skin Color Detection Techniques*, Graphics and Media Laboratory, Faculty of Computational Mathematics and Cybernetics, Moscow State University, 2002.

[17] Wilhelm Burger, Mark J. Burge, *Digital Image Processing - An Algorithmic Introduction using Java*, Springer-Verlag Berlin, Heidelberg, New York, 2008.

Babe-Bolyai University, Faculty of Mathematics and Computer Science, 1, M. Kogalniceanu Street, 400084, Cluj-Napoca, Romania

E-mail address: lcsi0417@scs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 4, 2011

ACHIEVING REAL-TIME SOFT SHADOWS USING LAYERED VARIANCE SHADOW MAPS (LVSM) IN A REAL-TIME STRATEGY (RTS) GAME

ALEXANDRU MARINESCU

Abstract. While building a game engine in Microsoft XNA 4 that powered a RTS (real-time strategy) tower defense type game, we were faced with the issue of increasing the amount of visual feedback received by the player and adding value to the gameplay by creating a more immersive atmosphere. This is a common goal shared by all games, and with the recent advancements in graphics hardware (namely OpenGL, DirectX and the advent of programmable shaders) it has become a necessity. In this paper we will build upon the shadowing techniques known as VSM (variance shadow map) and LVSM (layered variance shadow map) and discuss some of the issues and optimizations we employed in order to add real-time soft shadowing capabilities to our game engine.

1. Introduction to existing shadowing techniques

Shadowing techniques can be divided in 2 major groups: shadow volumes and shadow mapping. Each has its own advantages and disadvantages and is best suited to a certain scenario. Shadow volumes rely on extruding actual polygons from the geometry of the rendered model, in the direction of the light.

We can see in Figure 1, taken from [6], the actual geometry that is added to the model in order to achieve shadowing. Shadow volumes have the advantage that they provide extremely accurate shadows, regardless of the distance between the shadow caster and receiver. On the other hand, they require a computational overhead on the CPU for the actual computation of the extra geometry and an added fill rate on the GPU for the actual rasterization,

Received by the editors: November 9, 2011.

2010 *Mathematics Subject Classification.* 68U05.

1998 *CR Categories and Descriptors.* A.1 General Literature [**INTRODUCTORY AND SURVEY**]; I.3.7 Computing Methodologies [**COMPUTER GRAPHICS**]: Three-Dimensional Graphics and Realism – *Color, shading, shadowing, and texture* .

Key words and phrases. shadow map, VSM, LVSM, render target, filtering, camera and light space, probability and statistics.

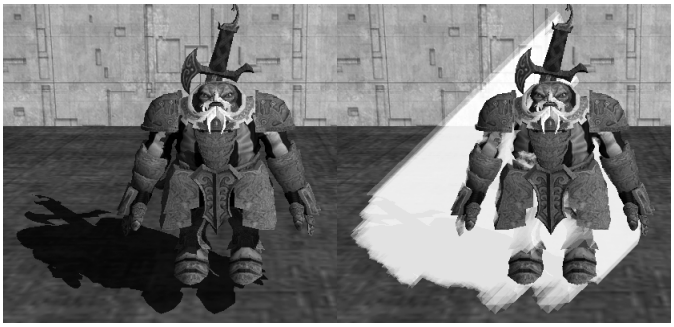


FIGURE 1. Shadow volumes and extruded shadow polygons.

depending on the volume density. In addition to this, shadow volumes are highly dependent on the complexity of the scene and naive implementations are sensitive to non-convex geometry while also suffering from self-shadowing artifacts (shadows cast by the geometry on itself).

Shadow maps utilize what is called "the light's point of view". The difference between camera and light view space is illustrated in Figure 2. The scene on which we perform shadowing is rendered twice. First, it is drawn as seen by the light (which means that we position our camera in the light's position, facing in the direction of the light) and store the depth of each drawn object relative to the light in a render target (we will see that only this depth is needed).

As a side note, one should visualize this whole process in the following way: everything that comes out as output from the pixel shader (GPU program responsible with filling rasterized geometry with color information) is implicitly sent to the graphics card's backbuffer and drawn on screen (the backbuffer can be thought of as a sort of standard graphics output). A render target is simply a region of graphics memory, and sometimes we wish to perform some post-processing before presenting the final scene to the user. This is why we hijack the backbuffer and instead draw to a render target. After we have drawn the distance from each object to the light, we obtain a render target which stores this information as a texture (depth map). Next we draw the geometry as seen by the actual game camera (with whom the user can interact) and take into account the depth map in order to compute the shadowing condition. In the vertex shader, each vertex of the geometry, besides being transformed into camera view space, is also transformed into light view space (so we know where to sample the depth map). The most basic condition for a visible pixel (as seen by the camera) to be in shadow is that its depth relative to the light is greater than the depth sampled from the depth map. Simply

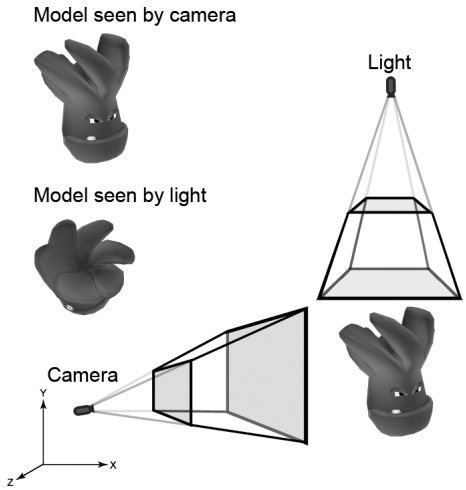


FIGURE 2. Camera view and light view frustums.

put, this states that, if the pixel seen by the camera is further away (from the light) than the corresponding pixel seen by the light, the pixel is in shadow; otherwise it is fully lit (since there is no occluding geometry).

Given the nature of GPUs, which natively operate with 32-bit floating point values, the above inequality $d_{camera} < d_{light}$ will inevitably yield floating point precision errors. These manifest visually as a flickering of the light/shadow boundary and grow in strength as the camera or light position move further apart. This can be solved by adjusting the camera and light view frustums such that the distance from the near to the far planes is as tight as possible. This will increase the actual precision of stored depth values; later on, we will utilize depth values scaled to the range [0..1] and this scaling will be done linearly using the formula:

$$d_{scaled} = \frac{d - nearPlane}{farPlane - nearPlane} \tag{1}$$

It works because, for example, a range near-far of 1-10000 provides us with 1/9999=0.0001 range per unit of distance, while a near-far plane distance of 1-100 provides 1/99=0.01 range per unit of distance, which is 100 times more accurate; also one should not forget that 32-bit floating point values are only accurate up to 10^{-6} which means that differences in depth finer than this minimum value will not be numerically recognized. Other methods for improving shadow quality include increasing the shadow map size, at the cost of increased memory storage and draw time. Nevertheless, there are situations when even

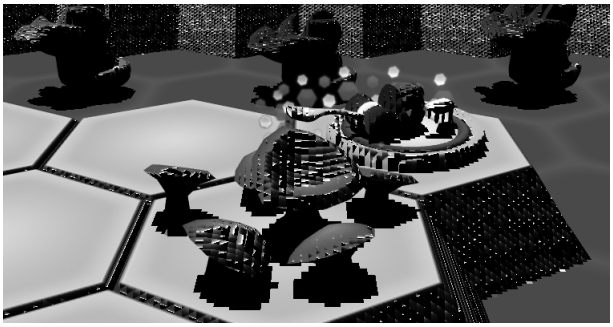


FIGURE 3. Regular shadow map, with point sampling.

a 4096*4096 render target is insufficient (which exceeds sizes supported by most graphics hardware). Still shadow maps are an extremely viable option, especially in our case where the scene is extremely complex and dynamic, due to their relative independence of scene complexity and because we can easily perform a blur post-process (using a Gaussian blur for example) to obtain soft shadows.

2. Variance Shadow Maps (VSMs)

Before delving further into VSM and LVSM shadowing techniques, we should discuss our available options regarding the depth map render target and their numerical implications. XNA provides us with 2 types of render targets: floating-point and pixel formats [5, 7]. Of specific interest to us are the Single (32-bit float format using 32 bits for the red channel), Vector2 (64-bit float format using 32 bits for the red/green channels), Vector4 (128-bit float format using 32 bits for each channel - red, green, blue and alpha), their half precision counterparts (HalfSingle, HalfVector2, HalfVector4) and the Rg32 (32-bit pixel format using 16 bits for the red/green channels) and Rgba64 (64-bit pixel format using 16 bits for each channel). Floating point render targets can store values in the same range as their underlying numeric type (float/half). These can store values in the depth map much more accurately, but most graphics hardware is limited to POINT (nearest neighbor) filtering when sampling from floating point surface formats, thus making shadows appear jagged. A more detailed discussion on texture filtering can be found at [3]. Figure 3 describes the severity of shadowing artifacts on a standard shadow map, using POINT filtering.

We could create our own filtering kernel and perform the actual texture filtering ourselves, but this is costly and we aim to keep any computational

overheads to a minimum, due to the fact that a game engine has to perform many other functions, apart from shadowing.

On the other hand, pixel formats allow efficient filtering, mip-mapping, and anti-aliasing, but any value written by the pixel shader is clamped to the [0..1] interval. So, in order to keep depth values relevant, we need to map them using a depth metric to the range [0..1]. Paper [2] states that any strictly monotonically increasing function will do, but linear depth metrics are the most commonly used; in the actual implementation we have used the metric given by (1).

With standard shadow maps, a texture unit (texel) of the depth map can only store the depth of a single point, so the basic idea behind VSMs is to store a statistical distribution of depths at each texel. In [2], this is done by storing the first and second moments μ_1, μ_2: the mean depth and the mean squared depth. Thus, we can approximate the average of two distributions by averaging the moments (which is done inherently by the filtering hardware of the graphics card). When drawing the scene from the camera's point of view, we will use the sampled moments to compute an upper bound on the probability of a pixel being lit. So, instead of dealing with a simple boolean situation as with simple shadow maps, now we consider that all pixels have a certain probability of being lit/shadowed. Chebyshev's inequality is applied on the mean μ and variance σ^2 derived from sampled moments (denoted M_1, M_2) to provide this upper bound (which is shown in [2] to be exact for the single planar occluder-receiver case).

$$mean = \mu = E(x) = M_1 \tag{2}$$

$$variance = \sigma^2 = E(x^2) - E(x)^2 = M_2 - M_1^2 \tag{3}$$

Let d be the depth of the current fragment seen by the camera, relative to the light, after it has been scaled using the chosen depth metric, then the probability P_{lit} of the fragment being lit is given by:

$$P_{lit} = \{1,\ if\ d \leq \mu\ and\ \frac{\sigma^2}{\sigma^2 + (d-\mu)^2}\ otherwise\} \tag{4}$$

It is worth mentioning that VSMs require double the amount of storage needed for simple shadow mapping techniques because we also store the depth squared in a separate color channel. Still an implementation at this stage suffers from shadowing artifacts (Figure 4), but there is room for improvement.

Chapter 8 of [8] discusses various techniques for improving shadow quality, some of which we have implemented and will be mentioned here. Numeric inaccuracy is still a potential issue; this problem can be dealt almost entirely

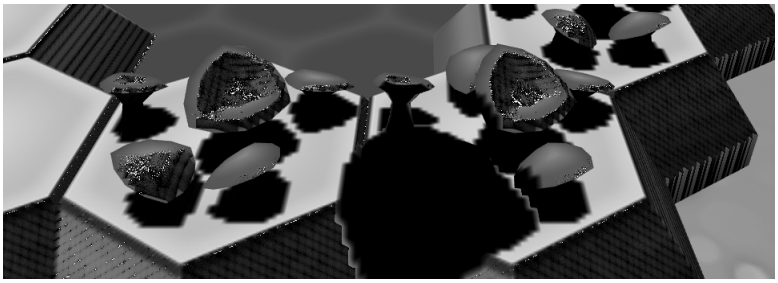


FIGURE 4. Simple VSM implementation.

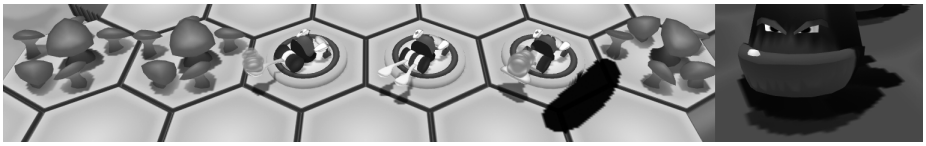


FIGURE 5. VSM with minimum variance clamped to a constant value.

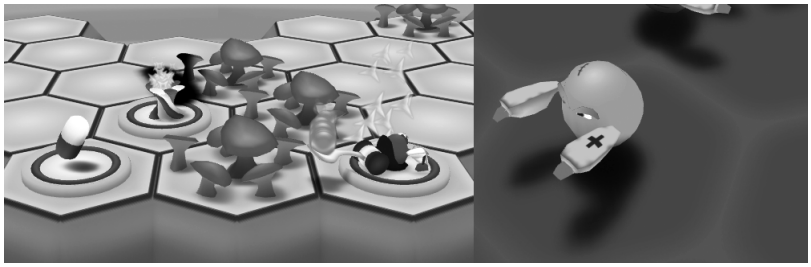


FIGURE 6. Additionally - bilinear filtering, MSAA and Gaussian blur.

if we clamp the minimum variance to a small value before computing P_{lit}; this minimum variance is a constant value, and once tweaked properly can yield significant improvement (Figure 5).

Blurring the shadow map may help hide shadowing artifacts; usually simply enabling mipmapping, trilinear/anisotropic filtering and multisample antialiasing while rendering the shadow map may greatly increase shadow quality at almost no extra cost (Figure 6).

Finally, we can attenuate the light bleeding phenomenon (visible in Figure 4), which is usually seen when the soft edge of a shadow is visible both on the first receiver (as it should be) and, to a lesser extent, on a second receiver (which should be fully occluded by the first). This can be done by modifying

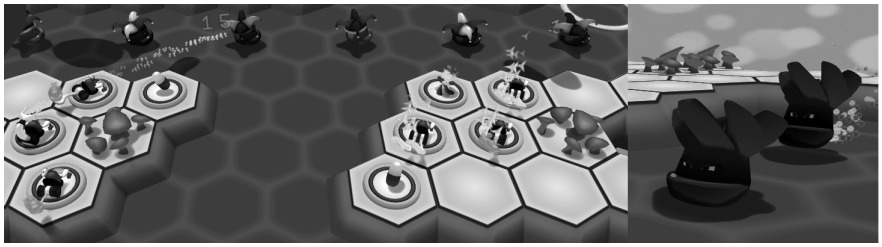


FIGURE 7. Additionally - light bleeding reduction is applied.

P_{lit} before applying lighting calculations, for example, any values below some minimum intensity are mapped to 0, while the remaining values are rescaled to [0..1] (Figure 7).

3. Layered Variance Shadow Maps (LVSMs)

Attempts to further build upon VSM and reduce the light bleeding artifacts are taken in [4], where LVSMs are first introduced. Instead of storing a single depth map with its associated depth metric (1), we partition the light's view space into multiple layers, each with its own depth metric and corresponding depth map. Thus, for each layer L_i covering a depth interval $(nearPlane_i, farPlane_i)$ and an unscaled light distance d, we define the depth metric:

$$d_{scaled} = \frac{d - nearPlane_i}{farPlane_i - nearPlane_i} \tag{5}$$

As stated above, this also has the effect of increasing shadow precision when compared to standard VSM because the near-far planes are now closer together. Furthermore, as shown in [4], when shading a surface it is enough to sample a single layer to determine the shadowing condition: the layer L_j that contains the receiver surface we are shading (at depth d).

We must choose our strategy for splitting the depth range carefully. If we want the best numeric precision overall, a simple uniform split is ideal. If we want to minimize shadow bleeding, we should analyze the distribution of shadow casters and receivers in our scene and choose depth intervals accordingly. Nevertheless, this improvement in quality comes at the cost of additional storage per depth layer.

4. Tradeoffs

As with all graphics applications, a trade-off must be reached between speed and quality and various techniques must be combined in order to attain

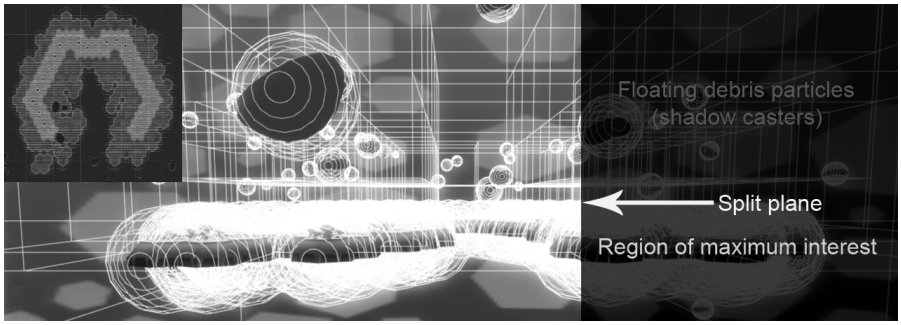


FIGURE 8. OcTree, bounding spheres for entities, and the VSM (from light's perspective).

a pleasing effect. It is a widely accepted fact that there does not exist a general method that works for all situations. A broader range of shadowing algorithms and the environments they are best suited to are outlined in [1].

Consider the following scenario: we need to apply real-time soft shadows to a large scale environment that is specific to a real-time strategy game. The gamer is expected to amass a significant number of units and buildings, which are also coupled to a complex animation system. The terrain (map) is illuminated by a single directional light (no actual position, just light coming from a general direction); this should enable all geometry to cast dynamic soft shadows and augment the gameplay feel. The camera allows for zooming in close to units/buildings and zooming out facilitating strategic thinking; shadow quality should not suffer greatly in neither of these situations. Furthermore, as seen in Figure 8, the shadowing system is expected to take into account for example a particle system that floats above the map and casts dynamic shadows on the terrain.

The fact that a directional light does not have an actual position in the 3D world is an issue which prevents us from constructing its proper View/Projection matrices. We have overcome this by constructing a bounding box around the geometry of the terrain. We compute the light's world position $LightPos$ by backing out in the reverse direction of the light from the centroid of this bounding box as far as it is needed (to encompass the floating shadow casters).

$$LightPos = \frac{1}{8}\sum_{i=1}^{8} BoundingBox.Corner_i - LightDir * Distance \tag{6}$$

For the light projection matrix, we build an orthographic projection (all projection lines are orthogonal to the projection plane), which allows to change the pixel depth calculation from "distance to light position" to "distance to light plane". It may not seem important at first, but this distance can be computed in the vertex shader and afterwards interpolated before passing to the pixel shader, where we output the variance depth map. This is a major gain in speed, because the distance to light position could only be calculated correctly on the pixel shader who is executed once for each fragment, whereas the vertex shader is executed once per-vertex. Development can be further taken in the direction of constructing light view/projection matrices that optimally adapt to the current view frustum of the player camera. Given the fact that all the units and buildings requiring detailed shadowing would be located within a small depth range, and the debris particles (which are also shadow casters) would wander within a much larger distance interval, we implemented LVSMs with 2 layers, one concentrated on the ground region and one that would deal with the floating shadow casters.

5. Conclusions

Finally we would like to summarize the whole shadowing algorithm steps and optimizations which we think can be applied not only to our example, but to any RTS type game:

- Construct optimal light View/Projection matrices only once (when loading map for example);
- Manage geometry through a data structure that enables efficient culling (we use an OcTree for spatial partitioning);
- Render layered variance depth maps into a 1024*1024 Rgba64 render target, with multisampling enabled, considering distance to light plane; we use red/green channels for the first layer and blue/alpha for the second;
- Blur depth map using a two pass Gaussian blur (separated into 2 blur filters, one horizontal and one vertical); we use a blur amount of 1.25;
- Change to camera view/projection and draw scene, enabling bilinear filtering for the shadow map; we clamp the minimum variance to a value of 0.01 and perform shadow bleeding reduction using a power function: $P_{lit} = P_{lit}^{13}$; we also offset P_{lit} with a value of 0.65 to lighten the shadows;
- Applying a post-processing bloom effect yields even nicer results since it also helps hide any remaining artifacts and makes the scene much more organic (Figure 9).

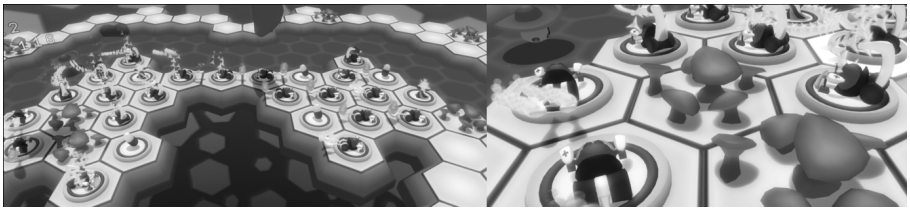


FIGURE 9. Final result, after applying a bloom post-process.

6. Acknowledgements

The author would like to thank Assoc. Prof. Ph.D. Simona Motogna[1] for her invaluable role as mentor for the Imagine Cup 2011 Game Design competition and Lect. Ph.D. Rares Boian[2] for tutoring his graduation thesis. Finally, the author would like to thank his team mates: Bogdan Tanasoiu[3] and Catalin Pintea[4] for creating such a wonderful competition atmosphere.

References

[1] L. Bavoil, Advanced Soft Shadow Mapping Techniques, Game Developer Conference, 2008.
[2] W. Donnelly, A. Lauritzen, Variance Shadow Maps, Proceedings of the 2006 Symposium On Interactive 3D Graphics and Games, 2006, p. 161-165.
[3] S. Hargreaves, Shawn Hargreaves Blog - Game programming with the XNA Framework, http://blogs.msdn.com/b/shawnhar/
[4] A. Lauritzen, M. McCool, Layered Variance Shadow Maps, Proceedings of graphics interface 2008, 2008, p. 139-146.
[5] Microsoft, App Hub - XNA main developer portal for Windows Phone & Xbox 360, http://create.msdn.com/en-US/
[6] Microsoft, DirectX SDK June 2010.
[7] Microsoft, MSDN Library - Documentation for developers using Microsoft technologies and tools, http://msdn.microsoft.com/en-us/library
[8] H. Nguyen, GPU Gems 3, Addison Wesley Professional, 2007.

Department of Computer Science, Babes-Bolyai University, Kogalniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: mams0507@scs.ubbcluj.ro

[1] motogna@cs.ubbcluj.ro
[2] rares@cs.ubbcluj.ro
[3] tbei0012@scs.ubbcluj.ro
[4] ppir0696@scs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 4, 2011

ENHANCING THE STACK SMASHING PROTECTION IN THE GCC

DRAGOŞ-ADRIAN SEREDINSCHI AND ADRIAN STERCA

Abstract. The paper addresses the problem of stack smashing or stack overflows in modern operating systems. We focus on a security solution for this problem, namely compiler generated canary protection and, to be more specific, we consider the Stack Smashing Protector (SSP) present in the most popular C compiler, the GCC. We first analyze the limitations of the GCCs SSP and then present three improvements that will harden the security offered by the SSP making an attackers attempt more difficult. All improvements refer to the most recent version of GCC, 4.6.2.

1. Introduction to Stack Smashing

The stack is an essential part of a running process (i.e. essential for implementing function calls), parts of it, called logical stack frames, correspond to every function which is being executed at a given moment; each time a function is being called, a new stack frame is created, and when the control flow returns to the caller, the related frame is popped out [11]. Smashing the stack is one of the most known and exploited variety of buffer overflow vulnerabilities, in the same time is considered to be the most readily treatable [4]. Even though attacks of this type are reported for over 25 years [12], the security patches, hardware and software protections seem to always be one step behind the latest type of exploit. Smashing the stack became a more popular issue once an interesting article with the title *Smashing the Stack For Fun and Profit* was published by Elias Levy (also known as Aleph One), in the Phrack online magazine [11] in 1996. This publishing did not only set off a multitude of system attacks, but, more importantly, raised the awareness of these problems and triggered the creation of defenses against them. Lets consider a simple C program which has a function named `func()`, that receives two

Received by the editors: November 5, 2011.

2010 *Mathematics Subject Classification.* 68Q01, 68Q01.

1998 *CR Categories and Descriptors.* D.3.4 [**Software**]: Programming languages – *Processors*; K.6.5 [**Computing Milieux**]: Management of computing and information systems – *Security and protection*.

Key words and phrases. stack smashing, canary protection, GCC, buffer overflow.

arguments and has as local variables, a character buffer `buf[80]` and an int. A depiction of the stack after the control flow reaches inside `func()` follows in Figure 1.

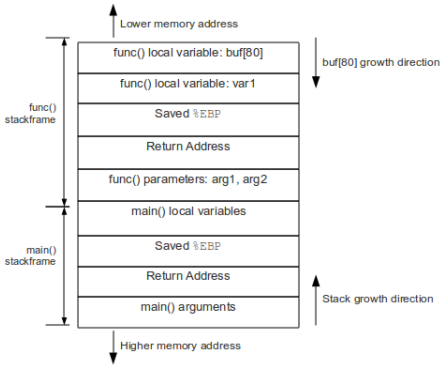


FIGURE 1. Representation of a standard process stack after entering the `func()` function

A simple unchecked `strcpy()` into `buf[80]` with more than 80 bytes will result in writing the excess data on the stack in neighboring places at consecutive higher memory addresses. Given that the stack grows towards lower addresses, overflowing `buf[80]` means altering the data beneath it (higher address), in this case: `var1`, the `Saved %EBP`, then the `Return Address`, and so on, based on how many surplus bytes were supplied. The `Return address` is saved on the stack whenever a function is called, in order to know where the control should return to (i.e. following instruction after the function call), so by changing this address an attacker can trigger the execution of arbitrary code when the vulnerable function finishes [11, 13]. In most cases, the data which is copied into `buf[80]` consists of an shellcode (i.e. machine instructions of code that typically spawns a terminal). By changing the `Return Address` to point inside this shellcode, as soon as the function is finished, the shellcode gets executed provided the privileges with which the process is run under are elevated enough to run those instructions and the address where the shellcode resides was correctly given (usually it is guessed). Other attacks involve changing the `%EBP register` [5], or altering some local data which is needed further in the function [5, 6].

2. Related work. GCC Stack Protection. Limitations

Leaving aside any hardware-enforced protections, which basically consist of a No Execute flag (NX) placed on certain areas of memory marking them as

non-executable, on the Linux family of operating systems, most of the progress that was obtained can be observed at two particular levels:

- GNU Compiler Collection
- Kernelspace

First of all, the compilers defense against exploits we are considering can be divided into two subcategories [1]:

(1) Array bounds checking - the design of the C language makes this method unachievable without compromising either the performance or the compatibility, as an example see the *Bounds Checking Project* [14].
(2) Data integrity checking - the principle is simple: guard the sensitive data using some predefined value and detect changes (corruptions) to the guarded data by checking the predefined value. Achieved through the StackGuard project [2], which was reimplemented in GCC under the name of Propolice, but generically referred to as Stack Smashing Protector (SSP).

Security enhancements were added in several steps to the GNU Compiler Collection, increased abstractization was desired in order to achieve a wider platform independence, and the most notable achievements that are now part of this compiler system are:

- *GNU_STACK* ELF Markings - A stack-controlling ELF program header, used to emulate the behavior of the NX bit. By default this is disabled, no marking being inserted, so, if no GNU STACK note is found, it is assumed that the stack needs to be executable (this might be needed for trampolines) [3].
- *D_FORTIFY_SOURCE* flag - Compiler flag that inserts compile and run-time checks for certain unsafe libc functions. Enabled whenever the optimization level is set to a value bigger or equal with 2 (-O2).
- *Stack Smashing Protector (SSP)* - An extension to the compiler based on the StackGuard security model, that is, guarding the stack frames of a process through a known value placed in certain areas on the stack. The discussion on the protections and flaws of this mechanism is done in the next subsection.

2.1. **SSP Protection Model.** Firstly, it is necessary to point out that the contents of this section refer to version 4.6.2 of the GCC which is the latest version up to the date of this writing. The idea behind the design of this protection lies in the fact that whenever a buffer overflows, all adjacent memory areas up to a certain point are overwritten, and by placing a known value just after a buffer, if an overflow takes place that value will be altered. By doing

a simple check to see if the value remained unchanged it can be deducted whether the guarded buffer has overflown or not. The value pushed on the stack is named a canary, and there are two types that are mainly used:

- **Random Canary** - the value of the canary will be random and the rationale behind this is to avoid the situation in which an attacker knows (or guesses) the value of the canary and fills the overflowing bytes with this value, thus when the canary state is checked it will not be detected as corrupted, but an overflow still took place.
- **Terminator Canary** - a canary with the value `0x000AFF0D` [4]. The significance of this special value stands in the fact that an attempt to overflow the buffer and preserve the canary value unchanged (thus, writing at some point `0x00`) will stop the `strcpy()` C function and, the same effect has `0x0A` on `*gets()` [5], `0xFF` represents EOF, and `0x0D` is the code for carriage return, used as well in order to stop string manipulation or copying functions.

As a side note, other two types of canaries exist: *Null Canary* - holding the null value, which was given up on, and the *XOR Random Canary* - represented by a random number which is xor-ed with the return address saved on the stack [5]. The SSP protection does not resort to any of these two canary types, so we shall not insist on them.

The location where the canary is placed, whatever its type is, is essential for the design of this defense mechanism. Initially - in StackGuard - the canary was placed between the `saved %EBP` and the `Return Address`, but in SSP the canary code resides above both `%EBP` and `Return Address` (remember that above on the stack in this context means being at a lower address because the stack grows from bigger to lower addresses) and below the local variables. In this way, if one of the variables overflows the stack it will inevitably change the canary before the overflow gets to `%EBP` or the `return address` of the function (see Figure 2).

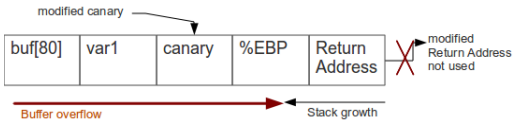


FIGURE 2. Canary placement on the stack

The canary protection code is automatically inserted in an application by the compiler at compile time. For each call to a user defined function two types of actions are performed:

- *Canary Insertion*: The value of the protective canary is inserted in one of the compilation phases, more precisely, at the function prologue level (i.e. entry-code).
- *Verification*: Canary state is verified for consistency before the data that the canary is meant to protect is used, that is, before the stack frame is discarded - action undertaken in the Return Code (i.e. epilogue level).

As said earlier, each protection was meant to stop one particular aspect of an exploit, the canary mechanism's job being to guard the important values of the `return address` and the `saved EBP`. However, together with the canary protection, other two important security enhancements were added to GCC along the way:

(1) *Reordering of function local variables*: If it is written in a local buffer more than its declared size, some other local variables will be altered before reaching the canary; moreover, in some cases the canary is not reached at all, and the intention is to only overwrite some local variable that is used elsewhere - the attacker could be after a format string exploit for example (more can be read in [6] or [5]). Thus, the variables are ordered respecting a single criteria: buffer or not. The buffers are all arranged adjacent to the canary, successively, and the other non-buffer locals will reside above the buffers - in this way if any overflow happens, only other arrays may be corrupted, no local data is altered [7]. A flaw in this type of protection can be observed here in the fact that not only non-buffers should be protected from overflow corruption, but also buffers that might hold important information. By reordering the local data in this way, the buffers are left to overflow into each other, posing a serious security risk to the application.

(2) *Copy of function arguments above the local variable buffers*: Through this, whenever inside a function the arguments are needed, they will no longer be taken from below the `%BP` register, but from an area preceding the local data of this function. Done in the first moments after entering the function, after the creation of the stack frame, each argument is copied from `%EBP+8`, `%EBP+12` etc., on top of the newly created frame. The logic behind this mechanism is the following: during the execution of instructions from the function, its arguments can be corrupted (along with the canary, `%BP` etc.) and then used all throughout that function, even passing them further to other subroutines. The canary state is only checked at the moment when the function returns, which is too late, because probably the execution flow was already corrupted. The parameters are so copied and they will precede the

buffers which could overflow [7], and the possibility of using corrupted data is solved.

Illustration of all the three SSP mechanism for protecting against stack-smashing are presented in Figure 3.

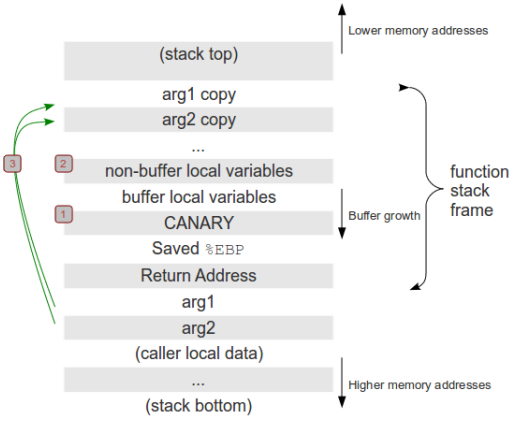


FIGURE 3. SSP Protection added on the stack: canary (1), reordering of local variables (2) and copying of the function parameters (3)

3. IMPROVEMENTS TO THE GCC STACK SMASHING PROTECTION

Since its beginning in 2001, when it was introduced as a FreeBSD GCC protection patch [7], several improvements were consecutively added to SSP to fix newly discovered ways of compromising the stack. The fact that limitations and flaws can still be observed along with the existence for some time of attacks that can bypass SSP [5], stand as proof that the system needs further development and maintenance. Based on the observed limitations and the evolution of SSP, a final version of SSP which would fix all possible security holes seems very hard to find, and, just as it is seen in all aspects of computer security, simply going on and continually hardening and limiting the attacker possibilities might be the best solution.

In this section we present three proposals for increasing GCC stack smashing protection and hardening a hackers effort to compromise the target machine. All our proposals refer explicitly to the most up-to-date, stable version of the GCC compiler which is, at the time of this writing, 4.6.2 [10]. They are intended to complement existing security protection mechanisms present in GCCs SSP (i.e. canary protection, variable reordering, and argument copying).

3.1. **Improvement #1 - Canary state verification not only when the function returns, but also when the function issues a call to another function.** The idea behind this proposal is that the canary signalizes a corruption, but the moment when the value is checked is essential. Currently, this check is performed only in the function epilogue (i.e. when the functions stackframe is released and the control is returned back to the caller through the Return Address) because the initial intention was to provide a protection to the saved `%EBP` and `Return Address` only - which are only used at this point in the execution flow. But, as it can be seen, not only the `Return Address` and the saved `%EBP` are corruptible. Even with the local variables reordered, a possible attacker can still overflow local buffers or variables (i.e. buffers and variables declared by the current function) or structure members can be corrupted by arrays from inside those structures (because inside a structure variables are not reordered by SSP) and these local buffers, variables or structures might be passed in subsequent function calls. The idea behind adding this enhancement lies in the fact that, applications compiled with the SSP protections do not benefit from corruption detection throughout a function code, only at the function end, and at that moment it could be too late, given that local data could have been altered and passed to other function, then used in a chain of exploits.

A method of avoiding the aforementioned problems is to check the canary of the caller function when the execution is about to enter in a new subroutine. In order to do this, there are two ways, or possible points, where the new verification can be placed:

(1) The callers canary could be checked to see if it is unchanged in the prologue of the called function. Doing this check here makes sense because canary operations (i.e. canary placement on the stack and canary verification) are performed in the GCC in a functions prologue and epilogue so in order to keep the injection code clean and compact the check of the callers canary should be performed either in the called functions prologue or epilogue and the called functions prologue is the closest point in execution time to a possible stack corruption in the caller function. However there is an important drawback of doing the callers canary check at this point, in the called function, because the prologue of the called function must be aware of the structure of the callers stackframe (more specifically, the location of its canary) which is normally isolated from the current (i.e. called functions) stackframe.
(2) Another place where the canary can be checked is when a call to a new function is issued, before issuing the call and creating the stackframe for the called function. The advantage of this approach is that the

corruption is detected a bit earlier than in the first case and there is no need of violating the stackframe isolation between function calls, but it also adds complexity to the SSP code making it less compact, as SSP code is normally inserted only in a functions epilogue or prologue, not when a call to a new function is issued.

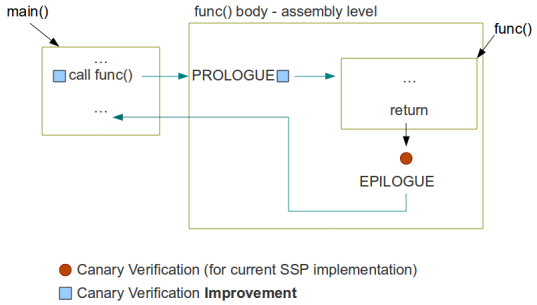


FIGURE 4. The Verification of the canary value done at multiple steps (the rectangles represent the two best positions where the new verification should be inserted)

A note regarding Figure 4: The rectangles represent points where the verification of the canary of the `main()` function are to be done, not `func()`; the canary for `main()` is inserted automatically by the compiler, `main()` being called from the global initializer (constructor) function localized in `__init` (see Section 17.21.5 of [1]).

This improvement mainly aims at doing a preemptive canary corruption detection by simply adding another verification of its state. The overhead should be considered especially for applications that are very modular and for which the flow of execution is expected to pass through many user-defined functions. Particular attention also needs to be given for programs that rely on recursive and highly recursive functions, for which this protection should probably not be used. The targeted set of applications upon which multiple canary verification will have the best impact are the special-purpose daemon services like `crond, ftpd, sshd, init, postfix` or `sendmail` that are critical for any Unix machine, very rarely recursive calls being found in these programs and their security is vital for the system defense. The overhead is comparable with the one for the current SSP canary verification step in the epilogue, so it should not be regarded as an drawback only in special cases as the ones we have mentioned above.

3.2. **Improvement #2 - Canaries for every individual buffer.** One of the obvious flaws of SSP is that the buffers are grouped together, consecutively,

thus enabling situations where one buffer could overflow into another without the canary even being reached and corrupted. This imperfection might seem somewhat harmless in the context where buffers are only used to hold data in transit (even in this case, the data is corrupted when a buffer is overflown into), but it poses a great security risk once the importance of that data increases (i.e. this data is passed to other functions). Take for example the registration of an user account for a fictive application, the password string will be kept in a buffer and if an overflow of other array into that buffer takes place, it will result in storing the corrupted password for that user. A small overflow of only a few bytes (such that the canary is not reached) will pass unnoticed by the current SSP protections, so scenarios like the one described earlier are possible, and they should not be overlooked.

Our second improvement to SSP implies using a different canary for protecting every individual local buffer of a function. In this approach, a canary space (i.e. 4 bytes) is reserved on the stack in front (i.e. higher address) of every local buffer (see Figure 5). Individual canaries for buffers will extend the functionality of SSP from guarding sensitive, controlling data (`%EBP` and `Return Address`) to guarding every vulnerable buffer of the program. This way the task of an exploiter will be even further hardened.

But this added security comes with an important overhead cost. The added overhead cost (i.e. canary generation, storing and verification) of having individual canaries for every buffer is proportional to the number of local buffer variables used in a function and is not negligible for applications dealing with large amounts of data (e.g. network applications, multimedia applications etc.).

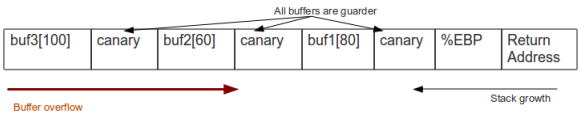


FIGURE 5. A different canary protects every buffer, not just the `Return Address` and `%EBP`

To be noted here that in our 2nd improvement each buffer is protected by its own canary, but still this approach shares the vulnerabilities that a regular canary check has (e.g. the byte-by-byte technique of canary discovery), just that it makes it more difficult to discover the canary value, because now we have several canaries.

Because using unique canaries for individual buffers incurs important additional overhead, this technique can be adapted to offer a good compromise between overhead and protection. For example, the compiler can decide at

compile time, based on its own estimations or based on user input through compiler options, how many canaries to use and in this approach, one canary could be used to protect several local buffers; thus, the protection achieved is less than when using unique canaries for individual buffers, but it has a smaller performance cost.

3.3. **Improvement #3 - Probabilistic canary protection.** The next defense suggestion aims at significantly hardening the process of uncovering the canary value. It does this by two distinct methods, methods which are supposed to be used in conjunction:

- canary position randomization
- probabilistic failure in case of canary corruption detection

The most important protection is given by the second method, the first being just a complementary protection for the second method. Prior to introducing these two methods, we must first briefly discuss the byte-by-byte technique of canary disclosure. This is done in the following paragraph.

The byte-by-byte canary value disclosure technique [8]

This method uses the fact that if child programs are spawned using `fork()`, then the terminator canary will be shared among the parent process and every child that it creates. `fork()`-ing is a crucial feature supported by the operating system, and it is used by an important set of applications that are essential for any Unix distribution. Network deamons usually create handling processes for each connection in this way, and the presence of this flaw in this kind of applications poses a great security risk for a computer exposed to the Internet. The technique is based on the fact that it is easier to guess the value of each single byte of the canary individually (0..255 possible values) rather the value for the whole 32-bit canary at once (0..4294967295 possible values). Consequently, the byte-by-byte technique first overflows the buffer with only 1 byte, thus overwriting the first byte of the canary. When the SSP detects canary corruption it causes the application to crash fast. If on the other hand, the first byte of the canary is overflowed with the correct value, the application will continue its execution (i.e. will not crash). since there are only 256 possible values for a byte, the attacker can try successive values for that byte until the value is found and in the worst case scenario, it needs 256 successive trials until the value of the canary byte is discovered (i.e. the overflowed byte for which the application does not crash). After the first canary byte is discovered the next byte is overflowed in order to discover its value and so on until all 4 bytes of the canary are discovered.

So the whole technique consists of successively trying values for every byte of the canary until the value is found [8], first only one byte is overflowed, then two bytes, and so on. The maximum number of trials an exploiter has to do is $256 \cdot 4 = 1024$, which, in fact, is not even that much for a networking daemon such as `httpd` for example, each individual try taking place in one child of the daemon process. Once the whole canary value was compromised, the protection itself for that process can be considered as nonexistent, the same canary being used for any function and for any child process. The whole process is depicted in Figure 6.

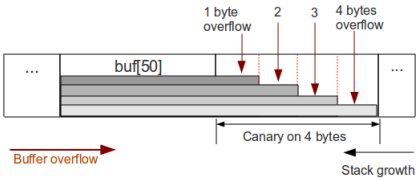


FIGURE 6. Multiple overflows of 1 byte will result in finding the first byte of the canary. Same idea is applied successively 4 times, for each individual byte

Currently, the canary position is fixed, it will reside three stack entries (remember that a stack entry is the same as a word - 4 bytes, on 32 bit architectures) above the `%EBP`, thus leaving 2 empty words, and right next to the first declared buffer. If that buffer overflows even with only 1 byte, the canary value will change, this being the reason why the canary and the buffer are sticked together. But having this specific information at his disposal an attacker can and will use it. By knowing the exact position on the stack of a canary the amount of work someone has to do at finding its value is halved. If the position were to be unknown, he would have to first find that position and than proceed to searching the value.

The first part of improvement no. 3 is canary position randomization. By setting up an area of 10 words space between the `Saved %EBP` and the first buffer and randomizing the position of the canary in this region, the canary value search technique is hardened, first having to find the position of the canary and then proceed to trying the byte-by-byte technique. The overhead added by this step is insignificant, the position could be established using a simple random number generator (e.g. kernels /dev/urandom in Linux). Ten stack positions used for every function in the current stacktrace is, likewise, not an important penalty from the memory point of view.

However, on a closer look, the same method of overflowing byte-by-byte can still be applied to finding the position. So, in fact, in this way only a few

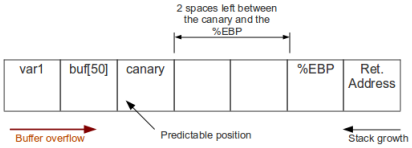


FIGURE 7. The position of the canary is fixed, lying at an predictable address relative to the buffers and the stack frame

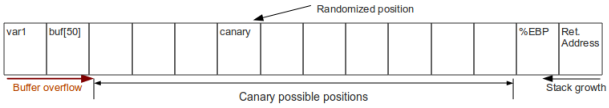


FIGURE 8. The position of the canary is randomized in an area of 10 stack entries

more tries are needed by the attacker, a maximum of 10, added to that of 1024, the results would be unsatisfactory. Therefore, along with randomizing the position of the canary, another method is required to make the process of canary value disclosure more difficult. This method is the *probabilistic failure in case of canary corruption*.

The basis of the probabilistic failure in case of canary corruption method is the observation that in the byte-by-byte technique, an attacker relies on a binary feedback from the SSP code when a byte of the canary is overwritten: failure/crash of the application meaning the value of the overwritten canary byte is incorrect and non-failure (i.e. successful execution) meaning the value for the overwritten canary byte is correct. If we somehow manage to make the SSP code still give an attacker a feedback in a binary form (i.e. failure/non-failure of the application), but the interpretation of a specific feedback is multi-valued and uncertain (i.e. it is not certain that a non-failure means the value for the overwritten canary byte is correct, it might just as well be incorrect), then the byte-by-byte technique will have a hard time guessing the canary. And a good source of uncertainty is randomness. Following this line of thinking, we want to modify SSPs behavior so that it does not crash the application every time the canary or a part of the canary gets corrupted. This is why we call this method probabilistic failure in case of canary corruption. If the application is crashed by the SSP on a random basis when a part of the canary is corrupted, the attacker can no longer infer whether he has guessed that part of the canary or not. The maximum that can be obtained from any canary disclosure protection method is to force an attacker to make 2^{32} = 4294967295 successive trials before he discovers the canary value. And this is exactly the goal of our probabilistic failure method - to force an attacker

to run a number of canary trials close to 2^{32}. Of course, if the canary is not corrupted at all the SSP should not crash the application because something like this will break the compatibility with almost all the applications which exist nowadays. Similarly, if the corruption has moved beyond the canary to the `%EBP` and `Return Address` (i.e. the last byte of the canary is corrupted), the application should necessary crash, because a non-failure will essentially remove all the protection offered by stack canaries.

Thus, our method of probabilistic failure in case of canary corruption will be implemented in the epilogue of a function where canary corruption verification takes place. If no byte of the canary is corrupted, the execution continues normally, failure-free. If the last byte of the canary (i.e. the byte closest to the `%EBP` and `Return address`) is corrupted, the execution crashes. In any other situation (i.e. when a part of the canary is corrupted) our method will compute a failure probability, P_{fail} and based on that, it will crash the application or execute successfully. The algorithm is described bellow:

```
if (last byte of the canary is corrupted)
        fail();
else if (no byte of the canary is corrupted)
        continue;
else
        r = random(0,1);
        P_fail = 1/e^n · r;
        if (P_fail >= 0.5)
            fail();
        else
            continue;
        endif
endif
```

In the above algorithm n is the number of modified bytes in the canary and $random(0, 1)$ is a function which generates a random number between 0 and 1. $fail()$ is a system call that immediately ends the current application returning an error code and *continue* continues the execution of the current application.

By multiplying the random number r with $\frac{1}{e^n}$ the failure probability formula we want to have a higher probability that the program will not crash when more bytes from the canary are corrupted. This is because we want more ambiguity when more bytes of the canary are tried by the attacker. Because of this probabilistic failure method, when an attacker overflows a new byte of

the canary and the program does not crash he can not conclude with certainty that he has discovered that canary byte.

Together with canary position randomization, the probabilistic failure in case of canary corruption method makes it significantly harder for an attacker to discover the canary value. In order to assess the overload induced by our improvement number 3 to the running code, we have implemented this improvement in the GCC's SSP and run 4 tests using 2 test environments. In the first test environment we have a simple program in which the `main()` function calls a simple function, `func()`, several times, first only one time, than ten times, then one hundred times and finally one thousand times. In each test, we have recorded the time (in microseconds) it takes to call function `func()` the specified number of times with the improvement patch enabled and without the improvement patch enabled. The measured times are depicted in the following table.

	Without improvement no. 3 patch	With improvement no. 3 patch
main() calls func() 1 time	2 μs	2 μs
main() calls func() 10 times	3 μs	3 μs
main() calls func() 100 times	4 μs	4 μs
main() calls func() 1000 times	19 μs	23 μs

For the second test environment, we have the same setup as in the first test environment, but this time, each time is being called, function `func()` calls another function `func_1()`. In each test, we have recorded the time (in microseconds) it takes to call function `func()` the specified number of times with the improvement patch enabled and without the improvement patch enabled. The measured times are depicted in the following table.

	Without improvement no. 3 patch	With improvement no. 3 patch
main() calls func() 1 time	2 μs	2 μs
main() calls func() 10 times	3 μs	3 μs
main() calls func() 100 times	5 μs	6 μs
main() calls func() 1000 times	34 μs	38 μs

It can be seen from both tables, that the improvement number 3 does not pose significant overload cost for the running code.

4. Conclusions

This paper presented 3 improvements to the Stack Smashing Protector (SSP) mechanism from the GCC compiler (most recent version being GCC 4.6.2), improvements which would harden the security of applications and will make an attackers job more complicated. The first improvement relied on performing the canary verification step as close as possible to the moment when the actual overflow happened. Improvement number 2 deals with unique canaries for each local buffer of a function and lastly, the most important improvement, probabilistic failure in case of canary corruption aims at making the canary discovering task a difficult one. The GCC patch containing the code for the probabilistic failure method can be downloaded from [9].

5. Acknowledgments

This work was partially supported by CNCSIS-UEFISCSU, through project PN II-RU 444/2010.

References

[1] *** *A GNU Manual*, http://gcc.gnu.org/onlinedocs/gccint/index.html, Free Software Foundation, Inc, 2010.
[2] Crispin Cowan, Calton Pu, Dave Maier, Jonathan Walpole, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, Qiang Zhang and Heather Hinton, *StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks*, pages 63-77, 7th USENIX Security Conference, 1998.
[3] Mike Frysinger, solar, *The GNU Stack Quickstart*, http://www.gentoo.org/proj/en/hardened/gnu-stack.xml?style=printable
[4] Perry Wagle, Crispin Cowan, *StackGuard: Simple Stack Smash Protection for GCC*, GCC Developers Summit, 2003.
[5] Gerardo Richarte, *Four different tricks to bypass StackShield and StackGuard protection*, 2004, http://www.coresecurity.com/files/attachments/StackGuard.pdf.
[6] *** *Format String Exploits*, www.acm.uiuc.edu/sigmil/talks/general exploitation/format_strings/.
[7] Hiroaki Etoh, *GCC extension for protecting applications from stack-smashing attacks*, 2005, http://www.trl.ibm.com/projects/security/ssp/.
[8] Ben Hawkes, *Exploiting OpenBSD*, 2010, http://sota.gen.nz/hawkes openbsd.pdf.
[9] http://www.cs.ubbcluj.ro/ forest/research/grants/pd-444/gcc-ssp
[10] *** *The GNU C Compiler homepage*, http://gcc.gnu.org/
[11] Aleph One, *Smashing The Stack For Fun And Profit*, Phrack #49 article 14, 1996, http://www.phrack.org/issues.html?issue=49&id=14#article.
[12] Michael Dalton, Hari Kannan, Christos Kozyrakis, *Real-World Buffer Overflow Protection for Userspace & Kernelspace*, 17th USENIX Security Symposium, 2008.
[13] J. Reynolds, *RFC 1135 - The Helminthiasis of the Internet*, 1989.
[14] *** *Bounds Checking Project*, abandoned GCC project, http://gcc.gnu.org/projects/bp/main.html

Babes-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania
E-mail address: sdsd0494@scs.ubbcluj.ro
E-mail address: forest@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 4, 2011

A NEW ARCHITECTURE SUPPORTING THE SIZING WINDOW EFFECT WITH STREAMINSIGHT

SABINA SURDU

Abstract. Data stream processing is a new paradigm that emerged in the last years in the field of data management. Dedicated systems are designed to deal with the challenges posed in the process of executing continuous queries over infinite data streams. One of the most sensitive issues in this context is resource usage. The current approaches to minimize system memory and CPU consumption don't take into account the size of a window which is input to a query. In a previous paper we proposed a novel technique, that tackles resource usage by assessing the Sizing window effect. In this paper we propose a new architecture for the Sizing window effect, built on top of a commercially available DSMS, namely the WindowSized architecture.

1. Introduction

For a long time, traditional information processing has been the answer to a large number of data management applications. Conventional database management systems were handling a significant amount of the data processing. These systems store information persistently as finite relations. Queries are issued against the database when needed and their results only reflect the current state of the data. Recently, this processing paradigm proved its shortcomings when it came to handling data that was not static, but continuous, i.e. *data streams*. New challenges are posed in the paradigm of data stream processing. One of the most important ones is resource usage.

The objective of this paper is twofold. First, we propose a new architecture for optimal query processing, using the Sizing window effect, on top of a commercially available Data Stream Management System (DSMS): StreamInsight. We highlight our main contributions, as well as the benefits and limitations of

Received by the editors: November 25, 2011.

2010 *Mathematics Subject Classification.* 68M20, 68N01, 68P01.

1998 *CR Categories and Descriptors.* C.4 [**Computer Systems Organization**]: Performance of Systems – *Measurement techniques*; D.0 [**Software**]: General; H.2.4 [**Information Systems**]: Database Management – *Systems - query processing* .

Key words and phrases. data stream processing, sizing window effect, continuous queries, data stream processing systems, windowsized architecture.

this approach. Second, we motivate the need for continuous query processing in a world where continuous data is present in an ever larger number of fields. Health care monitoring, network monitoring, telecommunications, astronomy or seismography are only some of the fields that are coping with data streams. In the end we conclude on the results of our work and provide future research directions.

2. Data stream processing

A data stream is a sequence of values produced over time by a data source. Most of the applications that cope with data streams are called monitoring applications, because they monitor an arbitrary number of data streams. This class of data-intensive applications scans data streams, performs some processing on read data values and computes desired output in real time. For a detailed description of this field please refer to our paper [13].

Data streams are processed by continuous queries, that perpetually run over time and provide updated results as their underlying input data changes. As the field matures, novel challenges are being posed in the process. One of the most important ones is resource usage.

2.1. **Resource usage in data stream processing.**

Taking into account the temporal dimension adds great complexity to the query processing paradigm. The number of data sources as well as the rate at which data arrives at a system can greatly increase. This can affect the system's ability to output desired results in a real time manner, when processing a great number of complex, continuous queries. Various query optimization techniques (*e.g.* operator scheduling or load shedding [15]) are proposed, in order to tackle resource usage when processing high throughput data streams with continuous queries. To the best of our knowledge, none of them takes into account the size of the input window. In this subsection we will briefly review related work on query optimization and resource usage in data stream processing.

STREAM is a DSMS developed at Stanford, which supports declarative continuous queries over data streams and stored relations [4]. The system tackles performance issues in three ways. First, it eliminates data redundancy, by sharing state within and across query plans. Second, it drops data that will not be used, by exploiting constraints on streams (referential integrity, ordered-arrival and clustered-arrival). Third, the system uses operator scheduling techniques, which aim at minimizing intermediate state [2].

[11] discusses approximation methods from STREAM, which are caterogized into static techniques and dynamic techniques. Load shedding is mentioned in the second category, as the process of dropping tuples from inter-operator queues in the query operator tree, when the queues become too large.

SoCQ, described in [8], is a Pervasive Environment Management System, which provides a declarative way of writing continuous queries against classical data, streams and services. The cited paper describes basic query optimization goals (*e.g.* reduce intermediary relations in queries) and rules (*e.g.* pushing selections down in the query operator tree).

NiagaraCQ introduces a novel approach to executing continuous queries. Apart from executing queries in a change-based manner, every time a tuple appears on a stream, NiagaraCQ introduces timer-based continuous queries, which execute at time instants specified by the user, as [6] shows. This manner of executing queries can greatly impact resource consumption.

Aurora and Medusa are two related DSMSs, described in [18]. The former is centralized, whereas the latter is distributed, using Aurora as a single-site processing engine.

Aurora is based on a combined train and superbox scheduling approach [5]. By train scheduling, Aurora batches multiple tuples as long input trains for operators (boxes in Aurora query diagram) to execute. This saves box call overhead (as the number of box calls decreases) and allows a box to better optimize its execution with an increased number of data elements. By superbox scheduling, Aurora pushes a tuple train through multiple boxes, which avoids the cost of going to disk. Among other optimization techniques, we can enumerate: inserting projections as soon as possible in the query diagram in order to reduce tuple sizes, combining boxes whenever possible to reduce box execution cost and reordering commutative boxes. Apart from these techniques, Medusa encompasses cross-site optimization methods.

Borealis is a second generation DSMS, based on Aurora (for stream processing features) and Medusa (for distributed stream processing capabilities) [12]. This system uses collaborative optimizers on three levels: local, neighbourhood and global. Their purpose is to tackle problems like locating throughput bottlenecks or latency [1].

2.2. **Linear Road.** In order to compare DSMSs, various benchmarking frameworks have been proposed. A key benchmark designed for this purpose is Linear Road [16]. We choose to discuss this benchmark, since our experiments in [14] were conducted on data from Linear Road and we plan to use it in the WindowSized architecture as well.

The Linear Road benchmark is designed to compare performances of Data Stream Management Systems relative to each other and to traditional relational Database Management Systems [3]. Linear Road simulates a variable tolling system in a fictional metropolitan area, Linear City. This urban setting contains 10 parallel expressways, that run horizontally from one another. Each expressway is 100 miles long and is divided into 100 one-mile long segments.

Each expressway has two directions of travelling, Eastbound and Westbound and 4 lanes in each direction (3 travelling lanes and one dedicated lane for entering into and exiting from the expressway). An MIT traffic simulator [17] generates input data for the benchmark, as a set of vehicles that take trips on the expressways. Each vehicle is considered to be equipped wich a device that emits a position report every 30 seconds (describing the time at which the report is generated, the vehicle's speed and location). Based on segment statistics (like average number of vehicles, average speed and proximity of accidents), a variable toll is issued for a vehicle, every time it enters a new segment. The purpose of variable tolling is to reduce expressways congestion at peak traffic periods.

Apart from the continuous queries that compute tolls and detect accidents, the system also supports historical queries issued each time a position report is emitted by a vehicle, with a given probability (like requests for vehicle account balance or travel time predictions). Each query response must fulfil response time and precision requirements specified by the benchmark. A DSMS that implements Linear Road is assigned an L-rating, which represents the number of expressways it can support while still meeting the benchmark requirements. Linear Road is supported by systems like STREAM and Aurora.

3. The Sizing window effect

In [14] we laid the bricks of our proposed novel approach to cope with resource usage in data stream processing, namely the Sizing window effect. Our technique attempts to compute an optimal window size for a given continuous query, thereby placing a minimal upper bound on the resource consumption for that query, as our previous paper shows. We are interested in aggregate queries [10] over windows that are not semantically significant.

3.1. **Formalization.** We will describe the formalization model that we developed in [14]. We consider a discrete, ordered time domain T, as the infinite countable set of time instants, starting from the past and going into the future. For simplicity, we will choose the domain of relative integers to represent T: $T = \mathbb{Z} = \{-\infty, \ldots, -2, -1, \ldots, 3, 4, \ldots, +\infty\}$, where $t_i = i$, adopting the approach described in [9].

Then we define a stream S as follows: $S = \{(s, t) | s \in D^{|schema(S)|}, t \in T\}$, where D is the countable infinite set of constants and $schema(S)$ is the list of attributes in the schema of S. In the rest of this paper we will consider that t_c is the current timestamp.

If we denote by Q a continuous query, then by writing $Q(S, t_c)$ we refer to the result of evaluating Q against stream S, at time t_c. We will denote the

result of executing a query against a stream, at current timestamp, with R_s:

$$R_s = Q(S, t_c) \tag{1}$$

A sliding window over stream S is defined as follows: $SW(S, [t_i, t_c]) = \{(s,t) \mid (s,t) \in S, t \in [t_i, t_c]\}$. The starting point of this window in time is t_i and the endpoint is t_c. For simplifying purposes, this alternative equivalent notation can be used: $SW_{ic}(S)$. Then the result of evaluating Q at time t_c against a sliding window over stream S that starts at t_i and ends at t_c is denoted by $Q(SW_{ic}(S), t_c)$. We will denote the result of executing a query against sliding window $SW_{ic}(S)$, at current timestamp, with $R_{w_{ic}}$:

$$R_{w_{ic}} = Q(SW_{ic}(S), t_c) \tag{2}$$

We will consider the size of a window to be the number of time instants contained by the window. The size of the window $SW_{ij}(S)$ is $t_j - t_i + 1$.

As we already mentioned, we focus on aggregate queries over data streams. This means that at any current timestamp t_c, both R_s and $R_{w_{ic}}$ contain aggregate results from the real numbers domain $\mathbb{R}$. We plan to extend our focus to queries that issue non-aggregate results, as explained in the last section.

We defined the following aggregate distance, for aggregate-queries results, which measures the precision of a windowed result, when compared with a result obtained from a query executed against the stream:

$$distance : \mathbb{R} \times \mathbb{R} \to \mathbb{R}, distance(R_s, R_{w_{ic}}) = |R_s - R_{w_{ic}}|, \tag{3}$$

where R_s and $R_{w_{ic}}$ are both obtained at current timestamp t_c, as explained above. The size of the window from which $R_{w_{ic}}$ is obtained can be deduced from the temporal coordinates t_i and t_c. R_s is considered to be the correct result, since it is obtained when executing the query against all the data from the stream.

3.2. **Experiments.** We performed the experiments in [14] on data from Linear Road, for three continuous queries. We will briefly present the strategy we used and the results we obtained for one of the queries.

In order to find an optimal window size, we started with a window that contained all the data that had arrived on the stream and evaluated R_s, the result of the query executed against this window (*i.e.* against the entire stream), which was the correct result. Subsequently, we constantly decreased the size of the window, up to the point when the result of the query did not fulfil specified precision requirements. At that point, we had just reached the minimal window size, which needed minimal resource usage, while still meeting precision exigencies.

Let us denote this window size by σ. This means the average of the *distance* function between the correct result and the results obtained from

windows of size σ is below a given precision threshold, whereas the average of the *distance* function between the correct result and the results obtained from windows of size $\sigma - 1$ or less is above the given precision threshold. At the same time, a query executed against any window of size greater than σ would use more system resources (CPU and memory to process and to store respectively more tuples).

For query: "Compute the average number of vehicles per time unit that have been travelling on a given segment", we reached an optimal window size of 1000 time instants, for a precision threshold of 1, as Figure 1 shows. This means choosing a window size of 10000 time instants would produce correct results, with considerably increased resource usage. On the other hand, a window size of 100 time instants would use less memory and CPU, but the precision of the result will be degraded.

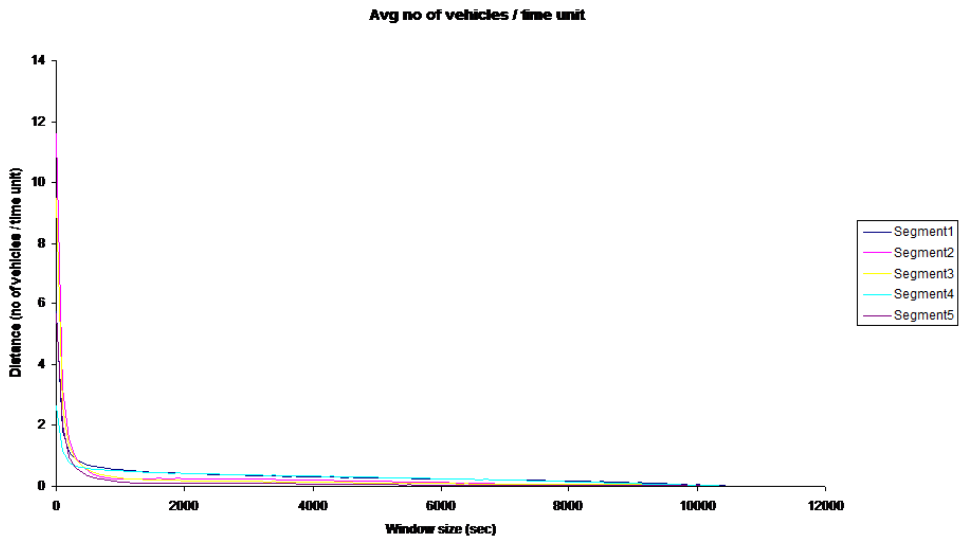


FIGURE 1. Average number of vehicles / time unit

4. WINDOWSIZED ARCHITECTURE

StreamInsight is a DSMS released by Microsoft, as part of SQL Server 2008 R2 [19]. It encompasses a temporal query engine, which executes *standing queries* (Microsoft's terminology for continuous queries) over input data streams, which flow through the system. Data streams are produced by data sources and are feeding input adapters, which translate them into a format

understood by the StreamInsight engine. The results of the standing queries are fed into output adapters. The latter translate these results into formats that the event targets understand.

In order to be able to write any kind of queries and register them on the server, a C# application must be developed. This encompasses continuous queries written in LINQ, which are registered into the StreamInsight server. The complete integration of LINQ queries in the C# application makes it easier to develop a monitoring application on this platform.

On top of this application we are planning the integration of a WindowSizing module, which takes into account the executing query and computes an optimal window size for the query. Figure 2 highlights the main components of such an architecture. This architecture is based on the StreamInsight application architecture approach, composed of event sources, input adapters, standing queries in the StreamInsight query engine, output adapters and event targets [7]. Our contribution is represented by the WindowSizing module, which communicates with the query engine to evaluate the nature of the queries, with input adapters - to change the window size and with output adapters - to obtain query results.

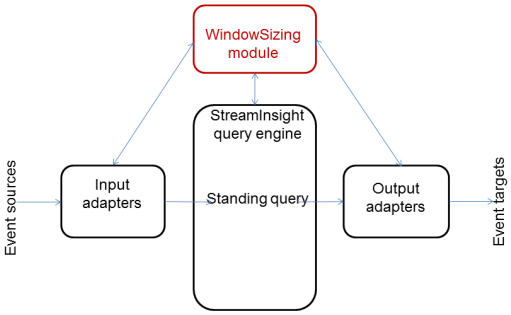


FIGURE 2. WindowSized architecture

The main difficulty is represented by the computation of the correct result. In the experiments we previously described, we bulk loaded the Linear Road data into a commercially available DBMS and computed the correct result on all the available data. But in a DSMS we cannot compute the correct result against an in-memory stored collection of 12 million records. Hence we

propose to choose a maximum window size, depending on available system resources. For this purpose we define the function:

$$maxWindowSize : CPU \times Memory \times TupleSize \times DataRate \to \mathbb{N} \tag{4}$$

This function takes as input the available CPU and memory in the system, the maximum size of the tuples from the input stream and the maximum data rate (number of tuples received every time instant). It outputs a natural number, representing the maximum number of time instants a window can contain, when being processed on the current machine. This is the maximum window size, expressed as number of time instants, accepted by our architecture.

Then we will consider the correct result to be the one obtained when executing the query against a window, whose size is computed by evaluating maxWindowSize. Even though this result is an approximation of the correct, ideal result, it is the best approximation we can obtain when executing continuous queries and will be used as a reference point for queries executed against considerably smaller windows.

The task of establishing the optimal window size is not left for the query developer any more. In cases where the result of a query is more and more accurate as the window size increases, resource consumption also increases. Hence a developer may not choose the best window size when writing continuous queries. By using the WindowSized architecture, the system can better reason about the available resources and can greatly minimize both memory and CPU usage. Another advantage is the use of the Visual Studio IDE and the complete .NET Framework integration, which can greatly reduce development time for monitoring applications, when the .NET platform is already used by an entity [7].

5. Conclusion and future directions

In this paper we proposed a new architecture for processing continuous queries over data streams, using our previously developed Sizing window effect: the WindowSized architecture. We designed a WindowSizing module that computes the optimal window size for a given continuous query, on top of a monitoring application developed with StreamInsight. We highlighted the benefits and limitations of our approach. Our implementation of a prototype based on the WindowSized architecture is currently in progress. We could have chosen any other DSMS. StreamInsight was chosen based on personal technical experience with .NET technologies and the availability of the platform on the MSDN Academic Alliance Software Center.

Future work will revolve around two research directions. First, we will rigorously formalize our WindowSized architecture. Subsequently, we will finalize the development of a prototype based on the WindowSized architecture. Eventually we will conduct a set of extensive experiments using the prototype. As a second objective, we plan to extend our focus of research on queries which issue non-aggregate results. We are working on a distance function for queries that issue collections of tuples as results. We will conduct experiments on these types of queries as well and integrate the newly obtained functionalities in the WindowSized architecture.

References

[1] D. J. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J.H. Hwang, W. Lindner, A. S. Maskey, A. Rasin, E. Ryvkina, N. Tatbul, Y. Xing, S. Zdonik, *The Design of the Borealis Stream Processing Engine*, Proceedings of the 2005 Conference on Innovative Data Systems Research (CIDR), 2005.

[2] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, J. Widom, *STREAM: The Stanford Data Stream Management System*, Technical Report, Stanford InfoLab, 2004.

[3] A. Arasu, M. Cherniack, E. Galvez, D. Maier, A. S. Maskey, E. Ryvkina, M. Stonebraker, R. Tibbetts, *Linear Road: A Stream Data Management Benchmark*, Proceedings of the 30th International Conference on Very Large Data Bases (VLDB '04), pp. 480-491, 2004.

[4] S. Babu, J. Widom, *Continuous Queries over Data Streams*, SIGMOD Rec., vol. 30, n° 3, pp. 109-120, 2001.

[5] D. Carney, U. Cetintemel, A. Rasin, S. Zdonik, M. Cherniack, M. Stonebraker, *Reducing Execution Overhead in a Data Stream Manager*, ACM Workshop on Management and Processing of Data Streams, 2003.

[6] J. Chen, D. J. DeWitt, F. Tian, Y. Wang, *NiagaraCQ: A Scalable Continuous Query System for Internet Databases*, Proceedings of Special Interest Group on Management of Data Conference 2000 (SIGMOD'00), pp. 379-390, 2000.

[7] T. Grabs, R. Schindlauer, R. Krishnan, J. Goldstein, R. Fernández, *Introducing Microsoft StreamInsight*, Technical article, 2010.

[8] Y. Gripay, F. Laforest, J.-M. Petit, *SoCQ: A Framework for Pervasive Environments*, 10th International Symposium on Pervasive Systems, Algorithms and Networks, pp. 154-159, 2009.

[9] Y. Gripay, *A Declarative Approach for Pervasive Environments: Model and Implementation*, Ph.D. Thesis, Institut National des Sciences Appliquées de Lyon, 2009.

[10] J. Li, D. Maier, K. Tufte, V. Papadimos, P.A. Tucker, *Semantics and evaluation techniques for window aggregates in data streams*, In SIGMOD Conference, pp. 311-322, 2005.

[11] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, R. Varma, *Query Processing, Resource Management, and Approximation in a Data Stream Management System*, Proceedings of the 2003 Conference on Innovative Data Systems Research (CIDR), 2003.

[12] E. Ryvkina, A. S. Maskey, M. Cherniack, S. Zdonik, *Revision Processing in a Stream Processing Engine: A High-Level Design*, Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006), 2006.

[13] S. Surdu, *Data stream management systems: a response to large scale scientific data requirements*, Annals of the University of Craiova, Mathematics and Computer Science Series, vol. 38, n° 3, pp. 66-75, 2011.

[14] S. Surdu, V. M. Scuturici, *Addressing resource usage in stream processing systems: sizing window effect*, The International Database Engineering and Applications Symposium ACM International Conference Proceeding Series, pp. 247-248, 2011.

[15] N. Tatbul, *QoS-Driven Load Shedding on Data Streams*, EDBT '02 Proceedings of the Workshops XMLDM, MDDE, and YRWS on XML-Based Data Management and Multimedia Engineering-Revised Papers, pp. 566-576, 2002.

[16] R. S. Tibbetts, III, *Linear Road: Benchmarking Stream-Based Data Management Systems*, M.Sc. Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 2003.

[17] Q. Yang, H. N. Koutsopoulos, *A Microscopic Traffic Simulator for Evaluation of Dynamic Traffic Management Systems*, Transportation Research Part C, vol. 4, n° 3, pp. 113-129, 1996.

[18] S. B. Zdonik, M. Stonebraker, M. Cherniack, U. Cetintemel, M. Balazinska, H. Balakrishnan, *The Aurora and Medusa Projects*, IEEE Data Engineering Bulletin, vol. 26, n° 1, pp. 3-10, 2003.

[19] ***, *Microsoft StreamInsight. Product documentation*, http://msdn.microsoft.com/en-us/library/ee362541.aspx, accessed: 03.11.2011.

Babeş-Bolyai University, Department of Computer Science, 1 M. Kogălniceanu St., 400084, Cluj-Napoca, Romania

E-mail address: surdusabina@yahoo.com