This volume contains 22 carefully selected papers by 59 authors from several universities. These papers were accepted in the main conference sessions of the 11th Mexican International Conference on Computer Science (ENC 2011) held in March 22nd-25th, 2011, in Toluca, State of Mexico.
The papers present the most recent developments in a range of areas related to computer science and applications. They are arranged into 7 thematic fields:

- Data management
- Human-computer interaction
- Distributed systems
- Applied mathematics
- Software engineering
- Industrial applications
- Formal methods and algorithms

We hope that the ENC 2011 proceedings will be fruitful for the scientific community.

ISSN:1870-4069
www.cic.ipn.mx/rcs

SEP

INSTITUTO POLITÉCNICO NACIONAL
"La Técnica al Servicio de la Patria"

Advances in Computer Science and Applications

Claudia Zepeda Cortés, Raymundo Marcial Romero, Abraham Sánchez López
José Luis Zechinelli Martini, Mauricio Osorio Galindo (Eds.)

Vol. 53

RESEARCH IN COMPUTING SCIENCE

ISSN:1870-4069

Advances in Computer Science and Applications

Claudia Zepeda Cortés
Raymundo Marcial Romero
Abraham Sánchez López
José Luis Zechinelli Martini
Mauricio Osorio Galindo
(Eds.)

Vol. 53

RCS
Research in Computing Science

Table of Contents

Índice

Page/Pág.

Software Engineering and Human Computer Interaction

Distributed Systems and Data Management

Applied Mathematics

Industrial Applications and Experience

Formal Methods and Algorithms

Solving the Lookup Problem for Non-member Functions in Presence of Namespaces

Rodrigo Alexander Castro Campos[1], Germán Téllez Castillo[1], Francisco Javier Zaragoza Martínez[2]

[1]Laboratorio de Simulación y Modelado, Centro de Investigación en Computación
Mexico City, Mexico
acastroa09@sagitario.cic.ipn.mx, gtellez@cic.ipn.mx

[2] Departamento de Sistemas, UAM Azcapotzalco
Mexico City, Mexico
franz@correo.azc.uam.mx
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. Current object-oriented programming languages make a distinction between a function tied to a type (member function) and ordinary functions. Since calling member functions requires a different syntax, this lack of uniformity complicates the implementation of generic algorithms. Additionally, while non-member functions may be declared by any user, member functions can be declared only by the author of the type limiting its adaptation to future functional or syntactic requirements. Namespaces were designed to allow independent development of software components. However, invocation of some non-member functions does not work as intended with the prefix qualification imposed by the use of namespaces. In the C++ programming language the lookup rules were thought to allow convenient use of some non-member functions without namespace qualification. However the rules tend to find unwanted names and are considered dangerous for library development. We present a novel set of lookup rules that solves the problem in C++ and similar languages.

Keywords: lookup, namespaces, ADL, library development, software engineering.

1 Introduction

The generic programming paradigm tries to express algorithms with minimal assumptions about data abstractions [1]. Generic functions are usually expressed in terms of types that are not known until the function is actually called. This allows the algorithm to be used with a variety of types not known beforehand by the original author of the algorithm. However, some assumptions must be made; for example, a sorting algorithm may assume operator < is defined so it can be used to compare the elements of the sequence to be sorted.

The assumptions made by the author of a generic algorithm are classified in three groups: syntactic, semantic, and complexity assumptions [2]. The last two are usually

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 3-11

expressed in documentation (manuals, comments in the source code) and failure to meet them results in a wrong behavior of the generic algorithm. However, failure to meet syntactic assumptions may prevent the compilation and use of the generic algorithm even if just a minor adjustment is necessary. When making that adjustment is not possible, the only solution available is to rewrite the generic algorithm.

This is complicated by many programming languages with support for the object-oriented programming paradigm when they include the notion of member functions. In these languages, a member function is a function that can be declared only inside the declaration of the type to which it is tied to and it must be called with a special syntax. Consider the following example in C++ [3]:

```
struct test {
    void member_function( );
};

void non_member_function(test);

test t;
t.member_function( );
non_member_function(t);
```

Given this difference, the writer of a generic algorithm is left to choose which syntactic notation (calls to member or non-member functions) to use in the implementation of the algorithm. Unfortunately, if the former is preferred the algorithm will be unusable for types for which their original implementors did not provide the required functions.

Very few languages try to lessen this problem. For example, the C# programming language [4] allows to declare functions that share the same syntactic properties of member functions outside the declaration of a type. Unfortunately this is not sufficient: many languages (including C++, C#, Java [5], Python [6] and D [7]) restrict the user from declaring constructors (a special kind of member function) and some or all of the overloadable operators as non-member functions.

The problem becomes more noticeable when dealing with operators. Many languages provide a feature to ease the development of independent libraries, usually by allowing the grouping of names inside namespaces. To use a name declared inside a namespace from outside, it is necessary to qualify the desired name with the name of its namespace. Unfortunately namespace qualification defeats the convenience of notation and use provided by operators.

```
namespace ns {
    void f(test);
    void operator+(test);
}

+t;  // error, operator+ not in global scope

ns::f(t);          // qualification required
ns::operator+(t);  // qualification required
```

While it may be argued that operators deserve special treatment or that they could be declared in the global scope, many functions are so common that the lack of an operator to denote them in source code is purely accidental (for example, the square

root function has a well-known mathematical notation but its typing in current keyboards is not easy) and declaring them in the global scope brings back the problem of name collisions namespaces were supposed to solve.

This is also problematic for generic programming: since the actual types used are not known while implementing the algorithm, the namespaces where they (and possibly their non-member functions) are declared are also unknown. This could be partially solved by providing a way to find the namespace of a type and using explicit qualification with such namespace but this not guaranteed to work correctly every time (for some types their non-member functions could be declared in the same namespace but any author may decide to declare them in the global scope). If this path is chosen, it would be necessary to provide a compile-time reflection library to query the information about namespaces and declarations. We consider that the implementation of generic algorithms would become too cumbersome to be practical or intuitive.

2 The C++98 Solution

During the standarization of C++ the problem caused by the interaction of namespaces and operators was discovered [8]. To overcome this, the argument dependent lookup algorithm (ADL) was proposed and later accepted as part of the C++98 standard [9]. ADL is performed for unqualified function calls and, in general, it considers all the functions that are declared in the namespaces where the types of the arguments used in the invocation are declared, plus the functions found by the ordinary lookup. This allows convenient use of operators and other common functions:

```
namespace ns {
    struct array;

    int size(array);
    bool operator==(array, array);
}

ns::array a1;
ns::array a2;

size(a1);          // calls ns::size
a1 == a2;          // calls ns::operator==
```

Unfortunately it was not until several years later that some problems resulting from the interaction of ADL and function templates (the feature used to implement generic algorithms in C++) were identified. In C++, a function template allows the user to write algorithms for types that will be determined until it is used (template instantiation). The following example is a typical implementation of the algorithm that finds the minimum of two values:

```
template<typename T>
T min(T a, T b) {
    return (b < a ? b : a);
}

min(1, 2);      // T deduced as int
```

Unfortunately the names injected by ADL are considered equally important than those found by the ordinary lookup. This is particularly dangerous in the context of unknown types and calls to function templates since ADL may misguide the overload resolution algorithm to select overloads not intended by the implementor of a library:

```
namespace vendor {
    template<typename T>
    void g(T);

    template<typename T>
    void h(T v) {
        g(v);    // wants to call vendor::g
    }
}

struct mine;
void g(mine);    // unrelated to vendor::g

mine m;
vendor::h(m);    // global g called due to ADL
```

Library implementors must protect themselves from ADL by using namespace qualification even inside their own namespaces. This gives little to no advantage in comparison to the identifier prefix alternative used in languages with no support for namespaces (for example, C).

3 Subsequent Attempts for Solution

Designers of new programming languages have tried to avoid coming up with something similar to ADL in an attempt to prevent the problems present in C++. However their designs cannot handle some simple cases that ADL can. The D programming language still needs to rely on member functions and does not not provide a feature similar to the one found in C#. In the Clay programming language [10] this rough translation to C++ is valid:

```
namespace ns {
    struct s;
    void f(s);
}

ns::s v;
f(v);             // calls ns::f
```

However the call will fail if the user overloads the function, even if it could not be called:

```
void f(int);       // hides ns::f
f(v);              // error, ns::f is hidden
```

In the context of C++ a good analysis and some proposed solutions are presented in [11, 12] but in our opinion, the best attempt to fix this problem was proposed by Herb

Sutter in [13] during the works for the new C++ standard, dubbed as C++0x. The proposed change to the ADL algorithm consists on requiring the injected functions to have a parameter of the same type of the argument that causes the injection (ignoring pointer, reference, const, and array modifiers) and in the same position. This change would greatly reduce the number of functions injected by ADL (that currently includes unconstrained function templates, likely to be semantically unrelated), removing most (but not all) sources of surprise. However, what we consider a major issue remains unsolved. Consider the following namespaces:

```
namespace lib1 {
    struct s;
    void f(s);
}

namespace lib2 {
    template<typename T>
    void g(T v) {
        f(v);     // wants to call f via ADL
    }
}
```

Unfortunately, if the user declares a global variable named f, he will inadvertently interfere with lib2:

```
int f;

void g( ) {
    lib1::s v;
    lib2::g(v);   // will find the global f
}
```

It is vital to solve this problem if non-member functions are to become a viable alternative to member functions for expressing functionality of types in languages with features similar to those present in C++. The name lookup problem for non-member functions in presence of templates and namespaces is considered open by Alexander Stepanov who, along with David R. Musser, defined generic programming back in 1987 [14, 15] and are authors of the Standard Template Library [16].

4 Proposed Solution

A key insight for solving this problem is recognizing that ADL ignores the information about the namespace hierarchy as it unilaterally injects a set of functions that are equally important as any others, even those that were in the same scope as the invocation (the ones the user probably intended to call). We propose to apply the following changes to the lookup rules:

1. Strengthen the requirements for ADL

We propose a similar change to the one suggested by Sutter. However, ADL currently searches for functions in the namespaces of the arguments of struct templates and to achieve uniformity the pointer, reference, const, and array type modifiers must be considered template-like type generators:

```
template<typename T>
struct pair {
    T first, second;
};

namespace ns {
    struct inner;

    template<typename T>
    void f(T);
}

pair<ns::inner> p;

f(p);           // calls ns::f via ADL
f(&p.first);    // calls ns::f via ADL
```

We propose to drop this feature completely and require an exact match between the type of the parameter and the type of the argument, ignoring only reference and const modifiers.

2. Continue the search after finding non-candidates

In C++ the search for a function stops after finding one with the desired name, even if it cannot be called. This means the following will not compile:

```
void f( );

namespace ns {
    void f(int);

    void g( ) {
        f( );   // error, f(int) unusable
    }
}
```

The rationale for this rule is documented in [17] and was considered correct in the context of object oriented type hierarchies and the absence of function overloading. The introduction of overloading in C++ casted doubts about the usefulness of this rule but backward compatibility was considered important enough to keep it. We propose to remove this rule and to allow the lookup algorithm to continue searching in enclosing scopes as long as no callable function has been found. This is necessary for applying the third proposed change.

3. Make ADL consider the namespace hierarchy

The set of functions considered by the ADL algorithm is no longer injected as is. The functions are injected from the scope where they are declared into the enclosing scopes until the global scope is reached. We call this process *downward function propagation*.

```
namespace ns {
    struct s;
    void f(s);
}

void g( ) {
    ns::s v;
    f(v);       // ns::f found by propagation
}

namespace mine {
    void f(ns::s);

    void g( ) {
        ns::s v;
        f(v);   // mine::f found
    }
}
```

This effectively turns namespaces into semantic spaces since it is possible to hide any function, even those honored by ADL. This could prove useful for redefining builtin operations if that were to be allowed:

```
namespace safe {
    int& operator*(int* pointer) {
        if (pointer == NULL) {
            exit("null pointer dereference");
        }

        return builtin::operator*(pointer);
    }

    void f(int* p) {
        *p = 1;   // safe::operator* called
    }
}
```

Except for namespaces declared directly in the global scope, nested namespaces are typically coded by the same team of developers. Since function propagation is only performed from nested to enclosing scopes, the propagation of any unwanted names coming from nested scopes is the developer's fault. The propagation remains active only during the resolution of the function call that triggered it.

4. Give preference to declarations found via ADL

The propagated functions found in a given scope have preference over any other declaration:

```
namespace lib {
    struct s;
    void f(s);
 }

int f;

void g( ) {
    lib::s v;
    f(v);   // will always find lib::f
}
```

This guarantees that unknown names will not interfere if the propagation mechanism is sufficient to resolve function calls between two different scopes or namespaces. For namespaces that represent libraries, this allows library developers to safely use declarations contained in another library, isolating them from declarations that may be added in the global scope unless no propagated functions are found and the searched name cannot be resolved locally.

While not always necessary in C++ since user declarations are typically processed after library inclusions, this rule is particularly important for templates as lookup may be deferred for some names until instantiation time and in languages where lookup is not affected by the order of declarations.

5 Conclusions

We presented a novel set of lookup rules that solves an open problem in programming languages design by allowing practical use of non-member functions for declaring the associated functionality of types. This reduces the syntactic variation of programming languages which is particularly important for generic programming, while performing an intuitive name lookup that considers the potential semantic differences between scopes. The lookup rules do not protect the developers from themselves but protect them from declarations contained in other scopes, including the global scope if necessary. These rules were designed as part of the design of a new programming language developed in [18].

References

1. Garcia, R., Jarvi, J., Lumnsdaine, A., Siek, J.G., Willcock, J.: A comparative study of language support for generic programming. Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications. ACM SIGPLAN Notices (2003).
2. Stepanov, A., McJones, P.: Elements of Programming. Addison-Wesley Professional (2009).
3. Stroustrup, B.: The C++ Programming Language. Addison Wesley (2000).
4. C# Language Reference. Microsoft Corporation (2007).
5. Java Language Specification. Sun Microsystems (2005).
6. The Python Language Reference, http://docs.python.org/reference/.

7. D Programming Language 2.0, http://www.digitalmars.com/d/2.0/index.html.
8. Koenig, A.: Reconciling overloaded operators with namespaces, (1995).
9. Programming languages - C++. ISO/IEC 14882 (1998).
10. Clay Programming Language, http://tachyon.in/clay/.
11. Dimov, P.: User-supplied specializations of standard library algorithms, (2001).
12. Abrahams, D.: Explicit namespaces, (2004).
13. Sutter, H.: A modest proposal: fixing ADL (revision 2), (2006).
14. Musser, D.R., Stepanov, A.: Generic Programming. Presented at the First International Joint Conference of ISSAC and AAECC, Roma, Italia (1988).
15. Stepanov, A.: Notes on Programming, (2006).
16. Musser, D.R., Derge, G.J., Saini, A.: Foreword. STL Tutorial and Reference Guide. Addison-Wesley, Boston, MA (2001).
17. Stroustrup, B.: The Design and Evolution of C++. Addison Wesley, Reading, MA (1994).
18. Castro Campos, R.A.: Un modelo de intérprete para un lenguaje de programación de sistemas basado en C, (2010).

Collaborative architecture to support active learning

Gerardo J Alanis-Funes, Luis J Neri-Vitela, Julieta J. Noguez-Monroy

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Ciudad de México
Calle del Puente 222 Col. Ejidos de Huipulco Tlalpan,
14380, México, D.F.
{gerardo.alanisf, neri, jnoguez}@itesm.mx

(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. With the advent of new software tools for collaborative learning, collaborative work and E-Learning either synchronously or asynchronously have become an important part of our lives. File sharing and e-mail communication do not necessarily promote learning. Combining the learning methodologies with the appropriate software tools creates a collaborative architecture that promotes active learning. Hence, a collaborative architecture that integrates and administrates distributed web interactivity tools with learning methodologies is the focus of this paper. The paper reviews related work on active learning, problem based learning (PBL), project oriented learning (POL), collaborative software tools and collaborative virtual environments, highlighting the impact of a collaborative architecture that supports and promotes active learning.

Keywords: Collaborative architecture, Collaborative learning, Active learning, Problem based learning (PBL), Project oriented learning (POL), Long distance education.

1 Introduction

Software development and advances in telecommunications have radically changed the way of interaction between humans [1]. Internet has enabled the teamwork between members that are not necessarily located in the same physical place and, moreover, between members who may have not met personally.

Through collaborative tools like e-mail or virtual meetings, it has been possible to carry out activities beyond the physical place of work or study; some can even be made while moving from one geographical location to another.

It is common nowadays that people use e-mail to send papers, academic files, family pictures, etc. People are more aware about what happens in distant places and may get somehow involved in those remote locations. All these events are opening up endless possibilities for collaboration between people not only from distant places but also with different abilities, skills and situations.

Remote work in several disciplines has become a reality. Staff training has paid good dividends by taking advantage of new tools for remote collaboration. The courses offered by the Internet have proliferated, especially in developing countries, and communicating knowledge has spread among greater number of people and at a

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 13-22

relatively less expensive way. These successful experiences in online personnel training motivate the idea of using these techniques and tools in education.

As is the case for any new human activity, remote collaboration will bring benefits but also problems to face and solve. The application of software and computers for learning is not only limited to sharing files, but it should evolve to become the main tool for interaction and collaboration generating knowledge through the work of all members with a well-defined common goal. The collaborative tools are emerging as an option for these teams to develop their projects as if they were in the same physical space.

2. Active Collaborative Learning

Active learning is about techniques developed such that students do more than simply listening to lecture. Students in an active learning environment perform extra tasks including discovery, processing and application of information [2]. Active learning is essentially the method that seeks to achieve the development of critical thinking skills as well as creative thinking. The learning activity is centered on the learner.

On the other hand, collaborative learning refers to an educational method in which there is a common goal and the students work together in small groups with one purpose: achieving that goal. Within each team students exchange information and work together on specific tasks therefore learning through collaboration. In other words, students are responsible for their own learning as well as the learning of each member of the team.

For the work presented in this paper, two active collaborative learning techniques were considered: Problem Based Learning (PBL) and Project Oriented Learning (POL); both described next.

2.1 Problem-based learning (PBL) [3], [4]

PBL is a learning-centered education method that challenges students to "learn how to learn", working collectively in groups to find solutions to real world problems. The problems are used to stimulate in students the curiosity and motivation to learn the subject matter. PBL prepares students to think critically and analytically, and find and use appropriate learning resources.

The aim of PBL is to provide students with learning skills tools so they become independent learners in their professional life. Hence, the teacher's responsibility is to provide educational materials (scenarios) and guidance that facilitate this type of learning.

Since PBL is based on real world problems that are often difficult and complex, students engage in discussions and critical thinking in every step. Because of this one would expect students acquire the practice and therefore the knowledge to solve future problems.

Basically, 7 steps and 3 documents compound PBL's methodology, as explained next.

Steps:

1. Presentation and reading comprehension stage
2. Defining the problem
3. Brainstorming
4. Classification of ideas
5. Formulating learning objectives
6. Research
7. Presentation and discussion of results

Documents:

1. Tutorial Guide
2. Scenarios
3. Assessment rubrics

2.2 Project-oriented learning (POL) [4]

Project-oriented learning seeks to train students for situations that lead them to not only understand and apply what they have learned in terms of tools to solve problems but also to be able propose improvements applicable to the communities where they operate.

This teaching strategy is an authentic instructional model in which students plan, implement and evaluate projects that have application in the real world beyond the classroom.

When using the project method as a strategy, students stimulate their strongest skills and develop new ones. They are encouraged in the interest of learning and to develop a sense of responsibility and effort.

The results of the learning process of students are not predetermined or fully predictable. This form of learning requires student´s research from many sources and disciplines that are necessary to solve problems or answer questions that are relevant. These experiences in which students are involved to learn to handle and use available resources, such as time and materials, promote they develop and polish academic skills, social and personal nature of the work through school and are situated in a context that is meaningful to them. Their projects often take place outside the classroom where they can interact with their communities, enriching all by the relationship.

The project's work distribution intent is to reduce competition among students, allowing them to collaborate rather than to compete among them. In addition, projects may change the approach to learning, leading from the simple memorization of facts to the exploration of ideas. POL's methodology has fundamental concepts and principles of the discipline of learning and selected topics based on student interest or facility that would lead to activities or results. This strategy may involve some presentations by the teacher and the student-driven work, however, these activities are not ends in themselves, but are generated and completed to achieve some goal or

solve a problem. The context in which students work is, if possible, a simulation of real life investigations, often with real difficulties to be faced with real feedback.

In summary, the organization of learning using the POL method, puts the student in front of a real problematic situation promoting learning that is more connected to the world outside school, which in turn allows one to acquire knowledge in a non-fragmented or isolated environment promoting collaborative work.

3. Collaborative Tools

The set of collaborative tools are software applications that are also called Collaborative Virtual Environments. These are information systems that integrate the work into a single project with many concurrent users at various workstations, connected through a network (Internet or Intranet) [5].

A group of people working together on a common task in the same environment using computers to generate learning is considered to be using CSCL (Computer Supported Collaborative Learning).

CSCL is a pedagogical approach where learning takes place during the interaction of team members using computational media over the Internet. This learning is characterized by the exchange of ideas to build knowledge among participants who use technology as their primary means of communication or as a common resource. Collaborative tools that have been applied to distance education could be classified into three groups:

3.1 Software applications for web conference

Web conferencing applications are useful for live meetings and presentations over the Internet with tools that facilitate the exchange of information, discussion and knowledge in an interactive (synchronous) collaboration, including:

- Data conferencing
- Conferences voice
- Conference video (or audio conference)
- Chat rooms or instant messaging
- Systems to facilitate meetings.

3.2 Learning content management systems (collaborative facilities)

These systems offer a set of functions that support the teaching activities. However, they are limited, as they do not offer full interaction like social networks or virtual rooms do. Nevertheless, these tools facilitate group activities, such as:

- Electronic calendars
- Management of projects

- Flow Control Systems Business
- Knowledge management systems
- Systems of social support networks

3.3 Generic collaborative environments

These are electronic communication tools used to send messages, files, data and documents between team members and facilitate the information sharing (asynchronous collaboration), including:

- Shared Files and Folders
- Discussion forums
- Chat
- Wikis
- Calendars
- Forms
- Track tasks and issues

Collaborative environments are organized into workspaces that activate the tools for collaboration mode according to specific needs. Among the most common tools we can find calendars, folders for storing documents, discussion forums and task management.

4. Collaborative learning architectures

The collaborative tools of the groups previously mentioned can be useful for collaborative work but these do not still provide a comprehensive collaborative virtual environment that promotes teaching and learning. It is necessary to have an architecture that integrates and administrates all these tools and functionalities. Hence, several research groups are developing architectures that can help active learning through collaborative virtual environments. Some representative works of these efforts are shown below.

4.1 Adding Process-Driven Collaboration Support in Moodle [6]

The collaborative facilities of Moodle are limited and do not ensure effective interaction among team members. Moodle in itself has no elements to identify if there exist an interactive collaboration among members, for example, how to ensure that the person interacting is indeed the one registered in the system.

In order to add this collaborative feedback functionality, Moodle involves a learning process administration through collaborative structures called “Learnflow”. This is just an extension of the information systems equivalent known as workflow.

In order to ensure effective collaboration, developers have integrated an oriented to learning activities workflow engine into Moodle. This engine's architecture has a service-oriented approach (SOA) to deal with the interaction of two information systems (Moodle and jBMP Workflow Engine). The functional control passes from Moodle to the workflow engine via Web Services calls.

The purpose of supporting collaboration through processes is to implement structures that manage the flow of activities, i.e. add functionality to keep track of team activities.

4.2 Framework for Collaborative Learning System Based on Knowledge Management [7]

According to the developers of the software, most collaborative virtual learning systems do not consider existing knowledge management for the collaborative process. Indicating that these systems have to support not only learning but also promote a gentle flow of knowledge among collaborative team members. The proposed framework integrates the technology of knowledge management into a system of collaborative learning. According to this, collaborative learning process via the Internet or Intranet can be more effective.

The framework is divided into five basic modules: Communication Tools, Knowledge Management, Workflow Management, Other Tools and an Area of Interaction / Learning. These are briefly described below.

Communication Tools. Due to the nature of the system it must provide some communication tools that meet the needs of information exchange, coordination of activities and scope of collaborative learning arrangements. These tools include, e-mail, wikis, blogs, BBS, Chat, videoconferencing, etc.

Knowledge Management. In collaborative work there is a massive amount of materials such as documents, records, web sites, repositories, knowledge flow, etc. In order to be able to mind them, one should count on tools for storing and access.

Workflow Management. This module administrates the tasks required for the teams to achieve all cooperative learning activities.

Other Tools. These are tools that assist in the analysis, semantic search and access in the material used for collaboration.

Interaction Area / Learning. It is the main area of knowledge sharing during the collaborative learning process. A typical collaborative learning process consists of four steps: question, discussion, verification and conclusion.

The developers created a prototype technology based on Service Oriented Architecture (SOA). They use J2EE technology with an application server (Jakarta Tomcat Server or JBoss), Java Server Pages (JSP), Servlets, JavaBeans and SQL 2000. Users enter the system-using HTTP through any browser such as Microsoft IE.

5. Problem

During the development of education using active learning techniques that are based on collaborative work "face to face meetings" for working sessions are necessary. Meetings are meant to share research advances, discuss the work that needs to be done and assign tasks for each member of the team based on published work to date.

However, in-person meetings are not always plausible either because there is conflict of agendas or because participants are in different and far away locations. Therefore, finding a CSCL architecture where students can interact continuously, regardless of location and also share documents if necessary, would open a great deal of opportunities for both academia and industry. In a scenario like this the teacher or group leader could have access to that system to monitor student work, communicate with them and participate in the discussion providing, where necessary, guidelines for the proper development of the project or problem depending on the case.

Even though there are several architectures for CSCL, as we have reviewed, they are still not suitable for active learning techniques such as PBL and POL. In this sense, the architecture proposed in this paper aims at facilitating the implementation of these teaching techniques.

6. ActivColLearn: An Active Collaborative Learning Architecture

For the implementation of a virtual collaborative work environment for active learning, we have initially proposed the design of an architecture based on SOA. For this, we have considered the implementation of three servers: a portal server, workflow server and repository server.

The portal server is the platform that hosts and serves a Web interface, publishing and managing content as well as adapting the view of the presentation.

The workflow server allows to structure tasks specifying how they should be performed, what their chronological order should be, how they are synchronized, how the information supporting the tasks flows and how the performance is tracked down.

The repository server accommodates the different scenarios developed by teachers as well as the activities performed and results developed by the students.

For clients considering the implementation of Web clients, smart clients (applications that allow you to work while disconnected from the server) and Web services for use by external applications are needed. A suggested architecture diagram can be seen in Figure 1.

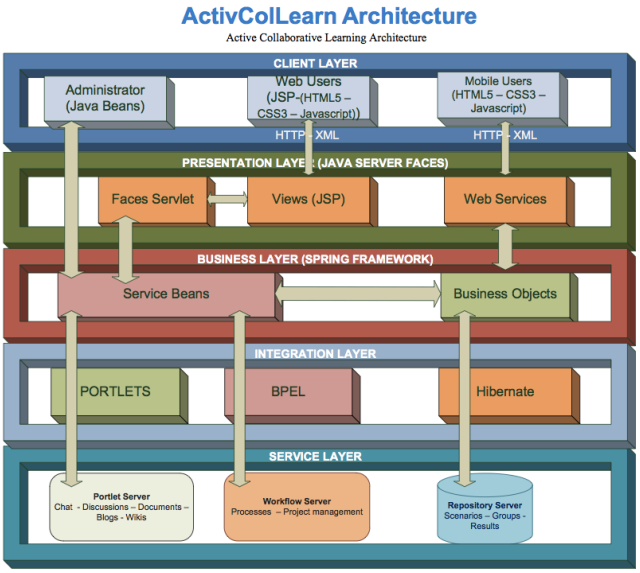


Fig. 1. Architecture Diagram

The proposed architecture is intended to provide the necessary services for remote collaboration among members during active learning work and monitoring tools to help teachers.

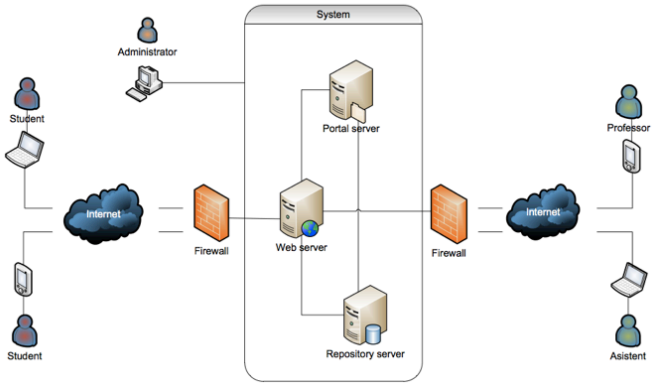


Fig. 2. Interaction Diagram

As shown in Figure 2, this architecture allows the implementation of a system that integrates collaborative, teaching, learning and monitoring activities.

The modules constituting the system will be:

1. Communication Module
    a. Collaborative tools (e-mail, Wikis, Blogs, etc.).
2. Teaching Module
    a. Problem Scenarios
    b. Reviews
    c. Intelligent system integration
3. Learning Module
    a. Managing workspaces
    b. Intelligent system for monitoring of cooperation
4. Project Management Module
    a. WBS
    b. Calendar of meetings and deliveries
    c. Workflow
5. System Management Module
    a. Defining Workspaces

Figure 3 shows a system block diagram for the proposed architecture:

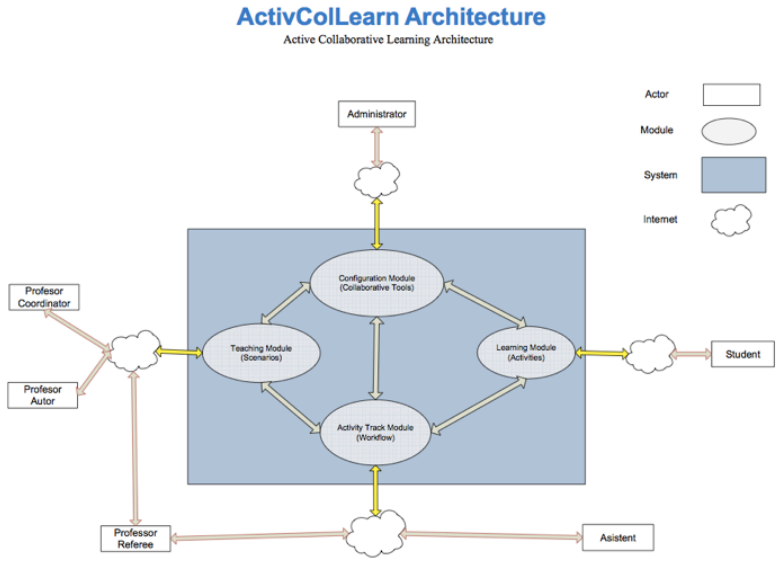


Fig. 3. System Block Diagram.

7. Discussion

Implementation of CSCL to education has not yet achieved a sufficient penetration in the learning process. One possible explanation for this fact could be that CSCL tools

are still not used at their fullest capability in terms of substituting a classroom teaching, as opposed to as an endorsement of such education. If properly applied as a support for classroom teaching, as well as with active learning activities, we believe that learning outcomes can be improved. We also consider that due to the nature of active learning techniques, they can be well adapted for the implementation of the proposed architecture.

8. Future Work

At present we have finished the basic research work and we are implementing this prototype system to work with focus and control groups of students. In a future research, we will try to improve the system and will use it in real active learning practices to assess their implications and impact on the actual student learning. In addition, we are currently working on integrating new technology and concepts into the system, such as M-Learning and Web 2.0, in order to make it accessible from different mobile devices as tables and smartphones.

References

1. Susuki, J., Yamamoto, Y.: Leveraging Distributed Software Development. Computer, 59-65 (2004).
2. Vicerrectoría Académica del Instituto Tecnológico y de Estudios Superiores de Monterrey http://www.sistema.itesm.mx/va/dide2/tecnicas_didacticas/
3. Vicerrectoría Académica del Instituto Tecnológico y de Estudios Superiores de Monterrey http://www.sistema.itesm.mx/va/dide2/tecnicas_didacticas/abp/qes.htm
4. Sola C.: Aprendizaje basado en problemas. De la teoría a la práctica. Trillas, México D.F (2005)
5. Ecmware. Collaborative Software. Competitiveness & Innovation. http://www.ecmware.com/respuestas/glosario/software_colaborativo.html.
6. Perez-Rodriguez, R., Caeiro-Rodriguez, M., Anido-Rifon, L.: Adding Process-Driven Collaboration Support in Moodle. In: 39th ASEE/IEEE Frontiers in Education Conference, pp. 18 – 21. IEEE Press, San Antonio, TX (2009)
7. Zhao, R., Zhang, C.: A Framework for Collaborative Learning System Based on Knowledge Management. In: First International Workshop on Education Technology and Computer Science (2009)
8. Dong, L., Marshall, J., Wang, J.: A Web-based Collaboration Environment for K- 12 Math and Science Teachers. In: 39th ASEE/IEEE Frontiers in Education Conference, pp. 18 - 21, IEEE Press, San Antonio, TX (2009)

An IDE to Build and Check Task Flow Models

Carlos Alberto Fernandez-y-Fernandez[1], Jose Angel Quintanar Morales[2], and Hermenegildo Fernandez Santos[2]

[1] Instituto de Computación, Universidad Tecnológica de la Mixteca, México
[2] Lab. de Inv. y Des. en Ing. de Soft., Universidad Tecnológica de la Mixteca, México
{caff, joseangel, ps2010160001}@mixteco.utm.mx
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. This paper presents the Eclipse plug-ins for the Task Flow model in the Discovery Method. These plug-ins provide an IDE for the Task Algebra compiler and the model-checking tools. The Task Algebra is the formal representation for the Task Model and it is based on simple and compound tasks. The model-checking techniques were developed to validate Task Models represented in the algebra.

Keywords: lightweight formal specification; software modelling; model-checking.

1 Introduction

There has been a steady take up in the use of formal calculi for software construction over the last 25 years [1], but mainly in academia. Although there are some accounts of their use in industry (basically in critical systems), the majority of software houses in the "real world" have preferred to use visual modelling as a kind of "semi-formal" representation of software. A method is considered formal if it has well-defined mathematical basis. Formal methods provide a syntactic domain (i.e., the notation or set of symbols for use in the method), a semantic domain (like its universe of objects), and a set of precise rules defining how an object can satisfy a specification [11]. In addition, a specification is a set of sentences built using the notation of the syntactic domain and it represents a subset of the semantic domain. Spivey says that formal methods are based on mathematical notations and "they describe what the system must do without saying how it is to be done" [10], which applies to the non-constructive approach only. Mathematical notations commonly have three characteristics:

- conciseness - they represent complex facts of a system in a brief space;
- precision - they can specify exactly everything that is intended;
- unambiguity - they do not admit multiple or conflicting interpretations.

Essentially, a formal method can be applied to support the development of software and hardware. This paper shows an IDE for modelling and checking task flow models using a particular process algebra, called Task Algebra, to characterise the Task Flow models in the Discovery Method. The advantage is that this will allow software engineers to use diagram-based design methods that have a secure formal underpinning.

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 23-33

1.1 The Discovery Method

The Discovery Method is an object-oriented methodology proposed formally in 1998 by Simons [8, 9]; it is considered by the author to be a method focused mostly on the technical process. The Discovery Method is organised into four phases; Business Modelling, Object Modelling, System Modelling, and Software Modelling (Simons, pers. comm.).The Business Modelling phase is task-oriented. A task is defined in the Discovery Method as something that "has the specific sense of an activity carried out by stakeholders that has a business purpose" (Simons, pers. comm.). This task-based exploration will lead eventually towards the two kinds of Task Diagrams: The Task Structure and Task Flow Diagrams. The workflow is represented in the Discovery Method using the Task Flow Diagram. It depicts the order in which the tasks are realised in the business, expressing also the logical dependency between tasks. While the notation used in the Discovery Method is largely based on the Activity Diagram of UML, it maintains consistently the labelled ellipse notations for tasks.

1.2 The Task Flow models

Even though Task Flow models could be represented using one of the process algebras described above, a particular algebra was defined with the aim of having a clearer translation between the graphical model and the algebra. One of the main difficulties with applying an existing process algebra was the notion that processes consist of atomic steps, which can be interleaved. This is not the case in the Task Algebra, where even simple tasks have a non-atomic duration and are therefore treated as intervals, rather than atomic events. A simple task in the Discovery Method [8] is the smallest unit of work with a business goal. A simple task is the minimal representation of a task in the model. A compound task can be formed by either simple or compound tasks in combination with operators defining the structure of the Task Flow Model. In addition to simple tasks and compound tasks, the abstract syntax also requires the definition of three instantaneous events. These may form part of a compound task in the abstract syntax.

2 The Task Flow metamodel

2.1 The Task Algebra for Task Flow models

The basic elements of the abstract syntax are: the simple task, which is defined using a unique name to distinguish it from others; ε representing the empty activity; and the success σ and failure φ symbols, representing a finished activity. Simple and compound tasks are combined using the operators that build up the structures allowed in the Task Flow Model. The basic syntax structures for the Task Flow Model are sequential composition, selection, parallel composition, repetition, and encapsulation. The algebra definition is shown in table 1.

$Activity ::= \varepsilon$		– empty activity
	$\|\sigma$	– succeed
	$\|\varphi$	– fail
	$\|Task$	– a single task
	$\|Activity; Activity$	– a sequence of activity
	$\|Activity + Activity$	– a selection of activity
	$\|Activity \parallel Activity$	– parallel activity
	$\|\mu x.(Activity; \varepsilon + x)$	– until-loop activity
	$\|\mu x.(\varepsilon + Activity; x)$	– while-loop activity
$Task ::=$	$Simple$	– a simple task
	$\|Activity$	– encapsulated activity

Table 1. abstract syntax definition

A task can be either a simple or a compound task. Compound tasks are defined between brackets '{' and '}', and this is also called encapsulation because it introduces a different context for the execution of the structure inside it. Curly brackets are used in the syntax context to represent diagrams and sub-diagrams but also have implications for the semantics. Also, parentheses can be used to help comprehension or to change the associativity of the expressions. Expressions associate to the right by default. More details of the axioms can be seen in [6].

2.2 Model-checking

A set of traces is the trace semantic representation for a Task Flow Diagram. The verification of the diagram may be made in different ways. The simplest operations could be performed by set operators but more operations may be applied over the traces using temporal logic. Temporal logic has being extensively applied with specification and verification of software. The set of traces, obtained from a task algebra expression, may be used to verify some temporal and logical properties within the specification expressed by the diagrams. For this reason, a simple implementation of LTL was built. This LTL implementation works over the trace semantics generated from a Task Algebra expression. Because the trace semantics represent every possible path of the Task Flow diagram expressed in the Task Algebra, it is straightforward to use LTL formulas to quantify universally over all those paths. In this section, some examples using Linear Temporal Logic (LTL) are presented, to illustrate the reasoning capabilities of the LTL module. LTL is a temporal logic, formed adding temporal operators to the predicate calculus. These operators that can be used to refer to future states with no quantification over paths. In addition, a CTL application was built to test CTL theorems against expressions in the task algebra. In this case, the application has to transform the traces in a tree representation before applying the expression. While LTL formulas express temporal properties over all undifferentiated paths, Computational Tree Logic (CTL) also considers quantification over sets of paths. CTL is a branching-time logic [5] and theorems in this logic may also

be tested against a set of traces obtained from a task algebra expression, in the same way that LTL theorems were tested above.

3 A tool for formal specification of Task Flow models

3.1 Analysis of Integrated Development Environments (IDE)

Through a search in surveys and articles published in digital media, Eclipse is chosen as the top two open source IDEs best positioned among developers. However, Eclipse showed a better performance due to the existence of robust tools for the development of plug-in, as it has with the Plug-in's Development Environment (PDE) which provides tools to create, develop, test, debug, build and deploy Eclipse plug-ins, modules and features to update the sites and products Riched Client Platform (RCP). PDE consists of three elements:

- PDE User Interface (UI) for designing the user interface;
- PDE Tools Application Programming Interface (API Tooling) useful pieces of code to develop applications;
- PDE Builder (Build), manager responsible for the administration of the plug-in.

Besides all this, the GMF frameworks (Graphic Modeling Framework - Framework for graphic editing) and Eclipse Modeling Framework (Eclipse Modeling Framework, EMF), which facilitate the construction. We can get a highly functional visual editor using EMF to build a structured data model enriched by GMF editors. The main advantage is that being all development based on building a structured model, the time spent on the maintenance phase will be substantially reduced.

3.2 The architecture of the task model tool

As mentioned above, our general architecture is based on the Eclipse framework. The first component is able to model Task Flow diagrams and translate them into a metamodel formed by Task Algebra expressions. The resultant file containing the metamodel is used by the Task Algebra compiler in order to generate the trace semantics.

In addition, the other component in Eclipse has the responsibility to receive LTL and CTL queries. The queries are sent to the relevant model-checking tool. Textual results are returned by the tool and have to be interpreted by LTL/CTL Eclipse plugin. Figure 1 shows the general dependency between the components of our project.

4 Formal modelling made easy

4.1 Design of the structured model

Once identified the use cases, classes were designed including the interaction between different objects of the tool, we then proceeded to design the structured

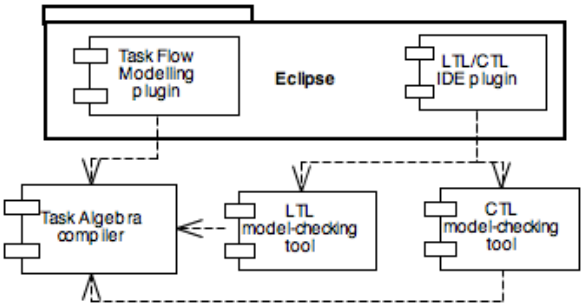


Fig. 1. Architecture of the Task Model Tool.

model. This model is presented in Figure 2. All development of the structured model is based on the use case diagram, when we should be extra careful as it migrates from an abstract model such as use cases and results in a diagram from which one has the possibility of building the computer application as such, in this case, set the application logic. Note that only cover part of the user interaction.

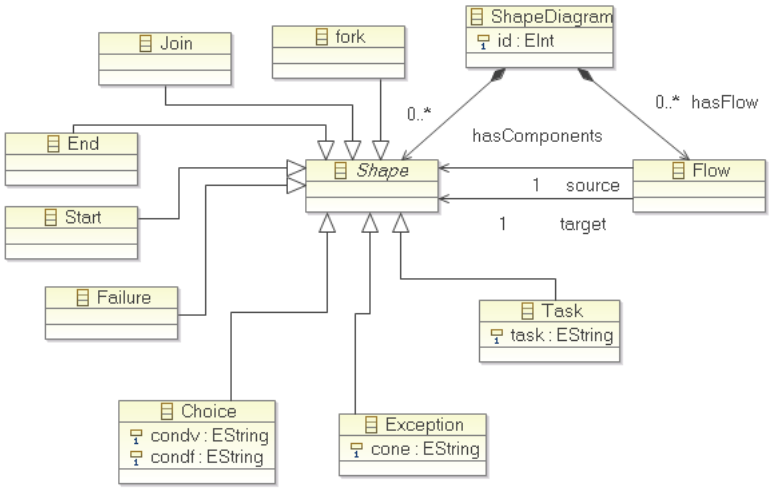


Fig. 2. Class model for the Task model plug-in,based on GMF.

4.2 Development of the graphical model

When the structured model is designed properly [2, 3], this can be transformed to the model graph. The model is a set of classes that represent real-world information. In our case, the components which are integrated with diagrams. For

example, the Choice component, is associated with a specific behaviour, therefore we need to store some additional information (i.e, this component implies information for the guards that will trigger the flow). All this without taking into account neither the manner in which that information will be displayed nor the mechanisms that make these data are part of the model; i.e., without regard to any other entity within the plug-in.

4.3 The domain model

The domain model (or the model itself) is the set of classes resulting from analysing the components needed to design a task flow diagram. Start, Task, Fork, Join, Exception, Failure, Choice and End are the classes that were defined for the domain model. The domain model is not related to external information, we have an overview of the components of each one of its elements.

4.4 The application model

The application model is a set of classes that are related to the domain model, are aware of the views and implement the necessary mechanisms to notify the latter on the changes that might give the domain model. The EMF framework, is responsible for this functionality, and which interacts directly with the structured model; i.e., the model built on EMF.

4.5 The view domain

The views are the set of classes that are responsible for showing the user the information contained in the model. A view is associated with a model. A view of the model gets only the information you need to deploy and is updated each time the domain model changes through notifications generated by the model of the application. GMF is responsible for receiving such notifications and for generating visual feedback on the plug-in.

4.6 The driver

The driver is an object that is responsible for directing the flow of enforcement due to external messages and requests generations of the algebra. From these messages, the controller modifies the model or open and close views. The controller has access to the model and views, but the view and the model are not aware of the existence of the controller. The controller itself is the result of the implementation code from the developer, which using GMF has the ability to interact with information from the visual editor plug-in. This operation is given by the $IWorkbenchWindowActionDelegate$ class implementation.

4.7 Integration

Finally when the two models have been integrated, we get almost all of the user interface plug-in. It is at this point when we have to develop the capabilities to manage graphics' performance and integration with the components of the translator (i.e., the logic implementation, where specific individual components).

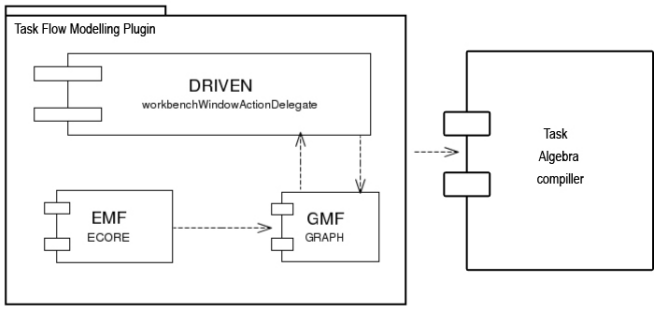


Fig. 3. View of integration and dependency of the plug-in for development of tasks diagrams.

4.8 Results

At this point we have obtained a comprehensive user interface, that is, the party responsible for managing the design process diagrams. It is worth noting that the code implementation has been rather small, since everything is generated from structured model. Up to this point we have managed to cover about half of the project. Figure 4 shows a screen user interface of this part of the project so far.

The development of application-based models implemented in the various tools for creating plug-ins, as is the Plug-in Development Environment, has resulted in optimization of time. The most important point is the possible modification, addition, facilitation and exploration of the plug-in, because you can just modify the structured model and its subsequent integration with GMF model to make accurate changes, all without writing a single line of code, so it is found that the design of the model implemented in a tool is superior to developments made entirely in code.

5 LTL que and CTL model-checking IDE

5.1 Verification Interface of Task Flow diagrams in the software specification

Some factors influencing the development of quality software are: Understanding of requirements, proper modeling of the use cases, verification of models and

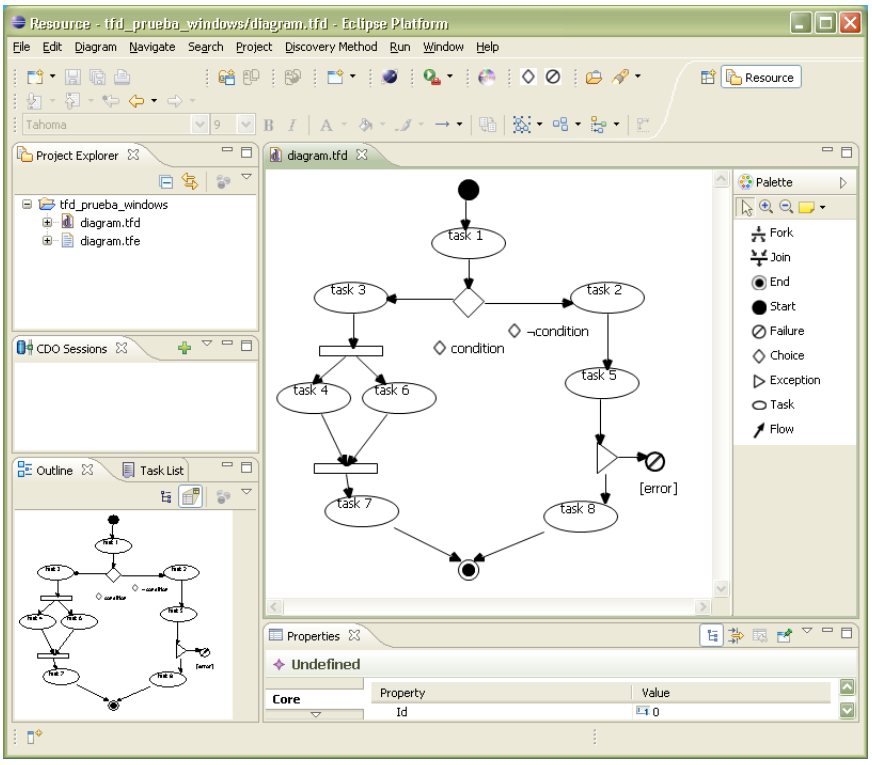


Fig. 4. Partial view of user interface for the Task Model plug-in.

development according to user needs. Task Flow diagrams from the Discovery Method are represented by a reduced and precise syntax. The verification over the Task Flow diagrams is performed using temporal logic functions. The most common temporal logics are Linear Temporal Logic (LTL) and Computational Tree Logic (CTL)[4]. The temporal logics are applied on an exhaustive set of states to see if a specification is true or not through time, it ensures verification of dynamic properties of a system without introducing time explicitly[7]. The Task Algebra proposed by Fernandez [6] offers already the tools (text mode) allowing you to verify Task Flow diagrams specified by the Discovery Method using temporal logic. This tool in text mode does not involve a visual representation of the operation and the logical transition of the model and it does not allow a full analysis of the results. The construction of an interface that allows to structure LTL/CTL queries and to graphically display results of the model verification represents the solution of the problem.

With the development of an interface to verify task diagrams, the user will have on hand a structured visual tool that lets him/her create logical expressions to refer to events in the algebraic model of work flow and display query results in a more meaningful and understandable way. With the creation of these

components the Task Algebra will become more accessible and with the help of appropriate technologies it will represent a contribution to the specification and design phase in software development.

5.2 Development Process

The flow of activities in the design phase can be modeled by Task Flow diagrams, which in addition to its graphical representation has a formal syntactic model. The formal model of the task diagrams is the basis for verification of system properties. The structure of a logical query(LTL/CTL) is complex, therefore it is necessary to assist it in the construction and comprehension of these expressions, as well as in the visualization of results.

Considering the ease of development, usage statistics and features offered in development environments, the interface of verification will be integrated as a plug-in in the Eclipse development environment. For best results, interface, testing and monitoring is necessary to take into consideration the following definitions for the task diagrams verification process:

- The plug-in should check the entry model that describes the task algebra.
- There should be a check of logical expressions created (LTL and CTL syntax).
- The test results should be displayed in an easy and simple way for user understanding.
- The verification interface should be efficient and effective.

Among the verification characteristics of the input model and the expressions syntax is used XText. In order to verify the input model and the syntax of the expressions we use XText. With XText, domain-specific languages (DSL) can be created in a formal and simple way. The framework supports the development of infrastructure in languages including compilers and interpreters and currently it has joined the Eclipse development environment. In interface development, Eclipse's core libraries such as org.eclipse.ui, org.eclipse.jface and org.eclipse.core are used. These packages allow to integrate icons and complete editor management, results in the interface development are shown in figure 5. As we can see, the task diagrams verification interface consists of the following elements: module expressions, work area and input models.

5.3 Modules Interactivity and Results

The input for this plug-in is a Task Algebra expression representing the Task Flow metamodel (see Figure 6a). This metamodel is used to generate the trace semantics needed to execute the query. A query construction is created and stored when the user builds LTL or CTL logical expressions (see Figure 6b).

The algebra model (tfa) and logical expressions created (tfq) are verified in continuous time using DSL grammars defined in the plugin (XText), which produces syntactically correct expressions. Combining the algebra model and

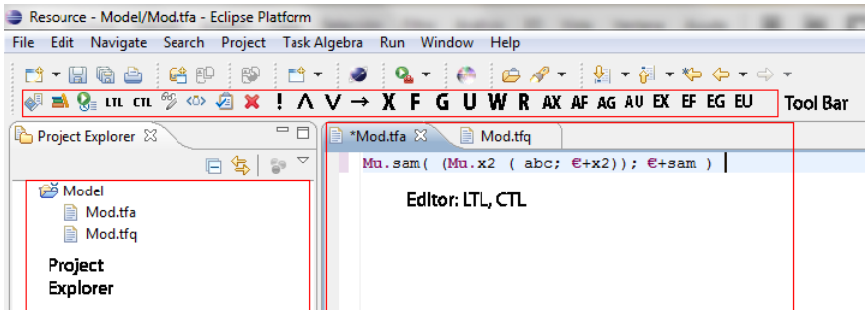


Fig. 5. Partial view of user interface for the model-checking plug-in: DSL Grammars, graphics elements and editor management.

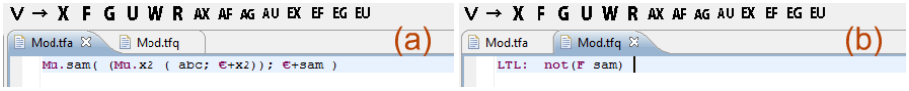


Fig. 6. Partial view of the user interface for the model-checking plug-in: (a) describing a Task Flow diagram, (b) describing logical expressions.

the correct logical expressions, the verification of properties in the model is executed using the text mode tool described in [6]. This part of the project is also responsible of the graphical display of the results. This is still a work in progress but it is considered relevant in order to facilitate the interpretation of the query results. In particular, the CTL results are the most difficult to understand in their present form.

6 Conclusions

Being Eclipse one of the most used environments for software development, we offer a tool that allows modelling and testing of software models that are defined usually in the specification phases. Our research presented the Eclipse plug-ins for the Task Flow model in the Discovery Method. The task algebra is based on simple and compound tasks structured using operators such as sequence, selection, and parallel composition. Recursion and encapsulation are also considered. The task algebra involves the definition of the denotational semantics for the task algebra, giving the semantics in terms of traces. Additionally, model-checking techniques were developed to validate Task Models represented in the algebra.

All of these was already available as console tools to prove the feasibility of the propose but, in order to be used by real-world developers, an IDE was necessary. With these tools, developers are not required to increase the quantity of artifacts when developing software. If developers create Task Flow diagrams, they will have an formal specification for their software which could improve

communications using the unambiguous notation. In addition, using software model-checking in early stages may increase the confidence that goes from a correct definition to the final design. The plug-ins developed facilitate the formal specification of the Task Flow models and the verification of these models in a visual and simple way. The queries are structured visually and with it the interpretation of results is even more simple. With this project the development time has been optimized and the quality of software has been guaranteed. In this project every module is easy to use and to understand for programmers due to its integration with Eclipse.

Acknowledgment

This work has been funded by the UTM.

References

1. Bogdanov, K., Bowen, J.P., Cleaveland, R., Derrick, J., Dick, J., Gheorghe, M., Harman, M., Hierons, R.M., Kapoor, K., Krause, P., Luettgen, G., Simons, A.J.H., Vilkomir, S., Woodward, M.R., Zedan, H.: Working together: Formal methods and testing, (2003)
2. Budinsky, F.: Eclipse Modeling Framework: A Developer's Guide. Addison Wesley, Boston, Massachusetts, firts edn. (2003)
3. Burd, B.: Eclipse for dummies. Wiley Publishing, Inc., Indiana, U.S.A (2005)
4. Chan, William, R.A.P.B.S.B.D.N., Reese, J.: Model checking large software specifications. IEEE Transactions on Software Engineering 24(7), 498–520 (1998)
5. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. 8(2), 244–263 (1986)
6. Fernandez-y Fernandez, C.A.: The Abstract Semantics of Tasks and Activity in the Discovery Method. Ph.D. thesis, The University of Sheffield, Sheffield, UK (February 2010)
7. Gurfinkel, A., C.M., Devereux, B.: Temporal logic query checking: A tool for model exploration. IEEE Transactions on Software Engineering 29(10), 898–914 (2003)
8. Simons, A.J.H.: Object discovery - a process for developing applications. In: Workshop 6, British Computer Society SIG OOPS Conference on Object Technology (OT '98). p. 93. BCS, Oxford (1998)
9. Simons, A.J.H.: Object discovery - a process for developing medium-sized applications. In: Tutorial 14, 12th European Conference on Object-Oriented Programming (ECOOP '98). p. 109. AITO/ACM, Brussels (1998)
10. Spivey, J.M.: An introduction to z and formal specifications. Software Engineering Journal IEEE/BCS 4(1), 40–50 (1989)
11. Wing, J.M.: A specifier's introduction to formal methods. IEEE Computer 23(9), 8–24 (1990)

Haptic devices on medical training applications, a brief review

Eusebio Ricárdez[1,3], Julieta Noguez[1], Lourdes Muñoz-Gómez[2]

[1] Departamento de Computación, Tecnológico de Monterrey Campus Ciudad de México, Mexico City, México
[2]Departamento de Tecnologías de Información y Electrónica, Tecnológico de Monterrey Campus Santa Fe, Mexico City, Mexico
[3]Departamento de Ingenieria en Computación, Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Culhuacan, Instituto Politécnico Nacional, Mexico City, Mexico
eusebior@ieee.org, {jnoguez,lmunoz}@itesm.mx
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. Current computer systems rely heavily on the senses of sight and hearing, making aside all other senses. Haptic devices, however, covers partial deficiency but also allows user to perceive tactile sensations. There are several developments in both hardware and software, showing various solutions. The tools used range from simple mathematical analysis, to issues of inference based on probabilistic models. Solutions use different types of haptic devices, some made or adapted by researchers and others that are already commercially available. This paper is a brief review of some applications of haptic devices in the field of medical training and professional use. A review of the state of art developments with haptic devices is described and opportunity areas in research are addressed to further develop a formal dissertation proposal.

Keywords: Haptic devices; virtual training environments; medical training; suture training system.

1 Introduction

In the field of medical training, like in many others knowledge areas, experience is a significant part of the learning process. However, even under the guidance of an expert a mistake can cause severe discomfort or death of the patient. Nowadays, it is required to update and reform the teaching and learning practices in this area; for instance, law reform and advocacy for human and animal rights have changed the way students are trained. Originally animals such as dogs, mice or rats were used to train new physicians. More recently mannequins to teach various techniques and surgical procedures are used. In the last two decades there has been an increment in the use of virtual environments (VE) that can simulate risky situations providing feedback to users. The interaction between humans and machines has focused mainly on visual and auditory issues without considering the sense of touch; the level of sensation has been relegated as tactile devices such as mouses, keyboards or touch screens used exclusively as hardware devices that enter information. In order to extend the usage of this hardware, haptic tools interfaces were developed allowing computer equipment to send tactile sensations to the user. In the field of experimental psychology and

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 35-45

physiology, the word haptic is used to describe the ability to perceive the environment through active exploration, using our hands, and feeling an object to perceive its shape and its properties according of the material the object is made up[1]. Haptic or haptics are words generally used to refer to all the feelings related to touch, including the ability to detect the position and movement of the limbs. More generally, haptics is commonly used today to refer to the science of touch in real and VE [1]. The idea of using touch as a means of communication comes from the years 1985 and 1999 [2], when it was considered that the channels of communication could be improved on the basis of combining various simple patterns and like a mouse to glide pixel by pixel on a graphic display. This raised the possibility of sending information to the user via a touch device as a mouse makes, thus, it is possible to send pixel by pixel to the haptic device information of shapes, depths and textures. This mechanism sometimes is called *haptic display* [2].

There are various studies that proposed the incorporation of haptic devices in virtual learning environments to improve the perception and performance of students [3][4]. Some applications can be found in medicine training. The incorporation of haptic devices in virtual learning environments represents new challenges for both electronics and computer science.

This paper is organized as follows: Section 2 defines haptic devices, describes the operation principles their classification and presents examples of commercial devices. Section 3 describes medical applications using haptic devices. Section 4 presents a summary of generic APIs to work with haptic devices. Finally, Section 5 is devoted to the discussion over research opportunity areas.

2 Haptic Devices

On this section we are going to describe the operation principles of haptic devices, their classification and some examples of commercial devices.

2.1 Description and operating principles

Haptic devices have been used in different areas and therefore are of different kinds of them. To get started, a haptic device can be defined as one that allows us to perceive tactile sensations from computer equipment interacting with it. A haptic device is said to be *under-actuated* when its number of actuators is smaller than its number of sensors, when a haptic device has the same number of sensors and actuators, it is said to be *fully-actuated* device. When there is a device with only sensors but no actuators, it is called *unactuated* device [5]. It is easier to develop under-actuated devices than fully actuated. Nevertheless there are many techniques to compensate for this disadvantage [5].

An ideal haptic device would be one in which a user could not distinguish between an object to be playing in the real world or virtual world [8], hard surfaces such as walls, would feel the same hardness in real life, the corners would be perceived even with sharp edges and the user would be able to distinguish surfaces with different textures. A real haptic device has a certain amount of friction, which add noise to the system, but in the long term and in extreme cases, can fatigue the user. In addition, the

device itself has some inertia, which is not a problem if the user moves slowly, but doing so quickly, the inertia of the system can generate an undesirable sensation, similar to be dragging extra weight and can also limit the maximum speed at which the device responds.

A real device must be properly balanced, in order to compensate different external forces, including gravity, likewise, must have mechanisms that provide sufficient reaction to stimuli and a sense of sufficient effort to simulate hard surfaces. [8] The resolution must be high, in order to provide as much detail as possible of the textures in the VE. The workspace of the haptic device must be large enough in order to simulate real workspace.

According to used control, there are two kinds of haptic devices: impedance control and admittance control. They differ not only in the type of control, but also in the mechanical structure [10], both of them have pros and cons. In the case of impedance control, the paradigm is the following: The user moves the device, and the device responds with a force if a virtual object is found. This means displacement as input and reaction force as output and implies that the user will inevitably feel the mass and friction of the device. The impedance control devices are light and manageable, normally has direct current (DC) motors.

The admittance control is the inverse of impedance, in this type of control the paradigm is: The user exerts a force on the device and the device responds with a displacement proportional to it; this is, force as input and displacement as output. This type of control, always consider freedom in the mechanical design of the device, because the displacement and inertia can be reduced by servo mechanisms, this makes such mechanisms more robust than impedance controlled and are able to move with great strength and stiffness. It is preferred to use these devices in industrial applications such as flight simulators, but due to its complexity, there are few haptic devices with such control, as they usually are bulky and must be carefully designed to interact safely with humans because of its strength and stiffness. An example of these devices is the HapticMaster from FCS Control Systems company (now Mog Corporation [11]).

The main features of the haptic devices, for both impedance and admittance, are the following:

Workload or workspace: Corresponds to the dimension or volume that can reach the device in the physical space considering the three coordinate axes.

Resolution: Specifies the slightest movement can be detected or made by the haptic device, shown in sub-multiples of a meter or in dots per inch.

Force: The force that the device may assert, as a combination of actuators at a given point. Nominal force is the one the device works normally with peak force or maximum, is the one that device may occur over short periods of time.

Maximum speed: Represents the speed which device can move or be moved, considering that inertia must be compensated to achieve more optimal simulations.

2.2 Classification of haptic devices

As we mentioned before, there are several ways to classify haptic devices, depending on its application, degrees of freedom, performance, etc. Bello [6] organizes haptic devices in large groups, related to the application or the form of interaction, touch screens, exoskeletons and stationary devices, gloves and wearable devices, interaction point and devices for specific applications. According to its application, devices can be described as follows:

Programmable Keyboard: One of the earliest examples of the implementation of a feedback keyboard was the Clavier Rétroactif Modulaire Project (1990). This consists of a keyboard similar to a piano that provides force feedback computer-controlled in each of its 16 keys [2], which focuses on music research.

Exoskeletons: Exoskeleton devices were developed by Bergamasco et. al. (1992), incorporating several biomechanical observations of the human body [2]. In order to become functional, the researchers used different techniques to improve engine performance, reducing friction and adding sophisticated methods to guide the cables.

Gripping or grasping devices: One of the first devices designed for this purpose was developed by Howe in 1992 [7], the device has two degrees of freedom and was designed for two fingers. The user's fingers interact unilaterally with the device on the inner side of the banks, generating a precision grip.

Point Interaction: This device assumes that you only need one contact point to perceive virtual objects. An example of this family is The Phantom™ device, described below.

Increased mice: This type of mouse was described by Akamatsu [2], is generally a device shaped and sized like a computer mouse, but includes two haptic features: One is an electromagnetic brake to program the forces of friction and the other one is a transducer that provides vibro-tactile sensations.

Joystick: Adelstein and Rosen [2] show an example of a joystick with two degrees of freedom and force feedback, from there; other devices have been designed another similar joysticks.

Virtual reality devices: There are many devices for VR applications. Their origins date back to 1992 on a paper presented by Burdea [9], who show a series of pistons positioned in the fingers, so you have to oppose opening and closing the hand currently. The leading exponent of this technology is CyberGrasp from VRLOGIC[1], which provides force feedback on every finger and allows you to locate the hand position in three dimensions.

Isometric devices (or admittance controlled): There are few examples of such devices, but one of the most representatives is the Haptic Master from the FCS Control Systems company [11]. This consists of a robotic arm and a control unit that generates six degrees of freedom in a force feedback device that allows very considerable magnitude about hundred of Newtons.

In this job we focus in Learning Virtual Environments using haptic devices to train physicians. For this reason, in the next section it will be described the most common haptic devices used on VE medical applications.

2.3 Commercial devices used on medical applications

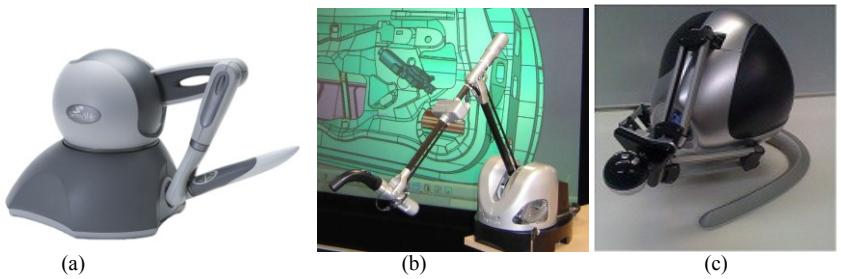
Following, a brief description for most commonly used haptic devices in VE for medical applications is presented.

Phantom Device: Originally developed by the Massachusetts Institute of Technology (MIT) [8], is a desktop device that provides an interface that shows strength between the user and the computer. It is controlled by an impedance control algorithm and there are currently several variants of this device, Phantom Omni (Figure 1a), Phantom Desktop, Premium Phantom and Phantom Premium 6DOF. Where the end-effector is a

[1] http://www.vrlogic.com/html/immersion/cybergrasp.html

stylus or a thimble, in which the user holds the stylus or insert his finger in the thimble. It consists of three DC motors connected to a broadcast and encoders for position coordinates x, y, z, torque engine is transmitted by a series of pre-tensioned cables that reduce the stiffness of the device because to its lightweight aluminum construction. The three axes coincide at one point, so it eliminates the torque at that point, this enables a single contact point in the virtual world, enhancing the sensation on VE. This device has been widely accepted and used by various researchers.

For applications development involving this device, SenSable has *OpenHaptics development platform toolkit* that works on Windows™, Linux and Mac OS™, using native OpenGL for graphical environment.



(a) (b) (c)

Fig. 1. (a) Phantom Omni device from Sensable Technologies [27], (b) Virtuose 6D35-45 [12], (c) Falcon device, Novint Technologies Inc. [13].

Virtuose 6D35-45: this is a device with 6 degrees of freedom (6DOF), shown in figure 1b it is designed specifically to work in VE. According to the manufacturer it is the only device in the market to have full feedback on the 6DOF. Operation freedom emulates a human arm, it is composed of two major joints fixed to a rotating base, the second segment ends with a "wrist", which can rotate on three concurrent axes and offers the possibility of exchange the end actuator to place different types of tools. It also has three buttons that can be used to interact with the GUI.

For the development of graphics applications, manufacturer offers three software solutions: Virtuose API, provides functionality for haptic devices; IPSI API, includes a solution for collision detection and Core PPI and IPP Human that allows the development of applications for physical interaction between humans and objects. All for Windows and Linux platforms, for the first two, the programming is done through C ++ language, while the latter, has its own visual programming interface. This device connects to computer via Ethernet interface and belongs to the classification of controlled impedance.

Falcon device: this device is manufactured by Novint Technologies Inc. (figure 1c) is primarily targeted for using with games. However, manufacturer provides as development kit a set of libraries for Windows platforms, using programming language Visual C ++. Graphic management is done through OpenGL to develop own applications. This device belongs to control impedance family, has 3DOF with a delta robot-type configuration, and it is possible to exchange the handle either with a sphere or a gun [13]. It is connected to the computer through a USB interface. The drivers are provided by the manufacturer, allowing communication and an effective update frequency between 800Hz and 1 kHz.

Table 1 shows a comparison of some features for haptic devices mentioned above.

Table 1. Comparative table of different haptic devices

Specification	Phantom Omni	Falcon	Virtuose 6D35-45
Workspace (mm)	160 x 120 x 70	101 x 101 x 101	150 x 150 x 150
Resolution (mm)	$\approx 55x10^{-3}$	$\approx 63.5x10^{-3}$	$6x10^{-3}$
Force feedback Nominal /peek	0.88 / 3.3 N	8.8 N	35 / 10 N
Control software	Open Haptics C++	Novint SDK C++	Virtuose API C++
Interface	IEEE1394 Firewire	USB 2.0	Ethernet
Degrees of freedom	6	3	6
Degrees of force feedback	3	3	6
Manufacturer	Sensable	Novint	Haption

3 Haptic devices in medical applications

In this section we present a summary of various studies that use haptic devices on VE for medical applications. There are developments in hardware and software, and some solutions include from simple mathematical analysis to inference issues based on probabilistic models. The use of haptic devices in medical simulations in the first instance intended to improve training applications and its used in a variety of specialties like palpation, needle insertion, suturing, laparoscopic surgery and so forth.

On palpation a doctor presses on interest area with his fingers to locate reference points below the patient's skin and feel the presence or absence of anatomical features and/or physiological abnormalities. Direct contact patient/doctor is required for palpation and requires the simulation of force and tactile feedback. There are not commercial devices that perform this function, which has limited current solution for palpation simulation. There are different applications related to palpation, either using devices specially built for that purpose or using generic haptic devices like the Phantom. Examples of these types of application are: palpation of the knee [16], malignant tumors in head and neck [17] or use fingertips to felt abnormalities from a modified haptic device [18] (Figure 2).

Fig. 2. Palpation device combining force feedback from a Novint Falcon and tactile feedback at the fingertips from piezoelectric materials. [18]

There are several commercial applications for needle insertion; an example is The Mediseus Epidural simulator (Medic Vision) that includes force feedback. Simulation runs on a laptop, gives vocal response if the user makes mistakes and produces a report for the student [20].

On many interventional procedures, the initial step is the insertions of a needle o trocar as a guide for introduce another tool. Most current commercial simulators for minimally invasive surgery are built with the introducer already in place e.g. CathSim AccuTouch System [21]. It contains a needle carrier with 3 DOF and one degree of force feedback (DOFF). This one DOFF allowed the simulation of a needle passing trough different tissues. Suture simulating is another kind or needle insertions, there are several studies on the construction of a virtual suture simulator that allows to teach to future physicians different techniques for human tissue sutures, simulating different conditions and circumstances [22] [23] [3] [24] [25].

O'connor [3] probed that the use of haptic feedback improves performance of student learning and describes the difficulties of training in laparoscopic suturing. Also shows the complexity of the tasks and performs an experiment with different students where one group uses haptic feedback and another group does not.

On Brown [24], there are different ways to represent deformable objects, soft tissues, and several types of collisions needed to work with suture techniques. Also raises collisions that can be used for the realization of knots. However leaves open the implementation of the haptic device in these tasks.

Oshima [26] develop a training system for suture/ligature that provides quantitive information about the process of student learning and proposes a solution not entirely virtual. From a modified device for venipuncture, they built a device through synthetic leather that allows students to practice different kind of stitches. Students are evaluated by an imaging system that provides the teacher with an assessment of the job. A disadvantage of this system is the use of artificial skin, which must be changed for each operation. This training simulator is showed at figure 3.

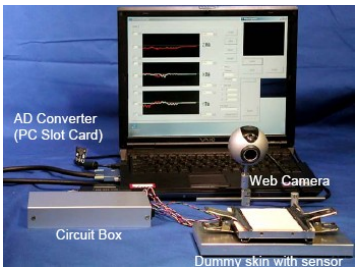


Fig. 3. Screenshot of the developed surgical skill training simulator at Waseda University. [26]

On applications involving virtual haptic devices, H. F. Shi [25] presents a model of knotted and sutures based on a Virtual Training Environment (VTE), throw an approximation of a real-time simulation of deformable linear objects (DLOs) with visual and force feedback. Special emphasis is placed on mechanical properties of a real thread such as stretching, compressing, bending and twisting, and these properties are considered in the propagation forces along the suture when the user pulls with one or both hands.

The user can practice basic suturing techniques in the simulator; the wounds are modeled on a spring mass system. Also, it simulates the instruments involved in the process, as well as collisions between soft tissue and the needle. One of the main

contributions of this work is the use of the graphic processing unit (GPU) to perform calculations of deformable objects.

Regarding the need to simulate with high fidelity models not only visually, but also the haptic modeling, there has been several ways to get the best representation of human tissues. The two main solutions proposed are the mass-spring system (MSS) as in the previous case [25] and methods based on finite element models (FEM). However, due to the computational cost of latter, most authors decide for using the MSS like in case of L. L. Lian [23], from this model he shows a different way to represent human tissue. It also deals with collisions in the process and the behavior of a needle and thread and the forces involved on them.

Roger W. [22], describes a virtual simulator of simple wound suture, there are a needle holder attached to the haptic device, the graphics of needle holders, needle, sutures and virtual skin are displayed and updated in real time. The simulator incorporates several components such as real-time modeling of deformable skin, tissue and suture materials, and a real-time recording of state of activity during the job of suture, figure 4 shows this work.

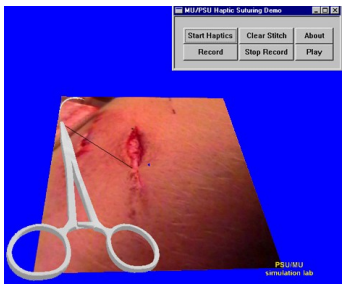


Fig. 4. Screen shot of haptic suturing simulator [22]

Recording positions is one feature that is only possible with haptic devices and can be added to training, Cagatay [27] called this “haptic recording” and allows an instructor register through haptic devices and the appropriate software, the sensations perceived during a particular action, such as surgery and with this shows the student what to expect in the real world.

Cagatay also describes a laparoscopy training system, using two Phantom haptic devices. Software highlights special features, like collision detection; also, special emphasis is placed on "deformable object model" used to represent internal organs. Finite element model is necessary for this representation.

4 Software for haptic devices

There are several Application Program Interfaces (APIs) that can assist during the creation of VE, some created by the manufacturers themselves, such case is OpenHaptics from SensAble Company[2]. This toolkit allows developers to integrate haptics to existing third-party applications and development of new applications. OpenHaptics provides a new and extensible architecture that offers a framework for

[2] http://www.sensable.com/products-openhaptics-toolkit.htm

developing multi-layer applications, allows developers to program the rendering forces directly, offers control over the dynamic behavior of drivers and provides utilities and debugging aids. OpenHaptics works on Windows, Linux and Mac OS, but is limited only to the Phantom family devices, for the graphics is preferably designed to work with OpenGL, although it may work with DirectX or any other graphics library.

Another API is CHAI3D library [15], this includes graphic components, force feedback and control algorithms for haptic rendering. This is an open source library, it is written in C ++ mainly for academic use, it is easy to add extensions and supports several commercial haptic devices.

Haptik Library [28] is a library with an open source component-based architecture, which acts as a hardware abstraction layer, to provide uniform access to haptic devices. It does not contains graphic guidelines, algorithms related to physics or classes of complex architecture, but instead presents a set of interfaces that hide the differences of the devices to applications. This means that any dependence on a particular device is removed from the code and executable programs, ensuring that the application to work regardless of installed driver.

H3DAPI is open source software that uses OpenGL and X3D standards. Unify the management of haptics and graphics in a single scene. It is made to get independence in the management of different haptic devices and allows the integration of audio and stereographic displays.

5 Research opportunity areas

In previous articles there are different applications involving haptic devices in the field of medical training. However we can distinguish several opportunities for developing new applications. E.g. in suture despite the existence of several works, none of them can be seen that the procedure used by doctors to play virtually. Several studies have focused on virtual representation of the tissues and behavior of the thread and needle [25], other works allow virtual sutures including stereoscopic vision [22]. However the procedure presented is different from that one used by doctors in real operations, because simulators use only one hand and a single haptic device, while doctors uses both hands. This provides the opportunity to develop an application that allows integrating a haptic for each hand. Also, we can propose another one to allow contact with virtual objects in more than one point, possibly using a haptic glove.

Additionally, most medical applications including haptics are too specific e.g. laparoscopic applications are designed to work in a specific part of the body: knee, liver, gallbladder and so on. If users want to change the experiment, they must redo the application and change hardware. This left open research to generate some kind of architecture that easily supports application change, or at least a wider range of experiments. Also, is pending the job of developing applications to evaluate performance of student. The use of a smart tutor can help student to repeat work without presence of teacher.

6 Discussion and Conclusions

Current trends in research are allowing the development of virtual laboratories for learning and training in different areas including medicine. Because of the existence of low cost commercial haptic devices, nowadays is easier incorporate to serious applications this kind of devices. To build a medical training application with a haptic device, it is necessary to take in mind several considerations. Including if we want to build our own haptic device, use a commercial device or modify an existing one. Also, according to the application, it should be consider the number of degrees of freedom needed; the required computational would be provides, because both haptic and graphic require significant resources. Once elected the device, it should de consider whether it is necessary to work with their libraries or can use a generic API to help us more quickly and easily perform our work. Considering a medical application, the main question could be, is haptical representation as true as real touch? It is also necessary to add some functionality to evaluate user performance, in order to facilitate the learning or training process and reduce the costs involved.

Although there are several works in the medical field, the evaluation of their performance is so far subjective, as it is very difficult to assess whether a feeling is right or wrong, the best we can say about touch is "seems fine" and several tests are needed to establish statistically a generality.

Evaluation mechanisms will allow students to know about their performance even without the presence of an expert (teacher). These mechanisms can be implemented using artificial intelligence techniques such as neural networks or Bayesian networks to develop specialized intelligent tutoring systems.

Acknowledgments. This work was supported by a grant provided by the Instituto Politécnico Nacional and by Tecnológico de Monterrey, Campus Ciudad de México, through the e-Learning research group.

References

1. Robles de la Torre, Gabriel. The International Society for Haptics. [Online] 2006. [Cited: 02 25, 2010.] http://www.isfh.org/haptics.html.
2. Hayward, V. Tutorial Haptic interfaces and devices. *Sensor Review* , 16-29. 2004.
3. O'Connor, A., Schwaitzberg, S. D., & Cao, C. G. How Much feedback is necessary for learning to suture? 2008. *22*, 1614 – 1619.
4. Botden, S. M., Torab, F., Buzink, S. N., & Jakimowicz, J. J. (2008). The importance of haptic feedback in laparoscopic suturing training and the additive value of virtual reality simulation. *22*, 1214 – 1222.
5. Barbagli, F., & Salisbury, K. (2003). The Effect of Sensor/Actuator Asymmetries in Haptic Interfaces. Proc. of the 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems (HAPTICS'03). IEEE.
6. Bello, F. (2006). Haptic for Surgical Training. Robotic Surgery: The Kindest Cut of All, 2006. The Institution of Engineering and Technology Seminar on (pp. 49-72). London: Imperial College London.
7. Howe, R. D. (1992). A force reflecting teleoperated hand system for the study of tactile sensing in precision manipulation. *Proc. IEEE Int. Conf. Robotics and Automation*, (pp. 1321-6).

8. Massie, T. H. (1993). Design of a three degree of freedom force-reflecting haptic interface. massachusets: MIT.
9. Burdea, G. C. (1992). A portable dextrous master with force feedback. *Presence: Teleoperators and Virtual Environments* , (pp. 18-28).
10. Van der Linde, R. Q., & Lammertse, P. (2003). HapticMaster a generic force controlled robot for human interaction. *Industrial Robot: An International Journal , 30* (6), 515-524.
11. Moog Inc. (n.d.). *Moog Inc.* Retrieved November 25, 2009, from http://www.moog.com/products/haptics-robotics/
12. Haption SA. (2010). *Force feedback for professional applications*. Retrieved 04 28, 2010, from http://www.haption.com/site/eng/html/produits.html.
13. Novint Technologies Inc. (2010). Retrieved 03 21, 2010, from http://www.novint.com.
14. Steve, M., & Hillier, N. (2009). Characterization of the Novint Falcon Haptic Device for Application as a Robot Manipulator. *Australasian Conf. on Robotics and Automation.* Australia.
15. Conti, F., Barbagli, D., Morris, D., & Sewell, C. (2005). CHAI: An Open-Source Library for the Rapid Development of Haptic Scenes. *World Haptics Conference (WHC'05).*
16. Langrana, N. A., Burdea, G., Lange, D., & Deshpande, S. (1994). Dynamic Force Feedback in a Virtual Knee Palpation. *Artif Intell Med , 6* (4), 321-333.
17. Gastal, M. O., Henry, M., Beker, A. R., Gastal, E. L., et al. (1997). Human Performance Using Virtual Reality Tumor Palpation Simulation. *Computers and Graphics , 21*, 451-458.
18. Bello, F., Coles, T. R., Gould, D. A., Hughes, C. J, et al. (2010). The need to Touch Medical Virtual Environments? *Workshop en Medical Virtual Environments at IEEEVR2010.*
19. R.L. Cunningham, R.F. Cohen, R.H. Dumas, G.L. Merril, P.G. Feldman J. L. Tasto. *Haptic Interface for Palpation Simulation. US 7202851* US, 2001. US Patent.20. Mayooran, Z., Watterson, L., Whithers, P., Line, J., Arnett, W., & Horley, R. (2006). Mediseus Epidural: Full-Procedure Training Simulator for Epidural Analgesia in Labour. *SimTecT Healthcare Simulation Conference 2006.*
20. Mayoran, Z., Watterson, L., Whithers, P., Line, J., Arnett, W., & Horley, R. (2006). Mediseus Epidural: Full-Procedure Training Simulator for Epidural Analgesia in Labour. *SimTecT Healthcare Simulation Conference 2006.*
21. Cunningham, R. (2003). Simulating Reality with immersion Medical Interview by Senhat S. Demir. *Eng Med Biol Mag , 22* (5), 11-13.
22. Webster, R. W., Zimmerman, D. I., Mohler, B. J., Melkonian, M. G., & Haluck, R. S. (2001). A Prototype Haptic Suturing Simulator. In J. D. Westwood (Ed.). (pp. 567-569). IOS Press.
23. Lian, L. L., & Chen, Y. H. (2006). Haptic Surgical Simulation: An Application to Virtual Suture. *Computer-Aided Design & Applications , 3* (1-4), 203-210.
24. Shi, H. F., & Payandeh, S. (October 11-15, 2009). Simulation in Surgical Training Environment. *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems.* St. Louis: IEEE.
25. Oshima, N., Solis, J., Ogura, Y., & Takanishi, A. (2007). Development of the Suture/Ligature Training System WKS-2 designed to provide more detailed information of the task performance. *Proceeding of the 2007 IEEE/RSJ international Conference on Intelligent Robots and Systems* (pp. 58 -63). San Diego, CA: IEEE.
26. Cagatay, B., De, S., Jung, K., Manivannan, M., & Hyun, K. (2004). Haptics in minimally Invasive Surgical Simulation. *IEEE Computer Graphics and Applications* , 56-64.
27. Sensable Technologies Inc. (2010). Retrieved 03 21, 2010, from http://www.sensable.com/haptic-phantom-omni.htm.
28. *M. de Pascale and D. Prattichizzo* "The Haptik Library: A Component Based Architecture for Uniform Access to Haptic Devices" *IEEE Robotics & Automation Magazine, vol. 14, no. 4, pp. 64-75, Dec. 2007*

ARSK: an edutainment application using augmented reality for basic education children to strength the knowledge of the human skeleton

Erik Ramos[1], Esperanza Pérez_Córdoba[1], Jorge Hernández[1], Mónica García[1], Hugo Martínez[1], Moisés Ramírez[1], Omar Cruz[2], Alfonso López[2], and Myriam Reyes[2]

[1] Instituto de Computación
Universidad Tecnológica de la Mixteca
{erik,mapercor,jahdezp,mgarcia,hugoe,moiseseg}@mixteco.utm.mx
[2] Ingeniería en Computación
Universidad Tecnológica de la Mixteca
{omar.omarneon,alponcho,kareninamy}@gmail.com
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. At the moment, the quality of education in Mexico is considered as poor, many basic education students do not reach the goals set at the study programs. Given this situation, it is necessary to develop strategies to strengthen the teaching-learning process in basic education. The use of technology in education represents an important factor in achieving those purposes. For years, the introduction of technological devices, as support educational tools, has proved significant benefits increasing the level of students learning. Moreover, with the development of new technologies such as mobile devices, this effect is even greater. The present work focuses on the development of an educational application for mobile devices. This tool is developed as a support for the learning of the bones of the human body by students of third year of primary school, in the natural science subject at school. We use augmented reality (AR) with the aim of improving the student experience and strengthen the educational concepts taught in the classroom. AR allows blending real world images with virtual objects, modeled with the use of tools for creating three-dimensional (3D) graphics. The results show that children could improve their knowledge with the help of the application. Usability testing results reached a user satisfaction of 93%.

Keywords: augmented reality; 3D models; edutainment; smart phones; virtual reality

1 Introduction

According to studies published by the Instituto Nacional para la Evaluación de la Educación (INEE), there is a low quality of education in Mexico [3]. Despite the undeniable progress in the education, this is not enough; basic school students exhibit low levels of learning and do not reach the goals set by government

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 47-57

programs in public institutions. On an international scope, the situation is even worse: 15 years old young Mexicans have by far lower levels of competence compared to the youth in developed countries. A considerable proportion of these students do not reach the minimum necessary to pass proficiency tests. The sources of these problems are usually the lack of attention by students, but also the problem lies in the learning methodologies used [28]. To address this problem it is unavoidable to find alternatives for strengthening the teaching-learning process in education, and the use of technology is a good option.

The application and use of Information and Communication Technologies (ICT) in various fields in education is growing [9]. The introduction of technological devices, media and general ICT as educational support tools have demonstrated significant benefits in increasing the level of students learning, especially with the development of new technologies such as mobile devices. The popularity of mobile devices has increased [20], and now it is common that most people have access to them, including children and youth. The acceptance of these devices is mainly due to certain characteristics such as portability, cost, and ease in handling. New generation mobile phones have more uses than communicating; they are used as means of entertainment, a feature that can be exploited for educational purposes.

There are many applications focused or developed to support education, such as games, collaborative systems, etc [13]. This will enhance the user experience and increase the use of systems. This paper describes the development of a learning support tool, which is the first step of a project to develop educational applications for mobile devices. As a case of study, we address the issue of the bones of the human body as part of the curriculum of the third grade in the natural science subject, according to the educational programs of the new Reforma Integral de la Educación Básica 2009 [4].

The purpose of applying AR in education is to enhance the student experience as well as reinforce concepts and knowledge acquired, as shown in [14, 22, 23]. AR is an emerging technology that allows us to generate 3D virtual objects, juxtaposing these objects on a real life scene in real time.

The aim of this paper is to incorporate AR as a teaching resource for education, seeking to reinforce the concepts discussed in our case study. Next section describes the pedagogic fundamentals, which specifies the topic that follows and which level address the issue of the skeletal system, as well as the expected learning. Section 3 deals with the techniques for the modeling in Blender and especially those that are employed to develop this project. Section 4 justifies the use of mobile devices. Section 5 discusses the issue of augmented reality, how it works and the benefits it provides when is used in education. Section 6 shows the results of usability tests of the system, and conclusions can be found at the last section.

2 Pedagogical Foundations

In Mexico 77.4% of students are registered in basic education: preschool, primary and secondary school [6]. Primary school is quite important, as at this level students develop their thinking skills. Although efforts have been made, basic education in our country remains in a poor state [7]. The government has launched projects to improve education, as mentioned in [8, 10, 5]. Thus, these projects seek to interpret and to communicate phenomena, concepts, processes or procedures from several points of view and assess the relevance of the results with appropriate pedagogical mediation, ensure better training processes [8]. The field of natural sciences at basic education is where students reach the highest percentage of performance (scores place them just 25 percentage points of achieving mastery of the issues set out) [2]. These results may be explained by the learning from daily contact with natural phenomena. Within the contents of third year, the least dominated topics in the area include: functions, care of human body systems, natural resources and their protection, as they are involved with improving quality of life. The main interest of addressing the issue of the human body is to strength the childrens knowledge on the osseous system

3 3D Modeling

A 3D model is a graphic representation of an object or entity within a three dimensional space (x, y, z). There are many ways to build a 3D model, from the use of OpenGL directives up to the 3D design tools. By using directives of a programming language like OpenGL, yields a model that spends less computing resources while rendering. But a more complex model requires a lot of effort to design it. Using a 3D design tool is possible to create complex models in an easier way and wasting less time. This implies that this model must be exported for using it in final applications, therefore rendering needs more memory space and processing resources.

We decided to use a 3D design tool, since the used models must provide the greatest possible realism. We opted for Blender [1] because it is a free, robust and popular tool.

The main modeling technique used was rotoscoping [24], which consists in shaping a 3D object using at least two views (front and lateral). These views are used as background images in the 3D design tool (see Figure 1), then a primitive whose shape is close to what you want to model is chosen (see Figure 2).

The technique of polygon mesh modeling was used to shape the primitive [15] (see Figure 3), it consists of selecting certain vertices and placing them on the outline of the background image (see Figure 4).

Once the primitive has the outlined shape of the views used to model the object, it is necessary to add details to shape special parts that bring relief or cavities, using boolean operations modeling [15].

The eyes cavities were then created using shaped sphere primitives (see Figure 5). Then, we applied boolean operations to create the cavities in the skull, obtaining the desired effect (see Figure 6).

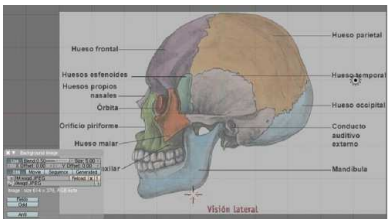


Fig. 1. Lateral view of a skull

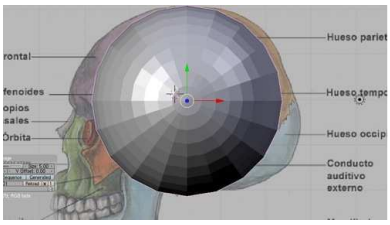


Fig. 2. At the background, lateral view of the skull. At the front, the primitive (sphere) that will be used to model the skull

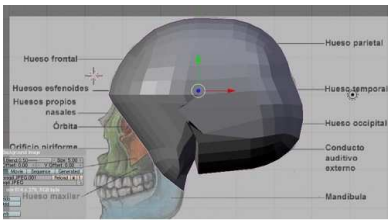


Fig. 3. Lateral view of skull

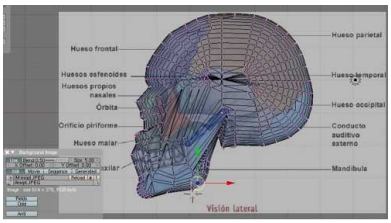


Fig. 4. Lateral view of skull

Finally, once the 3D object model has all the necessary details (see Figure 7), we applied NURBS modeling [15] to give a more organic or bent appearance (see Figure 8).

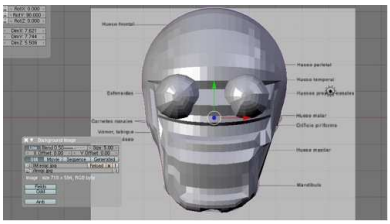


Fig. 5. Adition of shaped sphere primitives to create the eyes

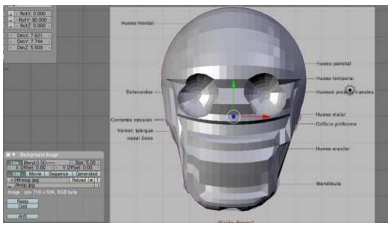


Fig. 6. Modified skull after applying boolean operations of subtraction to get the cavities of the eyes

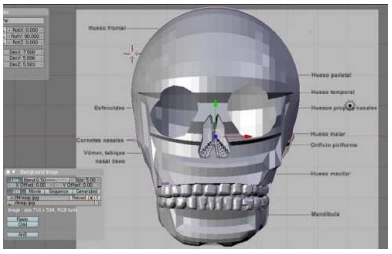


Fig. 7. 3D skull model after adding details off eyes, nose and teeth

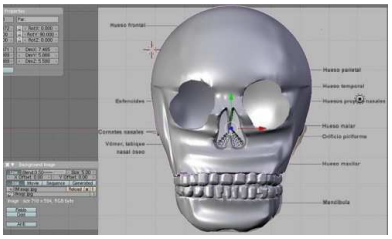


Fig. 8. 3D skull model after applying NURBS

4 Mobile Development

There is not a widely accepted definition of what a mobile device is, but it can be considered as a microcomputer that is light enough to be carried by one person,

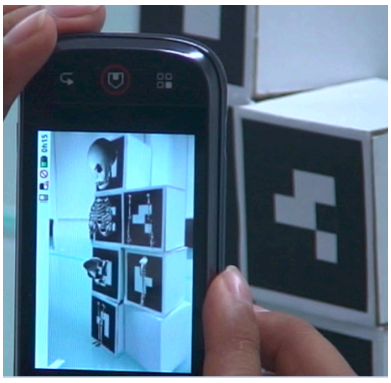


Fig. 9. Application running in a Motorola Dext smart phone

and has the ability to be operated autonomously [27]. The use of these devices is in contestant growing; as a parameter, last year sales increased 13.8% [20] and smart phones grew 19.0%. The most popular Operating System (OS) is Symbian, but the one that shows the biggest expansion is Android (3rd most popular), forecasts predict that Android will hold the first place by 2014. Android is an open source OS for handsets developed by Google and the Open Handset Alliance, based on Linux kernel. Android provides its own Java framework suited for a fast mobile applications development [19].

The use of mobile devices in fields like education is growing because of their capabilities (computational resources, sensors, cameras, etc.), and the variety and quality of novel educational contents [21]. Thereby we used a mobile device (see Figure 9) with an Android platform.

5 Augmented Reality

AR is a technology derived from Virtual Reality (VR) which allows the users to add virtual objects over a real stream letting the users to interact with them as if they were real. 3D models are used in most of cases, but it is not a compulsory requirement [11, 25]. In other words AR is the result of combination of virtual objects generated by a computer and the reality as seen by the user, thus a real time mixed reality is created. Unlike VR, AR combines real and virtual objects, whereas VR substitutes real-world features for virtual items.

During the present decade, AR has been widely used in educational environments, developments have been towards didactic material, which offer more realistic experiences than images or static contents [12]. Through AR techniques, a more diverse content can be added to treat complex concepts inside the classroom, ought to these concepts in the real world could be dangerous or expensive.

A useful AR features in education is [18] as it attracts attention because there is a direct manipulation of objects and encourages teamwork. Like most of

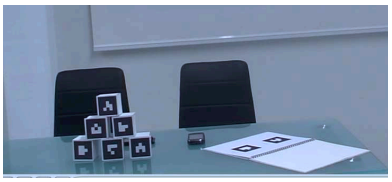


Fig. 10. On the left, cubes with printed marks for team tests. On the right, the bones album

new technologies, AR has the power to attract the attention of students by creating more complete and constructive environments, which stimulates learning and encourages involvement in the process of knowledge discovering by themselves [16, 26]. Other important features are that using the hands, it is possible to interact with the model, yielding a richest experience; it is possible to support teamwork because unlike the use of computers on which a working group directs their attention to the screen, using AR, students can concentrate on the model and interact with it at the same time. Another kind of possible manipulations of virtual objects are zooming in or out, changing of colors or textures and rotations [14, 17].

Overall, the AR scheme is: a camera captures reality that will be the modified to create the virtual scenario. Special marks or patterns are identified by processing the real stream, once identified such patterns the next step is to load and display a predesigned 3D model inside the real stream. The 3D model can be modified as mentioned in the preceding paragraph in real time and displayed inside the modified stream creating the illusion that the virtual object is part of the reality acquiring attributes that only real objects have, such as size, position, lighting and viewing angle.

6 Results

In order to determine if the application can be considered as a didactic tool, we applied usability testing. Those tests were performed at our university UsaLab (usability laboratory) to evaluate the application and to verify its ease of use. Also, we made another tests to measure the degree to which the developed application helps to strength the kids knowledge. A first test consisted in a questionnaire to determine the impact in the knowledge learned by students. In this case study the topic was: the bones of the human skeleton. We elaborated two questionnaires, one applied before and the second after the usability tests. These tests were planned with the objective to be applied to one person or a team. For the individual test we made an album which has marks or patterns, the smart phone displays a bone of human skeleton related to the mark; for team tests we made cubes with the same patterns (used in the individual test) printed only in one face (see Figure 10).

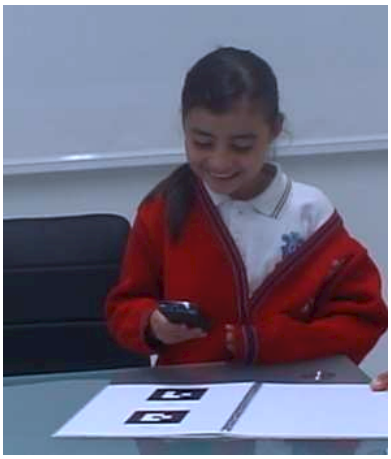


Fig. 11. First test using AR (Individual test).

Five children participated in the usability testing, two boys (A1 and A2) and three girls (A3, A4 and A5). The first activity consisted of answering a questionnaire to determine the level of knowledge of each student, about the bones of human body. In the first individual test, each student could appreciate the virtual image generated by the application (see Figure 11). We used two smart phones, Motorola Dext and Motoroi. Students showed an expression of surprise when they saw the virtual image. In general, for the children, the first test was fast and easy. Statistics were generated to measure efficiency of the use of the application according to the childrens behavior. A5 was notably the student that spent the longest time to detect the marks with 3min 60s. A1 was the fastest to perform the tasks in 1min. 3D virtual images showed were interesting for all of them, as an example A4 said: Ohhh .... It looks like in the cinema.

For the second test we organized three teams: T1 (A1, A2), T2 (A4, A5) and T3 (A3). A task was assigned for each team, consisting of building the human skeleton using the cubes. Each team created a different structure with the cubes. T1 built the skeleton as shown in the Figure 9.

The test was very interesting for the children, so they could work together, talking among themselves and discussing to decide the best way for assembling the skeleton [16]. When they commented about the activities, they said: We liked to work with the cubes for building the human skeleton.

After finishing the second test, students wrote a new list containing the names of the bones. Each child wrote more names than in the first questionnaire. The first time, the most students wrote only the name of the ribs, but when the same questionnaire was answered after the tests they wrote more correct names (at least five from six). Finally, the students evaluated the application with a score from 1 to 10, the degree of satisfaction reached was 96%.

7 Conclusion

We conclude that smart phones are an appropriate platform for the development of applications that seek to strengthen the teaching-learning process. Trends indicate that with the increasing sales and popularity of these devices, it can be expected to become a common element of multimedia classrooms in the near future.

Usability tests applied to basic school students, enforce the evidence that our application contributes to consolidate knowledge, providing a more enjoyable experience for them. The concepts taught in classroom were presented in an innovative and visual approach (using AR). Also, we observed that the application allows work in a collaborative way, because in order to resolve some tests, they organized themselves to discuss and finally found a solution.

Technology can help us to learn, as long as the application is well designed and clearly focused on the educational objectives to achieve. It is worth pointing out, that AR is not intended to replace a physical model that exists already; rather, it represents another option in which we can visualize abstract concepts or simply we can increase access to knowledge. This work confirm the usefulness of AR as a tool for strengthen teaching-learning process in our study case.

In the future, we will develop an application to present interactive information for museum artifacts, using mobile devices and AR, so we expect they become more attractive to general public. Also, we expected to develop collaborative AR applications for different basic grades and subjects to provide support to the teaching-learning process, reinforcing the constructivist approach for education.

Acknowledgment Authors would like to thank Motorola for donating mobile devices with Android OS necessary for tests. Also to Instituto Bernal Díaz del Castillo for facilitating work with their students. Finally to Professor Mario A Moreno Rocha and the UsaLab for their work in the preparation of the usability testing.

References

1. Blender, `http://www.blender.org`
2. Instituto nacional para la evaluación de la educación (inee): Panorama educativo de méxico 2007: Indicadores del sistema educativo nacional (2007), `http://www.oei.es/pdfs/panorama2007completo.pdf`
3. Instituto nacional para la evaluación de la educación: La calidad de la educación básica en méxico (2007), `http://www.inee.edu.mx/images/stories/Publicaciones/Informes_institucionales/2006/Partes/4o_libro_c_7.pdf`
4. Secretaría de educación pública (sep) : Programas de esdudio 2009, tercer grado (2009), `http://basica.sep.gob.mx/reformaintegral/sitio/pdf/primaria/plan/3Grado.pdf`
5. Sepiensa (2009), `http://www.sepiensa.org.mx/`

6. Cuarto informe de gobierno, 2010: Anexo estadístico 2008-2009 (2010), http://www.presidencia.gob.mx
7. Cuarto informe de gobierno, 2010: Anexo estadístico 2008-2009 comparaciones internacionales (2010), http://www.presidencia.gob.mx
8. Enciclomedia (2010), http://www.enciclomedia.edu.mx
9. Instituto latinoamericano de la comunicación educativa (2010), http://www.ilce.edu.mx/2010/
10. Secretaría de educación pública (sep): Didáctica de los medios de comunicación (August 2010), http://www.sep.gob.mx/es/sep1/sep1_Didactica_de_los_Medios_de_Comunicacion
11. Azuma, R.: A survey of augmented reality (1997)
12. Billinghurst, M., Kato, H., Poupyrev, I.: The magicbook - moving seamlessly between reality and virtuality. IEEE Comput. Graph. Appl. 21, 6–8 (May 2001), http://portal.acm.org/citation.cfm?id=616070.618818
13. Calvo, I.: Herramienta de aprendizaje para el apoyo de las matemáticas de primer grado de primaria utilizando dispositivos móviles (2006)
14. Chen, Y.C.: A study of comparing the use of augmented reality and physical models in chemistry education. In: Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications. pp. 369–372. VRCIA '06, ACM, New York, NY, USA (2006), http://doi.acm.org/10.1145/1128923.1128990
15. Corchado, D.J.: Modelado en 3d y composición de objetos (2002)
16. Dede, C.: The evolution of constructivist learning environments: Immersion in distributed, virtual worlds. Educational Technology 35(5), 46–52 (1995)
17. Dunleavy, M., Dede, C., Mitchell, R.: Affordances and limitations of immersive participatory augmented reality simulations for teaching and learning. Journal of Science Education and Technology 18, 7–22 (2009), http://dx.doi.org/10.1007/s10956-008-9119-1, 10.1007/s10956-008-9119-1
18. Gagne, R., Briggs, L., Wagner, W.: Principles of Instructional Design. Wadsworth Publishing, 3 edn. (1992)
19. Gartner: Forecast: Mobile communications devices by open operating system, 2007-2014 (2010), http://www.gartner.com/DisplayDocument?ref=clientFriendlyUrl&id=1428830
20. Gartner: Gartner says worldwide mobile device sales grew 13.8 percent in second quarter of 2010, but competition drove prices down (2010), http://www.gartner.com/it/page.jsp?id=1421013
21. L. Johnson, A.L..R.S.: The 2009 horizon report. Austin, Texas, The New Media Consortium (2009)
22. Lee, H.S., Lee, J.W.: Mathematical education game based on augmented reality. In: Proceedings of the 3rd international conference on Technologies for E-Learning and Digital Entertainment. pp. 442–450. Edutainment '08, Springer-Verlag, Berlin, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-69736-7_48
23. Medicherla, P.S., Chang, G., Morreale, P.: Visualization for increased understanding and learning using augmented reality. In: Proceedings of the international conference on Multimedia information retrieval. pp. 441–444. MIR '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1743384.1743462
24. Morcillo, C.G.: Animación para la comunicación, práctica 9. Escuela Superior de Informática, Ciudad Real, Universidad de Castilla, La Mancha (2004)
25. Shelton, B.E.: Using augmented reality for teaching earth-sun relationships to undergraduate geography students. In: Students, ART02, The First IEEE International Augmented Reality Toolkit Workshop (2002)

26. Shelton, B.E., Hedley, N.R.: Exploring a cognitive basis for learning spatial relationships with augmented reality. Technology, Instruction, Cognition and Learning 1, 323–357 (2004)
27. Tardáguila, C.: Dispositivos móviles y multimedia, http://mosaic.uoc.edu/wp-content/uploads/dispositivos_moviles_y_multimedia.pdf
28. Villa, M.P.: Por qué surge el palem?, http://www.latarea.com.mx/articu/articu0/palencia0.htm

A Web-Based Platform for Creation of IPTV Contents

Pedro C. Santana[1, 2], Luis Anido[2]

[1]School of Telematics, University of Colima, Mexico
[2] Department of Telematics Engineering, University of Vigo, Spain
psantana@ucol.mx, lanido@det.uvigo.es
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract: With the introduction of Interactive Television (iTV), many digital services have converged to this communication form. In particular, the use of Web based technology as a platform for content creation of iTV transmissions. This paper describes a platform proposal for creation of interactive content for iTV trough the Web.

Keywords: iTV, IPTV, MHP, Web.

1 Introduction

Nowadays our televisions have become an important part of our lives; through television (TV) many people know what is happening in the world and provides them fun and entertainment.

The TV is a common device with a high household penetration, and has a huge impact on virtually all areas, from information to entertainment and education. Therefore, we could argue that TV plays a major role in our society.

However, every day are emerging new forms of digital entertainment, these digital contents are using the Web as delivery platform. Blogs, video blogs, photo blogs, games, chats, news, online journals and instant messaging are some examples of Web applications that provide online content for entertainment and information [1]. With these new tools, users (especially the younger ones) seek challenges (as in games) and participation, therefore, a passive medium as the traditional TV cannot achieve it.

With the advent of the interactive television (iTV), viewers become from being passive players to take a more active role. The iTV term refers to the TV with interactive content and digital enhancements.

iTV combines traditional TV with interactive digital applications that are developed for use in a television set [2].

2 Interactive television in Mexico

The current state of Mexico with respect to the analog-digital transition, is a plan to have the analog switch for 2021 (currently the Mexican president tries to move the date to 2015). Therefore bring the iTV service via digital terrestrial television (DTT) is no viable in the country, that is why this work propose the use of IP Television

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 59-64

(IPTV), because the transmission is held by IP networks and this kind of networks are already operating on Mexico, particularly those provided by cable providers that offer the service called triple-play (view Fig. 1).

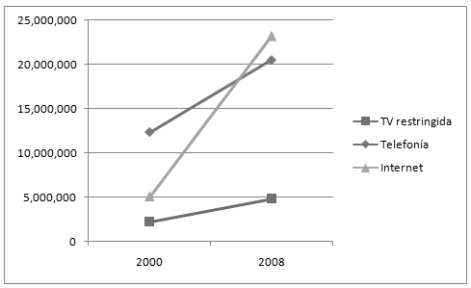


Fig. 1. Percentage growth of triple play users in Mexico.

This paper presents a research in progress which aims to provide a technological solution that allows using the Web as a platform for creation of interactive television's programming, particularly for IPTV.

3 IPTV

Among the different ways to bring interactive content to users via the TV we have the IPTV.

IPTV is digital TV distributed over an IP network instead of a traditional cable network. IPTV is not watch videos on websites, but refers to the way in which information is sent.

The video is sent as IP packets to reach the users [3], [4]. That is, the TV programming is encoded and converted to IP packets. Then, IP packets are distributed through the network to the end user, which with a decoder (set-top box) converts the digital data into analog television signals [3], [5].

The IPTV architecture (see Fig. 2) consists of the following functional components [6]:

1. **Content Source:** A device that receives, encodes and stores (in a database) video content or other sources.
2. **IPTV service node:** A device that receives the video stream and encapsulates it for properly transmission. These nodes enable the delivery of video to the clients.
3. **IPTV client:** It is a set-top box located in the client, which allows the processing functionality.

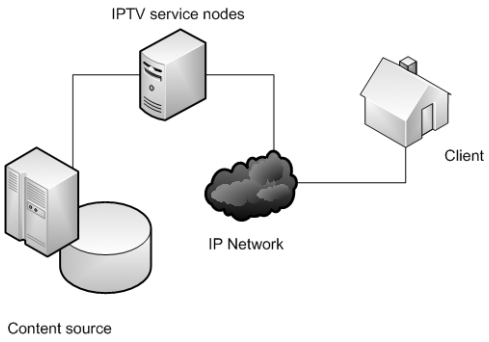


Fig. 2. Generic architecture of an IPTV system

4 The Web as a platform

Although the current set-top boxes provide Web browsers, most of them do not allow users to navigate freely on the web, some of them can only do it in a predefined list of sites that were previously tested for its use in an IPTV environment and those who allow it, may provide a poor user experience due to technical reasons (e.g. resolution of the screen) and lack of capabilities on the Web browsers.

The purpose of this work in progress is to design and develop a Web-based platform that will provide the tools for creating content, as well as a tool to automatically convert our web content to a subset of the XHTML language.

For content creation we propose to develop a multimedia content management system, which should be supported by a multimedia server (MMS) to perform the automatic indexing of the digital files required for the transmissions.

Once the contents have been created and are ready to be transmitted, we propose the use of a tool that is responsible for automatically convert our HTML content to a subset of the XHTML language, which will allow the proper display on a TV set, this tool will serve as a gateway between our CMS and a set-top box. A set-top box is a device that enables a TV set to become a user interface to the Internet and TV. It is the gateway to provide digital information to the home. Its primary function is to decode broadcasted video stream and transmits it to the television set. It also manages interactive applications placed in the video stream beside audio and video signals. The set-top box controls the interaction between the end user and the outside world. It handles user's requests and communicates with content provider through the return channel. In the set-top boxes we can find a technology called Multimedia Home Platform (MHP). The main objective of this technology is to provide interactivity with the transmitted audio/video. This means, MHP is a generic interface between interactive digital applications (those received from the XHTML conversion) and the set-top boxes in which these applications run.

5 Extended architecture

In order to achieve the system functionalities described above, we are proposing the following architecture as it is described in figure 3, the extended architecture includes the XHTML conversion process in order to display correctly the content on a TV set and the content is being transported through the MHP standard, which also serves to provide a return channel for the interactive applications running on the set-top box at the client location.

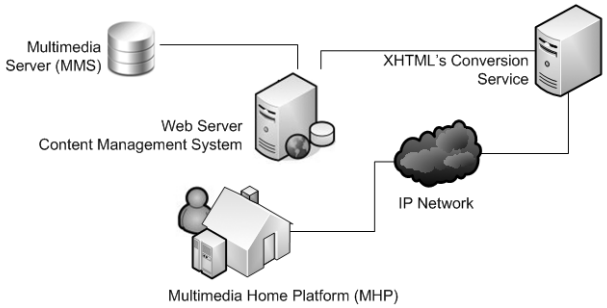


Fig. 3. Proposed extended architecture of an IPTV system.

The extended IPTV architecture (see Fig. 2) consists of the following functional components:

1. **Multimedia Server (MMS):** We will need a strategy in order to manage different multimedia documents (video and audio). Thus, we require a multimedia server component (MMS) to manage the documents used by transmissions. When the CMS requires retrieving a document, it will send the request to the MMS that will act as a gateway to the actual repository that maintain the files.
2. **Web Content Management System (CMS):** The trend of using Content Management Systems (CMS) to manage web content is gaining momentum with the introduction of automated publishing tools that facilitate the publishing process and improve the user experience and usability [7]. We will build our CMS on AJAX and PHP technologies. The Ajax engine, will allow the user to interact with the CMS synchronously by using a web based interface. We also need to implement and API (Application Programming Interface), this API will be using the Service-Oriented Computing (SOC) paradigm [8]; thus, we will use services as the fundamental elements for developing applications. This responds to the need of providing a uniform and ubiquitous information distributor for a wide range of computing devices (such a Tablet PCs, PDAs, mobile telephones, or appliances) and software platforms (e.g., LINUX or Windows).
3. **XHTML's Conversion Service:** The HTML's content generated by the CMS will be transformed into standard XHTML data; since well-formed XHTML documents can be easily managed by low end set-top boxes. Then, media objects, such audio

and video will be transformed in order to be made it MHP compliant before delivering the content to the client.

4. **IPTV client:** It is a set-top box located in the client, which allows the processing functionality of the interactive applications.

6 METODOLOGY

In order to provide adequate support for this platform proposal, we have to understand, from the perspective of those using TV and the Web. Consequently, for the design of our solution we propose to adopt an empirical approach and based it on a combination of interviews and in situ evaluations. The experience designing this system will give us not just a set of well-grounded requirements but, more important, a good understanding of the phenomenon experienced by those TV and Web consumers.

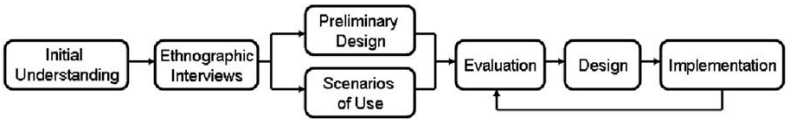


Fig. 4. Research methodology

Therefore, to achieve the research objectives we have selected a methodology based on the user-centered design (see Fig. 4).

6.1 Initial context (understanding, interviews and scenarios of use)

The inquiry to derive the design of the system must be oriented towards understanding the needs of the TV and Web users. We wish to gain knowledge about their experiences in regards to the following main aspects: internet use, TV use and digital content consumption.

We plan to elaborate several scenarios of use to illustrate the systems functionality and conduct interviews to inform the scenarios and to envision a preliminary design of the system.

6.2 Preliminary design

Using the literature review and the scenarios of use, we need to design a software architecture for the development of the system.

6.3 Prototype

In order to evaluate the design, we will build a prototype of the solution that would be evaluated by potential users.

Results of the prototype evaluation will enable us to improve the design and consolidate a final and more complete solution.

6.4 Evaluation

We propose to evaluate a functional prototype by potential users to gain feedback from all perspectives. The goal of the evaluation would be to explore the feasibility of the solution as well as its appropriateness. We will expect that participants, while evaluating the prototype, will raise more specific issues that would serve to refine the solution and, in general, the understanding of the challenges they face in their day by day.

6.5 Design and implementation

Based on an analysis of the data collected during the evaluation we will identify results both with respect to the system and with respect to potential users. Finally, we plan to redesign and develop the final system using the data collected on the evaluation.

7 Conclusions

This work in progress is proposing the design and development of a Web platform that facilitates the creation of interactive content for iTV transmissions, this contents will be distributed towards the Multimedia Home Platform standard in order to achieve interactivity in addition to facilitate the use of the return channel.

8 References

1. A. Gil, J. Paz, C. Lopez, J. Lopez, R. Rubio, M. Ramos, R. Diaz, “Surfing the WEN on TV: the MHP approach”, in Proc. of the International Conference on Multimedia and Expo, Lausanne, Switzerland, 2002.
2. C. Herrero, P. Cesar, P. Vourimaa, “Delivering MHP Applications into a Real DVB-T Network, OtaDigi”, in Proc, of Telecommunications in Satellite, Cable and Broadcast Service (TELSIKS 2003), Serbia – Montenegro, 2003, 231-234.
3. J. CORREA. UNE-EPM, “con TV por Internet desde marzo”. Portafolio. El Tiempo. Bogotá, 21, febrero, 2007.
4. J. WALKO. “I love my IPTV”. IEEE Communications Engineer. Vol.3, Nº 6. dic. 2005; p.16-19.
5. R. WARD. Internet Protocol Televisión. New Tech Briefs, dic. 2004.
6. JL Mauri, MG Pineda, FB Seguí, “IPTV: la televisión por internet”, Editorial Vertice. ISBN: 8492647221
7. Pedro C. Santana, Victor M. Gonzalez, Marcela D. Rodríguez. “Codice CMS: Towards a Multimedia Weblog Content Management System for Supporting Mobile Scenarios”. In the proceedings of the 4th Latin American Web Conference, Cholula, Mexico, October 2006.
8. Papazoglou, M.P. Service -Oriented Computing: Concepts, Characteristics and Directions. in Fourth International Conference on Web Information Systems Engineering. 2003: IEEE Computer Society.

Merging Technologies Models for Indoor Mobile Device Positioning

Erick Alonso Salazar Molina[1], Jose Martin Molina Espinosa[1]

[1] Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Ciudad de México, Calle del Puente 222, Tlalpan, 14380, México City, México
{A00969900, jose.molina}@itesm.mx
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. In this work, we discuss the advantages and disadvantages of several indoor positioning systems (IPS) and compare them in terms of designing services for users. We propose a set of critical factors for IPS evaluation. The main contribution consists of the proposal of some merging techniques models using a wireless technologies hierarchy to estimate the position of mobile device in indoors environments. Finally we present a study case using the proposed models to locate objects in indoor areas**.**

Keywords: Indoor Positioning Systems, context aware location, mobile devices positioning.

1 Introduction

Since wireless access from mobile devices is now widely available, there is great demand for precise positioning in wireless networks, including indoor and outdoor environments. The process of determining a location with wireless technology is called location detection, geographic location, or position location. Different applications may require different types of location information. The main types discussed in this paper are physical location, symbolic location, absolute location, and relative location. Physical location is expressed in the form of coordinates, which identify a point on a map of two or three dimensions. Symbolic location defines a location using natural language, e.g. in the boardroom, in Building A, on the fifth floor of the tower, etc. The absolute location system uses a common reference for all located objects. A relative location depends on a custom frame of reference. The location information is usually based on proximity to known reference points or base stations. While cases of location in outdoors have been well treated by GPS, mobile device positioning inside buildings poses special problems. First, the GPS does not work under a roof, because it requires line of sight (LOS) between transmitter and receiver. Second, many interest problems related to indoor location tracking demand for better accuracy and precision.

The use of RF signals is not the only option for indoor location tracking. Programs based on infrared signals, acoustic signals, pressure sensors embedded in the ground,

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 67-76

have been proposed and implemented. RF technology has the advantage of simplicity and low cost, especially with the proliferation of low cost wireless networks.

2 Location Technologies and Algorithms

It is difficult to model the propagation of radio signals in indoor environments, due to various RF signal multi-paths, low probability in the availability of line of sight (LOS), and site-specific parameters such as architecture and design materials, objects that change position, objects in motion, and reflective surfaces. To date, there is not a good model to address the characteristics of multipath indoor radio signals [2] [14]. Indoor Position Systems (IPS) use positioning techniques to locate objects and provide absolute, relative and proximity position information. There are three main techniques for estimating indoor position: 1) triangulation, 2) analysis of scenarios (fingerprint), and 3) proximity [3]. Triangulation, and fingerprints techniques can provide absolute, relative and proximity position information. The proximity positioning technique can only provide information with respect to reference points.

Several IPS use a combination of various techniques to compensate technical limitations due to the use of a unique positioning method [15]. Location algorithms have been specifically designed to define how to calculate the position of a target object. For example, in the technique of triangulation, when the distance between a target object and each reference point is obtained (at least using 3 references), the location algorithm calculates the location of the object.

2.1 Triangulation.

On the basis of the geometric properties of triangles, RSS, AOA and TOA can be used to calculate the position [1]. If the coordinates (xi, yi) of three reference points A, B, C, are known, the absolute position MS1 can be calculated using either the length or the directions R1, R2 and R3 [4]. Each triangulation method has its advantages and limitations. TOA is the most accurate technique, which can filter out multipath effects in indoor environments. However, it is complex to implement [4]. RSS and TOA need to know the position of at least three reference points to estimate the position of a target object. AOA requires only two reference points to determine a target object. However, when the target object is too far away, the AOA method may contain errors, which will result in less accuracy [11].

2.2 Scenario Analysis (fingerprints)

RF scenario analysis refers to the type of algorithms that first collect the identifying characteristics or mapping data (fingerprints) of a scene and then estimate the location of an object by matching the measurements with the fingerprint previously recorded. RSS is the preferred technique for when using a fingerprint location method. Positioning using scenario analysis refers to techniques that match the "fingerprint" of

some signal property that depends on its physical location. There are two scenarios for fingerprint positioning: the offline scene and the online scene (runtime scenario).

Location coordinates (labels) and signal strengths (measurement units) for every base station are collected. During the online scene, a location positioning algorithm uses the signal strength observed during the time and the information collected above to calculate the estimated location. The main challenge for location-based techniques (fingerprints) is that the intensity of the received signal may be affected by diffraction, reflection and diffusion of RF signals during their propagation in indoor environments.

There are at least five localization algorithms based on pattern recognition techniques: probabilistic methods, k-nearest neighbors algorithm (k-NN), neural networks, Support Vector Machine (SVM), and the Smallest M-vertex polygon (SMP).

2.3 Proximity

The proximity location technique examines the target object location with respect to a reference point. Proximity location technique is based on a set of detectors of known positions. When a sensor detects a tracking object, the object´s position is considered to be within the area marked by the proximity detector. When more than one antenna detects the moving target, then it is considered in the area whose the antenna is receiving the strongest signal [14].

3 Critical Factors for IPS

To evaluate the indoor positioning systems, we had proposed a set of performance and implementation factors. Performance on IPS is mostly defined on accuracy and precision.

Accuracy. Accuracy or location error is the most important requirement in positioning systems. In general, the average distance error is taken as a measure of performance, which is the average Euclidean distance between the estimated location and true location. The greater the accuracy, the better the system, however, there is often a tradeoff between solution accuracy and other features. *Precision.* Precision is defined as the probability of success of position estimation with respect to the predefined accuracy. This factor is a measure of the strength of the technique of positioning, as it reveals the variation in performance on several tests. *Complexity*. The complexity of a positioning system can be attributed to hardware, software, and operating factors, as well as human intervention during its implementation and maintenance. *Robustness.* A robust IPS should be able to keep running even if some signals are not available, or when some of the RSS and AOA values are not presents, when the device is locked, or a mobile device run out of battery. The only information to estimate the signal direction is from other units of measurement. *Scalability.* Defined as the number of mobile devices that an IPS can locate using certain size and infrastructure within a period of time. Usually positioning performance degrades when the distance between the transmitter and receiver increases. A location system

must scale along two axes, the geography and density. *Cost.* The cost of a positioning system may depend on many factors. Important factors include money, time, space, weight and energy [14].

4 Indoor Positioning Systems

Infrared (IR) positioning systems [1], [5] can provide absolute position but it needs a line of sight communication between transmitters and receivers [5]. The IR signal is affected by fluorescent light and sunlight. So, the range of coverage devices is limited to a room. *Ultrasound,* for indoor applications, the ultrasonic positioning systems have emerged using a combination of RF and ultrasound technologies [1], [6]. The mode of operation is setting a number of nodes installed on walls and ceiling, they transmit their location information via RF signals while emitting ultrasound pulses, which are synchronized with the RF signals [15]. *Radio Frequency Identification (RFID)* is a method for storing and retrieving data through electromagnetic transmission [7]. RFID positioning systems are commonly used in indoor complex environments. RFID as a wireless technology allows flexible and inexpensive identification of a mobile device or a person [8]. *Wireless local area networks (WLAN),* WLAN technology is very popular and has been implemented in WiFi access in public areas. Indoor positioning systems based on WLAN reuse existing infrastructures, which reduce its cost. The accuracy of WLAN IPS is estimated based on the received signal strength. RSS is affected by various elements in interior environments such as persons or object movement and orientation, overlapping Access Points (APs), etc. *Bluetooth,* The IEEE 802.15.1 standard is a specification for wireless person area networks (WPAN). The position of a Bluetooth mobile device is located through the efforts of other mobile terminals in the same group (cluster). Bluetooth IPS is a low cost technology that can use actual devices already equipped with Bluetooth technology [9]. *Sensor networks,* IPS based on sensors consists of a large number of sensors attached to predefined locations [10]. Sensor-based IPS provides a cost effective and convenient way for locating people and devices due to the decrease price and size of the sensors. *Ultra wideband (UWB),* RF positioning systems suffer from multipath distortion of radio signals reflected from the walls in indoor environments. The pulses of ultra wideband (UWB) [11], which have a short duration (less than 1 ns), allow filtering out reflected signals from the original signal, offering greater accuracy. UWB technology offers several advantages over other IPS technologies, it does not require line of sight, no multipath distortion, less interference, high penetration, etc [12]. *Magnetic,* the use of magnetic signals is an ancient and classical measurement of the positioning and tracking [13]. Magnetic positioning systems offer high accuracy and do not suffer from the problems of line of sight. However, magnetic systems have limited coverage range, which decreases location performance [15]. *Cell Identification (Cell-ID),* several systems have used GSM / CDMA technologies from cellular networks to estimate the location of mobile customers on the field. Mobile device location is based on cell identification implemented by the network, its main advantage is that it works for all phones and requires no modifications to the mobile phone. However, the accuracy cell identity

(Cell-ID) or Enhanced Observed Time Difference (E-OTD) methods is generally low (in the range of 50-200 m), depending on size cell. [14].

Table 1 shows a comparison among several indoor positioning technologies with some important features, including different algorithms that are used for location.

Classification	Accurrancy	Precision	Robustness	Complexity	Cost	Systems
WLAN (kNN)	3-5 m	50% on 2.5m	Medium	Medium	Lowo	RADAR
(SMP)	3 m	50% on 2.7m	Medium	Medium	Medium	MultiLoc
(PM)	2 m	90% on 2.1m	Medium	Medium	Low	Horus
(Bayesian)	1.5 m	50% on 1.5m	Medium	Medium	Medium	Robot-base
(PM)	1 m	50% on 2m	Medium	Medium	Lowo	Ekahau
RFID (kNN)	< 2 m	50% on 1m	Low	Medium	Low	Landmarc
UHF (LS)	2-3 m	50% on 3m	Medium	N/D	Low	WhereNet
UWB (LS)	< 0.3 m	50% on 0.3m	Low	N/D	High	Sappire Dart
(LS)	0.15 m	99% on 0.3m	Bajo	N/D	Alto	Ubisence
Celular (kNN)	5 m	80% on 10m	Medium	Medium	Medium	GSM Fingerprinting
BT / IR (PD)	2 m	95% on 2m	Low	N/D	Medium	Topaz

Table 1: Indoor Positioning Technologies Comparison

5 Merging Technologies Models

The availability of accurate information about the location of a particular mobile device enables mobile application development, value added in many vertical markets. For outdoor applications, GPS and positioning systems based on GSM offer performance and cost suitable for most applications. Unfortunately they do not work well indoors, for GSM location there could be arange error >100m. This has led to research other technologies that are becoming more and more available on ordinary mobile devices.

5.1 Mobile device location for contextual applications.

Contextual applications require knowledge about the position of an entity (user, mobile device, etc.) in order to offer operations, services, information. These applications do not need to interact with an estimate of position coordinates, but rather with a symbolic position, which understandably determine the location of the user [17]. Symbolic position is used because the final services are related to a certain level of information granularity, which means that although the position estimation is made for physical coordinates from the point of view of context applications these coordinates means areas, buildings, etc. An area can be estimated through simple infraestructure deployment, taking advantage of the capabilities of cell identification with many technologies (GPS, cellular, WiFi, Bluetooth, etc.). It is possible also to emply short-range technologies located at strategic crossing points (eg, RFID).

In the framework of context-aware applications, there are some features: there is not a need to obtain a very accurate location, most of the time it is sufficient to determine that a mobile device is within a specific area. Only for certain situations that require high accurate locations it is necessary to used another method, such as coordinates for example. This concept of cooperative and complementary use of different technologies to ensure a quality of context positioning, with a rational use of resources, is refered as "fusion of technologies".

5. 2 Fusion of technologies for location.

We have discussed some techniques of cellular and multicellular positioning, supported by different technologies, that can be used to estimate the position of mobile devices. The reason that these technologies have been described are: a) they are available, and increasingly, on mobile devices (Bluetooth, WiFi and RFID / NFC). b) They are now accessible through network infrastructure. However, deployments and normal use of these individual technologies in general do not ensure a minimal quality of service for location. Some limitations are: low postion accuracy, reduced scope coverage, roaming availability. These and other limitations of individual technologies can be solved by using a cooperative / complementary formed by several technologies.

The use of fusion or merging technologies can face major constraints on the use of individual technologies (Fig. 3) in relation to the four constraints mentioned previously we can observe the following: *The limitation on accuracy:* can be overcome by integrating the estimates provided by other technologies. *The limitation in coverage* can be improved by being able to substitute some other technologies to complete a successful deployment. For example, if the primary positioning system is WiFi, and at some point in space is losing coverage, the fusion system can try: 1. maintain the position of the user thanks to the inertial available on your mobile, 2. location on the base provided by a NFC / RFID sensor network, 3. seek close collaboration with peers ability to share his position, etc. *The limitation on the continuity and availability*, using redundant or alternative two or more technologies. *Calibration requirements* for a technology can be met by another.

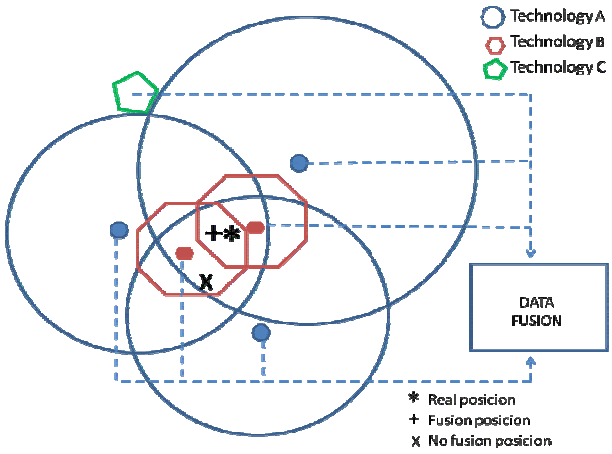


Fig. 3 Merging technologies

The possibilities are many and it is not to propose a redundant and costly deployment or use of complex algorithms as it is not necessary. A target to design a model for positioning, includes an abstraction of the technologies that are being handled, in this work we proposes two sets of hybridization / fusion, selected on the basis of application requirements. Model 1 combines two technologies WiFi multicellular and Bluetooth, wich may be combined in three different ways, cooperative, competitive or complementary. Model 2 combines a medium-range multi-cellular technology WiFi beacons in RFID checkpoints.

Model 1: WiFi and Bluetooth. Both WiFi and Bluetooth can be used as multi-cellular location technologies based on RSS. Different characteristics of coverage, functionality and deployment guide, make it possible to regard them as suitable to be merged technologies, especially to improve the accuracy and coverage issues [16] [17] (Fig. 3). The two technologies, infrastructure mode or terminal based, are able to measure the RSS of a series of beacons, access points/ devices collaborators. Depending on the coverage of technology, mobile devices, can be located in different ways.

According to the classification of fusion methods Durrant-White, one can infer three basic procedures that can improve the individual accuracy possible for separate technologies: *Fusion complementary, fusion competitive and fusion cooperative.*

Model 2, WiFi and Micro RFID cooperative Technology. The presence of geolocated microcells in which users enters or whom they interact may allow estimates to calibrate other technologies. In this perspective, a microcell is a small region in which the user is precisely located, at this point the concept of precision is relative. We can consider the RFID / NFC proximity zones where users can interact (like opening a door etc.) or markers placed at crossing points. This zones establish reference positions whichs relate to measurements taken with other technologies. But it could also be other microcells geometrically referenced on the map room.

The fundamental limitation of multicellular WiFi location model is the variations in the electromagnetic environment, decreasing their accuracy. Depending on the method used for positioning, it is possible to use a mapped database, describing electromagnetic situation depending on the state of the environment. Figure 5 gives a

simple representation of the stage. The mobile is equipped with a NFC reader and the cards are deployed on well known locations of the environment that encourage user interaction (PCs, point-of-way or concrete objects). The cards are associated with a position. When the mobile device reads a card, it sent position data and signal strength measurements that capture WiFi access points.

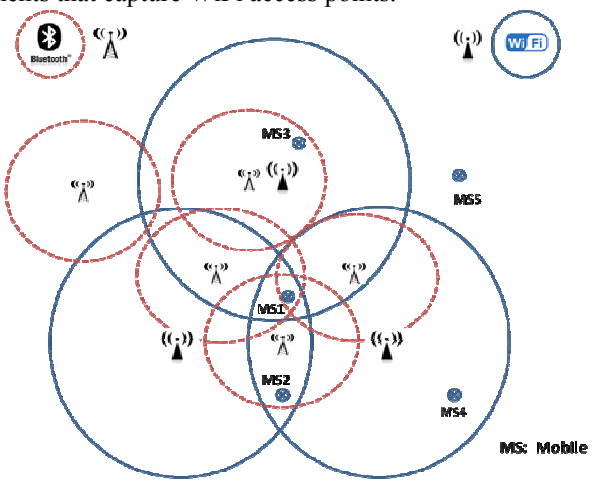


Fig. 4 Model 1, merging technologies WiFi – Bluetooth

6 Case of Study

We have tested our merging models in a scenario for mobile device location within a university campus. The scenario consists of locating a book withitin the library campus using a mobile device. The study case is proposed with hierarchy of areas, associated technologies and positioning methods.

The design of the positioning system must provide a solution which achieves the desired granularity, from the stated minimum aimed to find out if the user, via a mobile device, is or is not on campus, to the maximum, which is the detection if front of the shelf on which is located the desired book. For which we propose the following technical solution.

First Level, determine whether the user is or is not on campus. Solution, check device's connectivity WiFi and keep track of access doors in / out of campus in the door sensor. Second Level, determine the area of campus where the user is located. Solution, use the multicellular system of campus WiFi technology that allows us to determine on which floor of the library the user is located, with which there is at least three WiFi access points at each level. Third Level, determine specific areas of books by subject. Typically these areas will be associated with some shelf. Solution, the solutions to this level will be dependent on the shelves whose area of action aspires to be. The general solution of discovery will be the implementation of a microcellular technology such as ZigBee technology combined with proximity waypoints, or even only the latter. Fourth Level, association of presence with the use of an object. This

level is specifically considered to detect the book (objective) in question, such as intelligent devices that emit a buzz. Solution, in general, the answer would be to use a proximity technology, which often come included in the same subject.

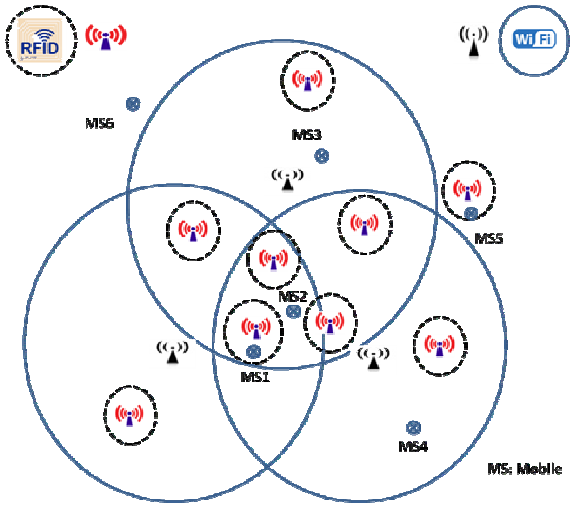


Fig. 5 Model 2, merging technologies WiFi – RFID

All these technological solutions require the user to carry a mobile device with location technologies or more mobile devices that allow the location. The system will keep track of hierarchical position, to anticipate potential communication with sensors in the active area. That is, if the user is in the area of electronic books, the system will activate the ZigBee motes for the various points of interest established to refine the position calculation. With additional background information, this activation can be made smarter, for example, the system can display the date of publication, thematic index, etc.

Conclusions

In this paper, we present an IPS indoor technology comparison made on the proposal and definition of five key factors on IPS performance: accuracy, precision, robustness, complexity, scalability and cost. IPS technologies employed as monolithic solutions present limitations on several key factors that limits their quality of service. We have defined two merging IPS technologies models to determine location of mobile device in indoor locations. Using a combination of several IPS technologies made and enhance in terms of accuracy, precision and robustness. The first model merges multi-cellular approaches based on WiFi and Bluetooth. The second model merges multi-cellular with micro-cellular approaches based on WiFi and RFID.

Finally, we have tested our models in a scenario where the objective is the localization of a book on a library university campus.

References

1. J. Figueiras and S. Frattasi, ”Mobile positioning and tracking, from conventional to cooperative techniques”, Wiley, 2010.
2. K. Pahlavan, X. Li, and J. Makela, “Indoor geolocation science and technology,” *IEEE Commun. Mag.*, vol. 40, no. 2, pp. 112–118, Feb. 2002.
3. J. Hightower and G. Borriello, ”Location sensing techniques”, Technical Report UW CSE 2001-07-30, Department of Computer Science and Engineering, University of Washington, 2001.
4. K. Pahlavan, X. Li, and J. Makela, “Indoor geolocation science and technology,” *IEEE Commun. Mag.*, vol. 40, no. 2, pp. 112–118, Feb. 2002.
5. R. Casas, D. Cuartielles, A. Marco, H. J. Gracia, and J. L. Falc, ”Hidden Issues in Deploying an Indoor Location System”, *IEEE Pervasive Computing*, vol. 6, no. 2, 2007, pp. 62-69.J.
6. H. Piontek, M. Seyffer, and J. Kaiser, ”Improving the Accuracy of Ultrasound-Based Localisation Systems”, *Proc. International Workshop on Location-and Context-Awareness*, Berlin, Germany, 2005.
7. L. M. Ni and Y. Liu, ”LANDMARC: Indoor Location Sensing Using Active RFID”, *Proc. IEEE International Conference on Pervasive Computing and Communications*, 2003, pp. 407-416.
8. H. D. Chon, S. Jun, H. Jung, and S. W. An, ”Using RFID for Accurate Positioning,” *Proc. International Symposium on GNSS*, Sydney, Australia, December, 2004.
9. M. Rodriguez, J. P. Pece, and C. J. Escudero, ”In-building location using Bluetooth”, *Proc. IWWAN*, 2005.
10. D. Niculescu and R. University, ”Positioning in Ad Hoc Sensor Networks”, *IEEE Network Magazine*, vol. 18, no. 4, July/August 2004.
11. J. Ingram, D. Harmer and M. Quinlan, ”UltraWideBand Indoor Positioning Systems and their Use in Emergencies,” *Proc. IEEE Conference on Position Location and Navigation Symposium*, April 2004, pp.706-715.
12. Y. Zhang, W. Liu, Y. Fang, and D. Wu, ”Secure localization and authentication in ultra-wideband sensor etworks”, *IEEE J. Select. Areas Commun.*, vol. 24, no. 4, 2006, pp. 829-835.
13. F. Raab, E. B. Blood, T. O. Steiner, and H. R. Jones, ”Magnetic Position and Orientation Tracking System”, *IEEE Trans. Aerospace and Electronic Systems*, vol. AES-15, no. 5, September 1979, pp. 709-718.
14. Hui Liu, Houshang Darabi, Pat Banerjee, Jing Liu, "Survey of Wireless Indoor Positioning Techniques and Systems", IEEE Journal 2007.
15. Yanying Gu, Anthony Lo, *Senior Member, IEEE,* and Ignas Niemegeers "A Survey of Indoor Positioning Systems for Wireless Personal Networks", IEEE Journal 2009.
16. Aparicio S., Casar J.R., Pérez J., Bernardos A.M.,. “A Fusión Method Based on Bluetooth and WLAN Technologies for Indoor Location”. Proceedings of the International Conference in Multisensor Fusion and Integration for Inteligent System. IEEE Computer Society (2008)
17. Bernardos Borbolla Ana. “Provision de servicios moviles basados en localizacion y contexto”. Universidad Politecnica de Madrid. 2008.

Modeling Intelligent Agents in Virtual Worlds

Israel Guzmán-Pérez[1], Darnes Vilariño-Ayala[1], María J. Somodevilla-García[1] and Ivo H. Pineda-Torres[1],
[1] Benemérita Universidad Autónoma de Puebla
ig89852005@hotmail.com, {darnes, mariasg, ipineda}@cs.buap.mx
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. Virtual worlds become more popular day by day. Today in these virtual worlds are carried out various activities (cultural, educational, entertainment, leisure among others). So it is necessary to develop intelligent software entities within these environments, with the aim of providing various services to users. The purpose of this paper is to propose a model that is able to meet the needs virtual environment. Besides, a case study was presented to implement the proposed model on an avatar driven by an agent to teach the abacus, in the virtual world of Second Life.

Keywords: Virtual World, Second Life, Intelligent Agent, Avatar.

1 Introduction

For many people virtual world comes to be a work place or a place for knowledge acquisition or for entertainment. Universities have joined to the virtual worlds by offering common spaces to students and professors for education-learning process [1].

A metaverse is a virtual world. Second Life (SL) [2] is a metaverse developed by Linden Research Inc.. It was launched in 2003 and from 2006 has gained a crescent international attention. SL was inspired by "Snow Crash" science fiction novel written by Neal Stephenson and the literary movement "Cyberpunk" [2].

Agents' theory is booming from the last days and it is inside of education-learning process, financial and TAC CAT y TAC SCM [3] market simulations and information retrieval among others.

One of the most ambitious projects of the last years is the Edd Hifeng [4] and it is a result of the intelligent software development incorporation to the virtual world. A group of researchers from Rensselaer Polytechnic Institute working to change that by engineering characters with the capacity to have beliefs and to reason about the beliefs of others. The characters will be able to predict and manipulate the behavior of even human players (avatars), with whom they will directly interact in the real, physical world, according to the team [5].

There are now intelligent agents in virtual worlds, but each agent has its own model trying to fill a need or trying to provide a specific service, the intention of this paper is to propose a model which considers the most needs or services and from which it is easy to develop intelligent agents in virtual environments.

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 77-88

2 Background

Virtual world environments, such as Second Life have been around for some years now. After the initial hype that virtual worlds would unleash new unlimited commercial success, the focus is now on more pragmatic and serious virtual environments applications. Several authors have identified a classification for how both profit and non-profit organizations can use virtual environments, including areas such as marketing, support for mass customization, virtual markets/shopping, communication and collaboration, consumer research, innovation, virtual education and learning, and recruitment [6]. RunAlong is developing an international web community, primarily for women joggers, that is entirely designed through a user-driven innovation approach based on a series of physical user innovation workshops [7].

The largest existing worlds include South Korea's Cyworld, and its 20 million uniques, World of Warcraft with its 8 million subscribers, and Europe's Habbo Hotel with its own 7 million regular users. Also, 26 million online world users by 2011 in China alone are estimated [8]. 18 million accounts were registered in Second Life in January 2010, although there are no reliable figures for actual long-term consistent usage [9] [10].

Agents in Metaverse, built using Active Worlds, are capable of performing the tasks typically associated with human beings, such as taking tickets for rides and acting as shopkeepers. However, these agents are basically reactive agents which work in a hard-coded manner. Virtual psychotherapist ELIZA [11], designed to take care of the 'patients', is also achieved with rule-based, adeptly modeled small talk. A conversational virtual agent Max has been developed as a guide to the HNF computer museum, where he interacts with visitors and provides them with information daily [12].

Agent in the virtual world capable of providing a kind of agency for collaborating with other agents and interacting with the virtual world, and on the other hand acts as a Design Agent capable of designing and constructing dynamic virtual places for the Avatar agent as needed.

3 Proposed Model

A general model was designed so that agents to manage avatars within a virtual world can be developed following this methodology. The proposed model is a generalization of previous works on intelligent agent on Second Life [13][14].

The architecture of the proposed model is shown in figure 1 and comprises five basic modules as follows:

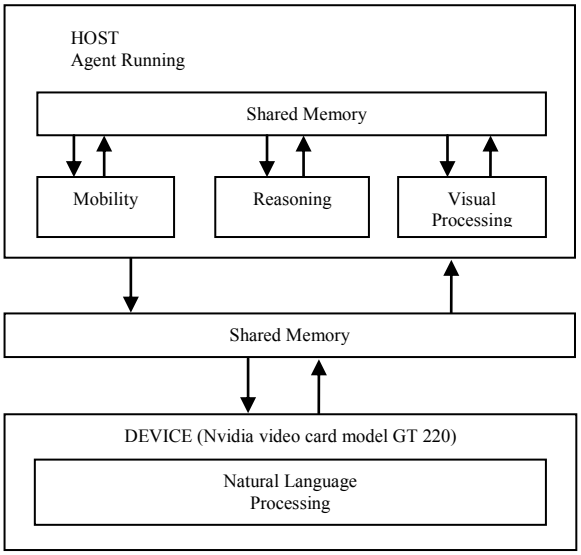


fig. 1. Model General Architecture.

Module 1 Natural Language Processing (NLP): Agent responds messages from other avatars, according the situation.
Module 2 Reasoning: Agent should be able to determine which action to carry out.
Module 3 Visual Processing: Agent processes information coming from virtual world, in order to determine the direction to be taken for the avatar which it is controlling.
Module 4 Mobility: It is used for visual processing; the agent must see what is in their environment.
Module 5 Skill(s): In this module, the agent must possess the necessary behaviors to perform a certain task (Example: teaching, dancing, public address, find information, build things, etc.). The implementation of this module can be composed of more modules, where each is designed to perform a particular task.

3.1 Modules description

Module 1 NLP
Agent should fulfill the following:

a. Text analysis
b. Audio analysis
c. An initial knowledge base and capacity to grow.
d. Analysis of sources such as internet (Google, Wikipedia, Yahoo, etc ...).

Module 2 Reasoning
Must have certain essential features such as:

a. Analysis of the environment in real time.
b. Analysis of the current target.

c. Analysis of the procedure for carrying out the objective (point b).
d. Feedback from the procedure being carried out.
e. Ability to change the procedure if the objective is not being met.

Module 3 Visual Processing

a. Analysis of the environment in real time.
   - Identification of avatars.
   - Identification of objects.
b. Analysis of real time target.
   - Locate avatars in the environment.
   - Locate avatar in the environment.

Module 4 Mobility

a. Define the displacement on the axes (X, Y) or (X, Y, Z).
b. Draw a path to the target.
c. Ability to change the way towards the target.

Module 5 Skill(s)

a. Define the skills set that the agent can develop:
b. $S = \{S_1, S_2, \ldots, S_n\}$.
c. Define the actions set that allow them to develop each skill defined:
d. $Action_j = \{Act_1, \ldots, Act_k\}\ j \in H, k \in [1, \ldots, |Action_j|]$.
e. Interact with the 4 remaining modules to carry out the corresponding actions.

4 Case Study

As a case study Israel Agent is presented. This was designed with the purpose of teaching other avatars, controlled by humans, the basic abacus operations.

4.1 Agent's modules

4.1.1 Natural language processing module

This module interacts with a knowledge database in order to process natural language. A class to permit the connection with the database was created. This class defines methods for receiving and sending messages in the virtual world.

NLP module makes a conversion from text to voice (only in Spanish), thus the agent is endowed with voice, and it makes http connections to Google and DRAE (Dictionary of the Real Academic Spanish).

4.1.2 Reasoning module

Agent Israel applies deductive reasoning to react to diverse avatar's behavior. Logic is used to codify a theory which establishes the best action to be taken for a given

situation. A set of rules and a logic database were defined in order to describe the world current state and the set of actions to be performed by agent (see Table 1 for a complete set of predicates).

Table 1. Israel agent predicates.

Predicate	Description
LocateI(Isle)	Agent is located in an (Isle)
Locate(X,Y,Z)	Agent is located in (X,Y,Z)
Facing(d)	Agent is facing to direction (d)
Teach(j)	Agent is teaching abacus' operations to an avatar (j)
Leisure()	Agent is in leisure state ()
Move(X,Y,Z)	Agent is moving to (X,Y,Z)
MoveI(Isle)	Agent goes to the (Isle)
Search(j)	Agent searches an avatar (j)
Identify(u)	Agent identifies objects or avatars (u)
RMessage(m)	Agent receives messages from avatars(m)
SMessage(n)	Agent send a message to other avatars(n)

Using predicates in Table 1 possible actions of the agent are defined as follows:

Act = {move_forward, move_back, turn, write_Message, send_Message, teletransport, interact, adition, subtraction, multiplication, division}

Nine rules were defining using previous actions of the agent (see table 2)

Table 2. Israel agent rules.

Number	Rule
1	LocateI (I) ^ Leisure() ^ ¬Search(A) ^ ¬Identify(A) > Do(teletransport)
2	LocateI (I) ^ ¬Leisure() ^ Search(A) ^ ¬Identify(A) ^ Move(X,Y,Z) > Do(move_forward)
3	LocateI (I) ^ ¬Leisure() ^ Search(A) ^ Identify(O) ^ Move(X,Y,Z) > Do(move_back)
4	LocateI (I) ^ ¬Leisure() ^ Search(A) ^ Identify(O) ^ Move(X,Y,Z) > Do(turn)
5	LocateI (I) ^ ¬Leisure() ^ Identify(A) ^ RMessage(M) ^ Facing(D) ^ Locate(X,Y,Z) > Do (write Message)
6	LocateI (I) ^ ¬Leisure() ^ Identify(A) ^ Locate(X,Y,Z) ^ Teach(Op) ^ SMessage(MA1) ^ RMessage(M) > Do(sum)
7	LocateI (I) ^ ¬Leisure() ^ Identify(A) ^ Locate(X,Y,Z) ^ Teach(Op) ^ SMessage(MA2) ^ RMessage(M) > Do(subtraction)
8	LocateI (I) ^ ¬Leisure() ^ Identify(A) ^ Locate(X,Y,Z) ^ Teach(Op) ^ SMessage(MA3) ^ RMessage(M) > Do(multiplication)
9	LocateI (I) ^ ¬Leisure() ^ Identify(A) ^ Locate(X,Y,Z) ^ Teach(Op) ^ SMessage(MA4) ^ RMessage(M) > Do(division)

where:

^ means "and".

∨ means "or".

> means “then”.

also:
I = {I1, …, In } accessible isles
A = {A1, … , Am} visible avatars
(X,Y,Z) = {(0,0,0),…,(a,b,c) } possible coordinates inside an isle
O = {O1, …, Ok} accessible objects inside an isle
M = {M1, …, Mp} messages from public channel
D = {north, south, east, west, northeast, northwest, southeast, southwest}
Op = {sum, subtract, multiplication, division} abacus possible operations
MA1 = {teach to sum}
MA2 = {teach to subtract}
MA3 = {teach to multiply}
MA4 = {teach to division}

4.1.3 Visual processing module

Agent Israel resides in the server and continuously tries to detect an avatar arrival in order to begin the learning process. SL provides some sensors to establish the agent‘s visibility range. These sensors return an avatars and objects list found in a predetermined range.

4.1.4 Mobility module

Avatar should be continuously moving inside a determined action range. In doing that, it moves from an initial point within a prefixed angle.

4.1.5 Skills module

Skills and actions set are defined as follows:

S_i = {*use of the abacus*} i = 1.

To achieve complete fully with any basic operation on the abacus, it performs the following actions:

ActionSet = {*establish representation on the abacus, receive message, retrieve message, detect the type of operation received, detect the basic operation to be performed*}, j =1,…,5.

In order to perform the above actions the algorithm 1 should be applied.

Algorithm 1

```
Step 1: At first the agent sets the quantity represented on the abacus
to 0.0
Step 2: The agent receives and retrieves the message (message, type).
  If (type = abacus)
     Retrieve value (bead)
     Update environment variables
  Else
     Using NLP module,
```

```
  Analyze the message
  Determine operation type
   (addition, subtraction,
    multiplication, division or
    representation)
If (operationtype=representation)
    Extract from the message the
    number to represent it, and
    Then
      reset back the abacus.
     Else
      represent the operation, by
      moving the bead and follow
      the abacus rules
```

To understand the movement of the beads in the abacus (specified in step 2 of algorithm1), you need to know that it is divided into two sections [15]. It has a section that only has two balls, called heaven-beads, and another section which has more than 5 balls, known as earth-beads. Importantly, the abacus has 49 beads, spread over 7 columns. Each earth-beads in the first column has a value of 0.01, and each heaven-beads a value of 0.05, the second column represents the 0.1 to 0.5 for earth-beads and heaven-beads. To perform the operations of addition and subtraction are used 17 rules, some of which have been modified to use the maximum number on the abacus (166,666.65), adding the use of end heaven-beads and end earth-beads. The same problem arises when performing subtraction, multiplication and division so they make similar changes to the general rules [14].The third column represents the unity; and so on both types of beads increase their value by multiplying by 10. One important thing to note is, that the first time that the agent detects that it will carry out an operation, i.e. addition, subtraction, multiplication or division, is preparing to retrieve the operator (s) and / the operator (s), operator can be given in numerical or written form. So that, the agent can interpret what is written, a table is generated, only once, and then maintained throughout the life cycle of the agent. In this table are stored numbers from 0 to the 9, and a mechanism is created for generating the rest of the numbers.

4.2 Experimental results

To debug the agent, OpenSim [16] server was installed because of its resemblance to Second Life and the final tests were executed inside Second Life agent having the same results within OpenSim. In a computer OpenSim server was installed and it was configured everything necessary for proper operation. Another computer was operated another Second Life viewer controlled by a human, to carry out local tests while the agent was running on another computer. The following describes only the most important tests, all that is not mentioned is because it worked properly and efficiently.

4.2.1 Visibility and mobility

With respect to visibility messages are sent between the agent and sensors. The arrival of these messages is asynchronous, there are two factors that influence this communication, first depend on the speed of your Internet connection and also depend

on the speed of the server to run scripts. In testing the agent had an average yield, due to the two previously mentioned factors, with a slight delay in the update of the sensors, which caused a slight displacement between the avatars (see Figure 2). Another important factor is the time it takes the agent to process the information received by the sensors because the sensors send a message with a specific format which is then processed by the agent. The sensor detects the avatars send a message with the format "X Y Z | Avatar1Name X Y Z | Avatar2Name X Y Z | … | AvatarNName X Y Z", As can be seen is a list of names of avatars and their positions, except the first element does not contain name, as is the current avatar position Israel as the character delimiter "|" between each avatar. The objects responsible for detecting sensor sends a list of objects detected in a radius of one meter from the avatar Israel. Once the agent retrieves the sensor information is passed to the respective module and this triggers a series of actions such as re-routing of the avatar to dodge obstacles. One very important thing to mention is that these objects that are an obstacle may be in motion at a certain speed or variable speed or just static, i.e. no movement. This is because in the virtual world can be built motorcycles, cars, airplanes, beads, dogs, cats, etc.., i.e. they are in motion.

Fig. 2. Avatar Israel (man shaped) controlled by the agent. The agent is trying to place near avatar Israel and avatar Darnes (woman shaped).

4.2.2 Natural Language Process (NLP)

To address each message in the virtual world a thread is created, and so, N threads are created to address the messages received by objects and avatars. The creation of these threads is dynamically and the number of active threads depends heavily on the power of the computer you are using to run the agent. Therefore, the agent can respond to multiple messages at one time, but not necessarily in the order they were received, since each thread consumes a different time depending on the information to be processed (see Figure 3). Although the agent is time consuming to connect to the Internet, the results are satisfactory. Example of this, is that the agent can answer questions like "*what color is Napoleon's white horse*" the answer given is "*is white because are questions of mental agility and trick*" as you can see the correct answer is just "*is white*", but the answer given by the agent's response was found on the Internet using Google, and this is a very good approximation to the real answer.

Despite *eSpeak* [17] [18] was used to give voice to the agent, still it has errors in pronunciation, but these can be fixed by improving *eSpeak*, as it is open source.

Fig. 3. Avatar Israel (man shaped) controlled by the agent. The agent responds an avatar Darnes question where avatar Darnes is controlled by a human.

It is important to note, the Second Life viewer [19] makes revisions on the state of the display of the agent to see if there is one person handling it and avoid it to be expelled. The agent to deceive the Second Life viewer reviews must occasionally send keystrokes to trigger events that produce animations, which are associated with a function key on the keyboard, so it tricks the viewer at the time of make revisions. For example:

1. When the agent sends the message "hello" then triggers an animation that makes the avatar move an arm in greeting.
2. When the agent will represent some number on the abacus, this triggers an animation of the agent pointing to the abacus.
3. When the agent is putting together a message activates an animation of the avatar "writing" and sent the message after the animation stops.
4. When the agent is accessing the Internet makes an animation of thinking.
5. When the agent is doing a sum and will represent the number on the abacus was made an animation of marking.
6. When the agent cannot answer a question makes a movie of frustration.

All these animations are made using different applications, such as Avimator (See Figure 4) and QAvimator (See Figure 5).

The main difference between Avimator and QAvimator is that QAvimator allows greater control over the movements of the avatar, while Avimator have less control on the movement of the avatar.

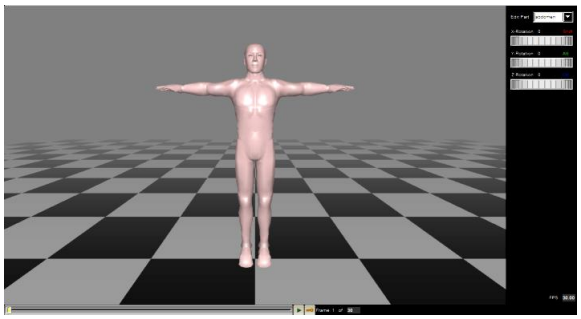


Fig. 4. Dummy avatar that shows how animation is made, moving or rotating any part of the avatar as are the arm, abdomen, legs and head.

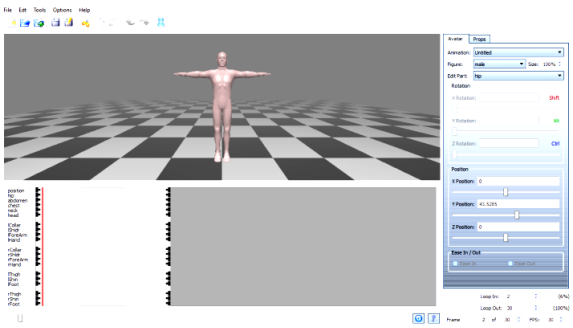


Fig. 5. Greater control over the movement of the avatar using QAvimator.

4.2.3 Using the abacus

EasyTalk [20] protocol is employed to use the abacus, which allows the agent to know the status of the abacus (see Figure 6). The drawback is the time it takes to update the state, since it depends on two factors, one is the speed Internet connection and the other is the time it takes to run the abacus' scripts. The later time depends entirely on the Second Life server and cannot be measured. However, the agent is able to update the status of the abacus, but due to this delay, not controllable by him, on occasion may misinterpret the status. The state of the abacus is updated by the agent considering multiple messages.

Fig. 6. Israel (man shaped) controlled by the agent. Agent is representing the number 5 on the abacus.

5 Conclusions and Future Work

A general model was designed so that agents to manage avatars within a virtual world can be developed. In particular we have shown the results of applying the model on an avatar driven by a management agent to teach the abacus, in the virtual world of Second Life. It is important to note that for the development of such applications is necessary to have a great amount of computational resources.

The module of natural language processing, was extended for including Internet search, in order to provide better results, but the agent's response time was affected significantly. Three different versions of the agent presented in the study case were developed, which differ substantially in how they implemented the natural language processing module.

Version 1, which is the more stable is the one described in this paper. However, the other versions have served as a comparison. Version 2 uses Gecko [21] for connections made by the agent to the Internet, when a response is requested by an avatar handled by humans. This is much more reliable but requires more computational resources. Version 3 takes DLL to connect to the database, which causes them to consume more run time and resources.

In future work is proposed to replace the current vision system which is based on sensors for 3D vision system. This improvement allows agent teleportation be enhanced since agents no longer depend on objects programmed into LSL and thus can teleport between metaverse using the protocol OGP [22].

Besides, it is proposed to improve the natural language processing, adding context analysis, this no doubt that the agent will take longer to respond, but would improve the results in the response.

Also consider that can improve response time [14] using CUDA technology from NVIDIA [23], for parallel computing, because the Second Life viewer does not use it. This module could add support for multiple languages, this can be achieved using the application *solicitudgetCLR* also the agent must connect to the Google translator, and so do the translation of the sentences received a certain language. It also aims to improve the implementation *solicitudgetCLR* (in charge of links to Google and the SAR) to make a better analysis of Web pages or the other option is to completely

replace that part of Gecko [21]. Gecko undoubtedly is the best option, but requires a greater amount of computational resources.

References

1. "Institutes in Second Life". [Online] 2010. http://secondlifegrid.net/casestudies.
2. "Definición de Metaverse". [Online] 2010. http://en.wikipedia.org/wiki/Metaverse.
3. TAC Trading Agent Competition Homepage. [Online] 2010. http://www.sics.se/tac
4. "Enseñan a pensar a un avatar de Second Life". [Online] 2010. http://www.tendencias21.net/Ensenan-a-pensar-a-un-avatar-de-Second-Life_a2293.html
5. "Bringing Second Life To Life: Researchers Create Character With Reasoning Abilities Of A Child". [Online] 2010 http://www.sciencedaily.com/releases/

A. Barnetta. "Fortune 500 companies in Second Life, Master Thesis". [Online] 2010 http://www.smi.ethz.ch/education/thesis/Barnetta_SecondLife.pdf

6. RunAlong. [Online] 2010 http://www.runalong.se/
7. Paul R. Messinger, "A Typology of Virtual Worlds: Historical Overview and Future Directions", Journal of Virtual Worlds Research, Austin, USA, pp. 1-18, 2008.
8. "Second Life" [Online] 2010 http://en.wikipedia.org/wiki/Second_Life.
9. Ranking de mundos virtuales. [Online] 2010. http://gigaom.com/2007/06/13/top-ten-most-popular-mmos/
10. Yilin Kang, "Self-Organizing Agents for Reinforcement Learning in Virtual Worlds" in Proceedings of WCCI 2010 IEEE World Congress on Computational Intelligence, Barcelona, Spain July, 18-23, pp. 3641-3648, 2010.
11. S.Kopp, L. Gesellensetter, N.C.Krmer, and I.Wachsmuth, "A Conversational agents museum guide-design and evaluation of a real-world application", IntelligentVirtualAgents, pp.329–343, 2005.
12. Israel Guzmán P. , Darnes Vilariño A., Fabiola López L., Maria J. Somodevilla, Mireya Tovar V., Beatriz Beltrán M. ¨ "ISRAEL: An agent inside second life virtual world "Digital Scientific and Technological Journal, Editorial Amate, pp 226-231, 2009.
13. Israel Guzmán, Darnes Vilariño, María J. Somodevilla," An Intelligent Agent who teaches the Basic Operations with an Abacus inside Virtual World Second Life", Research in Computing Science, Advances in Soft Computing, Vol 49, 2010, pp 165-176.
14. How to use an Abacus. [Online] 2010. http://www.educalc.net/144267.page
15. What is OpenSim?. [Online] 2010. www.opensimulator.org, 2010
16. Speech Synthesizer. [Online] 2010. http://espeak.sourceforge.net/
17. Voice over Internet Protocols. [Online] 2010. http://www.asterisk.org/, 2010
18. Herramientas para compilar en Windows el visor de Second Life. [Online] 2010.
19. http://wiki.secondlife.com/wiki/Microsoft_Windows_Builds
20. Protocol EasyTalk, [Online] 2010. http://wiki.secondlife.com/wiki/LSL_Protocol/EasyTalk
21. Motor de Navegación para Firefox. [Online] 2010. http://developer.mozilla.org/es/Gecko.
22. Protocol OGP. [Online] 2010. http://wiki.secondlife.com/wiki/OGP_Base.
23. Cálculo Paralelo de NVIDIA utilizando la tecnología CUDA. [Online] 2010. http://www.nvidia.es/object/cuda_home_new_es.html
24. Framework para desarrollo de Agentes en JAVA, [Online] 2010. http://jade.tilab.com/
25. Bringsjord S., Shilliday A., Clark M., Werner D., Taylor J., Bringsjord A., Charpentier E.. "Toward Cognitively Robust Synthetic Characters in Digital Environments". In Artificial General Intelligence Proceedings of the First AGI Conference, 2008.

Multi-robot coordination strategies for exploration

Ket-ziquel Hernández, Abraham Sánchez, Maria A. Osorio†, Alfredo Toriz P.‡ and Francisco Sosa

Computer Science Department
†Chemical Engineering Department
Benemérita Universidad Autónoma de Puebla
‡Robotics Department
LIRMM, UMR 5506 - CC477
ket2107@hotmail.com, asanchez@cs.buap.mx, alfredo.torizpalacios@lirmm.fr
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. The approach to explore an unknown environment by multiple robots is a parallelization of the SRT method, which is based on the random generation of robot configurations within the local safe area detected by the sensors. In this paper, we propose several coordination strategies to solve the cooperative exploration problem. We focus our attention in the cooperative policy strategy, which is completely decentralized as each robot decides its own motion by applying some rules only on the locally available information. Simulation results in various environments are presented to compare the performance of proposed coordination strategies.

1 Introduction

Multiple robots are increasingly used in different applications to cooperatively solve complex tasks. Many successful robotic systems use maps of the environment to perform their tasks. There are numerous studies to find efficient ways for exploring and creating maps for an unknown environment [14], [7], [15], [10], [8], [4]. Unfortunately, most of the studies deals with single robots. However, exploring an unknown environment with single robots has several disadvantages, e.g., real-time applications with single robots takes more time than using multi-robots. Besides, single robots can not produce accurate maps like multi-robots. The only advantage of using a single robot is the minimization of the repeated coverage. Even though repeated coverage among the robots decreases the mission's efficiency, some amount of repeated coverage is a desirable situation for better efficiency, this better efficiency can be achieved by coordination among robots. If multi-robots can explore an unknown area faster than a single robot, there is a very important question: How can we coordinate the behavior of robots in the unknown area?.

This paper presents a method to explore an unknown environment by multi-agent robots, the Multi-SRT approach. This method is a parallelization of the SRT (Sensor-based Random Tree) idea, which was presented in [8]. The present

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 89-99

paper builds on the previous work presented in [11], [12], [6]. The basics of the Multi-SRT method are briefly presented in Section II. The proposed multi-agent robots coordination are detailed in Section III. Simulation results in different environments are discussed in Section IV. Finally, conclusion and future work are detailed in Section V.

2 The multi-SRT method

Consider a population of n identical robots. Each robot is equipped with a ring of range finder sensor or a laser range finder, the sensory system provides the local safe region $S(q)$. The robots move in a planar workspace, i.e., $\mathbb{R}^2$ or a connected subset of it; the assumption of planar workspace is not restrictive. Each robot is a polygon[1] or another shape subject to non-holomic constraints. Each robot also knows its configuration q, one can eliminate this assumption by incorporating a localization module in the method. The robots know its ID number and each robot can broadcast within a communication range R_c the information stored in its memory (or relevant portions of it) at any time. The robot ID number is included in the heading of any transmission. The robot is always open for receiving communication from other robots inside R_c.

The exploration algorithm for each robot is shown in Figure 1. First, the procedure BUILD_SRT is executed, i.e., each robot builds its own SRT, $\mathcal{T}$ is rooted at its starting configuration q_{init}. This procedure terminates when the robot can not further expand $\mathcal{T}$. Later, the robot executes the SUPPORT_OTHERS procedure, this action contributes to the expansion of the SRTs that have been built by others robots. When this procedure finishes, the robot returns to the root of its own tree and finishes its exploration. For more details of the approach, one can consult the work [12].

BUILD Multi-SRT(q_{init})
1 $\mathcal{T}.init(q_{init})$
2 BUILD_SRT($q_{init}.\mathcal{T}$);
3 SUPPORT_OTHERS(q_{init});

Fig. 1. The Multi-SRT algorithm.

In each iteration of the BUILD_SRT, the robot uses all available information partially collected by itself and partially gained through the communication with other robots. The procedure SUPPORT_OTHERS can be divided into two major phases, which are repeated over and over again. In the first phase, the robot picks another robot to support it in his exploration, or, more precisely, another tree

[1] Polygonal models make it possible to efficiently compute geometric properties, such as areas and visibility regions.

that helps it to expand (there may be more than one robot acting on a single tree). In the second phase, the selected tree is reached and the robot tries to expand it, tying subtrees constructed by the procedure BUILD_SRT. The main cycle is repeated until the robot has received confirmation that all the other robots have completed their exploration [11]. Figure 2 shows two different views of the execution of the Multi-SRT algorithm in an environment that contains 10 nonholonomic mobile robots.

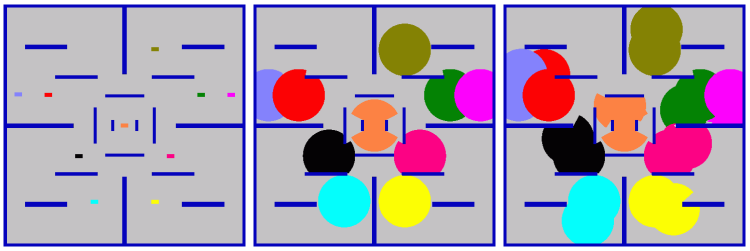


Fig. 2. Snapshots showing the execution of the Multi-SRT algorithm.

BUILD_SRT$(q_{init}, \mathcal{T})$
1 $q_{act} = q_{init}$;
2 **do**
3 BUILD_AND_WAIT_GPR();
4 $S(q_{act}) \leftarrow$ PERCEIVE(q_{act});
5 ADD$(\mathcal{T}, (q_{act}, S(q_{act})))$;
6 $\mathcal{G} \leftarrow$BUILD_GER();
7 $\mathcal{F}(q_{act}) \leftarrow$ LOCAL_FRONTIER$(q_{act}, S(q_{act}), \mathcal{T}, \bigcup \mathcal{T}_i)$;
8 $q_{target} \leftarrow$ PLANNER$(q_{act}, \mathcal{F}(q_{act}), q_{init})$;
9 **if** $q_{target} \neq NULL$
10 **if** $|\mathcal{G}| > 1$
11 $(\mathcal{G}_f, \mathcal{G}_u) \leftarrow$ CHECK_FEASIBILITY$(\mathcal{G})$;
12 **if** $\mathcal{G}_u \neq \emptyset$
13 $q_{target} \leftarrow$ COORDINATE$(\mathcal{G}_f, \mathcal{G}_u)$;
14 $q_{act} \leftarrow$ MOVE_TO(q_{target});
15 **while** $q_{target} \neq NULL$

Fig. 3. The BUILD_SRT procedure.

The local coordination procedures implemented in the proposal guarantees that the collective motion of the robots are feasible from the collision viewpoint. The approach does not need a central supervision. The selection of exploration

actions by each robot is spontaneous and it is possible on the basis of the available information.

3 Coordination strategies

Since exploration task requires cooperation and coordination among robots, the achievement of the task will be accidental if robots work independently. This task can not be done unless robots cooperate and coordinate their behaviors. Therefore, it is necessary to have cooperation strategies that allow multiple robots to help each other in the problem solving process.

To solve a multi-robot task, either centralized or decentralized (distributed) approaches can be used. A centralized model uses a powerful robot to plan and schedule the subtasks for every robot. This control robot has a global knowledge concerning the environment and the problems. It can deliberately plan for better performance. On the other hand, a distribute approach decreases design complexity and cost, while increasing the reliability. Robots are autonomous and equal. A robot plans for itself and communicates with the others in order to accomplish the global task. Since every robot interacts directly with the environment, it is reactive. However, each robot has only local knowledge of the task and the environment. Hence, it cannot make the best decision of the global task alone. Furthermore, negotiation and social cooperation rules for conflict resolution are required to coordinate among them.

In this work, several strategies were utilized to solve multi-robot exploration tasks, in the next paragraphs we introduce two new strategies, a blackboard approach and a decentralized cooperative policy for conflict resolution. The two originally proposed strategies in the work presented in [12] were also considered (i.e., coordination via arbitration and coordination through replanning [11]).

A blackboard system in general, is a distributed, opportunistic approach to system design. It is characterized by a set of knowledge sources that can communicate with each other via an area of global memory called a blackboard. Each knowledge source is designed to solve a specific component of the problem that the system is presented with. The blackboard is a global section of memory that is accessible to all of the knowledge sources. The blackboard contains the data, as well as partial solutions to the problem at hand. In a robotic system, the blackboard could be seen as a representation of the world state, through sensor input, actuator positions, world maps, and other pertinent information. The set of knowledge sources comprises the problem-solving component of the system. Each of the knowledge sources is tailored to a specific function.

Frazzoli et al. [5], proposed a novel policy for steering multiple vehicles, the policy rests on the assumption that all agents are cooperating by implementing the same rules. They mentioned that their policy is completely decentralized, as each robot decides its own motion by applying those rules only on the locally available information processed by each robot. Their policy applies to systems in which new mobile robots may enter the scene and start interacting with existing

ones at any time, while others may leave. The proposed spatially decentralized control policy is based on a number of discrete modes of operation [9].

In order to explain the rules on which this coordination strategy is based, it is necessary to define the suitable annotation to refer us to the robot position at certain time. A configuration describes the pose of the robot, i.e., it can be represented using two parameters (x, y). We call plan to a pair of positions (g_s, g_g), which are a start configuration and a goal configuration, respectively. One can define a plan as follows: plan(g_s, g_g) is a safe movement of a robot in an environment, where g_s is the current position and g_g is the candidate position. During the exploration time, each robot collects all starting points in a set G_s and their goal positions in a set G_g. Let $G_s = \{g_s, s = 1, ..., n\}$ the set of all start robots configurations at certain time and $G_g = \{g_g, g = 1, ..., n\}$ the set of all goal robots configurations at certain time, where i denotes the ID robot. We can mention the following rules of the proposed policy, which can guarantee to the system to be collision-free.

Admissibility. One can consider an environment in which new robots may issue a request to enter the scenario at an arbitrary time and with an arbitrary plan, consisting of a start and goal configurations. It is important to have conditions to efficiently decide on the acceptability of a new request. The new proposed plan is compatible with the properties $\mathbf{P_1}$ and $\mathbf{P_2}$.

> $\mathbf{P_1}$: A configuration set $G_s = \{g_s, s = 1, ..., n\}$ is unsafe for the policy ζ, if there exist a set of target $G_g = \{g_g, g = 1, ..., n\}$ such that ζ leads to a collision.
> $\mathbf{P_2}$: A target configuration set $G_g = \{g_g, g = 1, ..., n\}$ is blocking for the policy ζ, if there exist a set of configurations $G_s = \{g_s, s = 1, ..., n\}$ from which ζ leads to a dead-lock or live-lock.

A plan (G_s, G_g) is admissible if it verifies the predicate $\neg\mathbf{P_1}(G_s) \wedge \neg\mathbf{P_2}(G_g)$, i.e., there are no collisions with the robots plans in the environment.

Well-posedness. The first step of this coordination policy is to verify that each robot is a well-posed dynamical system, i.e., a solution exists and is unique, for all initial conditions within a given set. In other words, from the beginning and during the exploration process of the environment, each robot does not have to invade the safe regions of others robots. Next figure illustrates this rule. The safe region is a geometrical form created from its sensors.

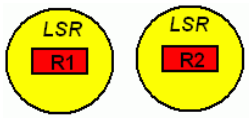


Fig. 4. The safe regions of each robot initially do not overlap between them.

This rule is defined by the next theorem, for more details, see [9].

Theorem 1 *The system is well-posed, for all initial conditions in which the interiors of local safe regions are disjoints.*

Safety. This rule of the cooperative policy proposes that the safe regions of robots are not overlapped during the exploration process; if at some given time there exists an overlapping, the robot does not have to advance, i.e., the robot searches for a new candidate position, in case of not finding it, the robot will remain in its position. The figure shows overlaps of safe regions of two robots, this problem is corrected with the safety rule.

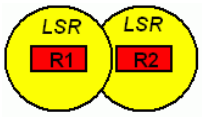


Fig. 5. The safe regions of near robots never must be overlapped.

According to the previous description, this safety rule is related with the property $\mathbf{P_1}$, from which the following theorem is derived [9].

Theorem 2 *If the safe regions (SR) of at least two robots overlap, property* $\mathbf{P_1}$ *is verified. In this case, a backtracking policy or a replanning one is implemented. Since robots always are within their local safe region, it is possible to be assured that the system is collision-free.*

Liveness. Liveness is related with the property $\mathbf{P_2}$, it is based in the definition of a condition that maintains separated the safe regions, which are associated to the robots goal configurations. In other words, the liveness of a robot is in charge of maintaining to certain distances the robots goal positions to avoid overlap of these regions and with this possible collisions. This rule is important since it allows to maintain a considerable distance between the robots goal positions, in addition, aid to avoid live-lock, see the following figure.

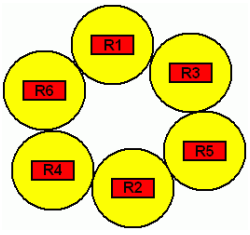


Fig. 6. Live-lock during the exploration process.

Since it is important to avoid live-lock during the exploration process, we can define the following property.

P3: A configuration set $G_g = \{g_g, g = 1, ..., n\}$ is clustered if the distance between goals configurations is smaller than the radius of the safe region.

From the previous property the following theorem is derived, the proof is detailed in [9].

Theorem 3 *The property* $\mathbf{P_2}$ *is valid for the coordination policy if the property* $\mathbf{P_3}$ *is also valid.*

In other words, the sparsity of goal configurations is a necessary condition to rule out the possibility of blocking executions in this policy, a sufficiency condition is presented in the following theorem.

Theorem 4 *Consider two mobile robots such that the center of the safe regions in final configurations are at distance larger than* $2d + 4$.

where d is the Euclidean distance between two configurations.

This policy allows the mobile robots to reach their final destinations in finite time, from all initial conditions such that the safe regions are disjoint. Following the rules previously discussed, we propose the following algorithm for the coordinated exploration with multi-agent robots in unknown environments, see figure 7. It is possible to note that the new algorithm is similar to the original multi-SRT algorithm, i.e., we only adapted it for the cooperative policy as the coordination strategy among robots.

Two procedures have been added to the algorithm: CHECK_POSEDNESS and CHECK_SAFETY_LIVENESS. The first procedure is in charge of checking that the robots safe areas (LSRs) must be disjointed and the robots poses must be to at least a distance of two times the perception radius. The second procedure is in charge of reviewing at each step that the target configurations are not within the local safe regions of other robots, this rule is carried out with a query, in which the goal configuration is searched within the safe local regions of other robots, in the case of that a target configuration exists within the areas, it is necessary to execute again a coordination with the procedure COORDINATE($\mathcal{G}_f, \mathcal{G}_u$), which only realises a search of a new target. The procedure CHECK_SAFETY_LIVENESS also verifies that a blocking does not exist, this is obtained with a condition in the target positions, which must be to less than a distance of two times the perception radius.

4 Simulation results

The tests were performed on a Celeron © 430 processor-based PC running at 1.80 Ghz with 2 GB RAM. The strategies were implemented in Visual C++, taking advantage of the MSL library's structure and its graphical interface[2]. The GPC library developed by Alan Murta was used to simulate the sensors perception systems[3]. The polygonal representation facilitates the use of the GPC

[2] http://msl.cs.uiuc.edu/msl/

[3] http://www.cs.man.ac.uk/∼toby/alan/software/

```
COOPERATIVE_POLICY(ID, T_i, q_act, S(q_act))
1  q_act = q_init;
2  CHECK_POSEDNESS(q_act)
3  loop
4   S ← PERCEPTION(q_act);
5   ADD(T, (q_act, S));
6   G ←BUILD_GER();
7   F(q_act) ← LOCAL_FRONTIER(q_act, S(q_act), T, ⋃ T_i);
8   q_target ← PLANNER(q_act, F(q_act), q_init);
9   UPDATE_INFORMATION(ID, T_i, q_act, S(q_act));
10        if |G| > 1
11          (G_f, G_u) ← CHECK_SAFETY_LIVENESS(G);
12          if G_u ≠ ∅
13           q_target ← COORDINATE(G_f, G_u);
14        q_act ← MOVE_TO(q_target);
15 while q_act = q_init ∧ F(q_act) = ∅
```

Fig. 7. The cooperative policy algorithm.

library for the perception algorithm's simulation. If S is the zone that the sensor can perceive in absence of obstacles and SR the perceived zone, the SR area is obtained using the difference operation of GPC between S and the polygons that represent the obstacles.

One can consider two possible initial deployments of the robots. In the first, the robots are initially scattered in the environment; and the second, the exploration is started with the robots grouped in a cluster. Since the Multi-SRT approach is randomized, the results were averaged over 10 simulation runs. We consider that an increment of the number of evenly deployed robots corresponds to a decrement of the individual areas they must cover. When the robots are far apart at the start, they can exchange very little information during the exploration process.

Figures 8 and 9 illustrate the Multi-SRT and explored regions with clustered and scattered starts respectively. We can see the difference when the robots are evenly distributed at the start of are clustered. At the end of the exploration process, the environment has been completely explored and the SRTs have been built. In these figures, one can observe that each robot built its own SRT and when one of them finished, this entered the support other phase.

Exploration time for teams of different cardinality are shown in Figures 10 and 11, both in the case of scattered and clustered starts. In theory, when the number of robots increases, the exploration time would quickly have to decrease.

Agent coordination and communication are important issues in designing decentralized agent systems. Various communication strategies are presented in this paper: blackboard, replanning, arbitration and cooperative policy, as

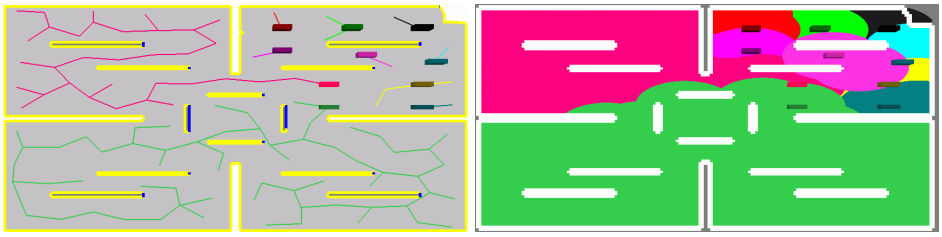


Fig. 8. The Multi-SRT and explored regions with clustered starts with a team of 10 robots.

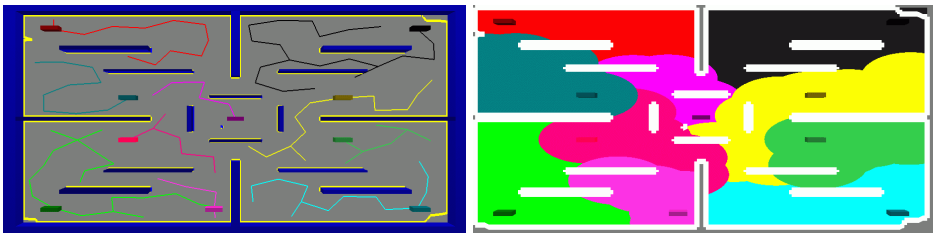


Fig. 9. The Multi-SRT and explored regions with scattered starts with a team of 10 robots.

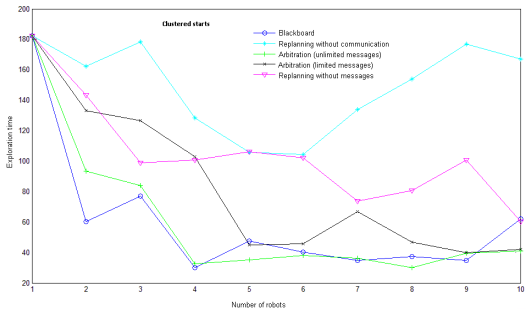


Fig. 10. Exploration time versus number of robots for the clustered starts situation.

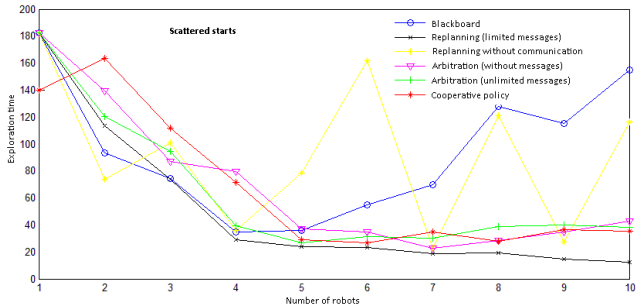


Fig. 11. Exploration time versus number of robots for the scattered starts situation.

part of the coordination for environments exploration, which, by nature, are implemented in limited and unlimited communication ranges.

The graphic generated with the exploration times obtained, show that for those environments with a scattered initial configuration, the communication strategies more efficient are the limited communication with messages and the limited communication with no messages. The disadvantage of the limited communication without messages strategy, is that it will require more re-exploration in the presence of more robots, i.e., increasing the exploration time, unlike the limited communication with messages that prevents re-exploration. In the graphic of the initial cluster configuration, it can be seen that, as more sophisticated and less centralized is the communication between the robots, as more optimal are the results corresponding to the exploration and the resolution of conflicts. The analysis of the results obtained for the case of the initial configuration in cluster shows that any communication strategy either limited or unlimited is useful for resolving conflicts related to collisions and to carry out the exploration efficiently.

The communication strategy with blackboard tends to be sluggish when the number of robots in the environment is increased, because the structure used as a board is shared and only one robot at a time can access it; if more than a robot wants to access the board at the same time, it must wait in line for a turn. The cooperative policy shows the best results, even in those environments considered difficult to explore. From a particular point of view the good performance of this policy is because, it initially performs a quick exploration of the environment. This can be verified after analyzing the implementation of our approach with all the coordination strategies, and compare their performance in the graphic obtained with the strategy of cooperative policy, in the support others phase that the robots use to complement the exploration.

Finally, it can be concluded that the integration of a communication strategy in the robots, as a way of coordination to avoid conflicts, is very useful in the task of environment exploration, because it can help them to share information in order to avoid conflicts related to collisions and, in some cases, to prevent re-exploration areas in the most important phase, when a good communication strategy can contribute in reducing the exploration time.

5 Conclusions and future work

Exploration using multiple robots is characterized by techniques that avoid tightly coordinated behavior. The implemented policy gives rise to a hybrid system, which can be shown to be well posed and safe, if the initial configurations satisfy a rather nonrestrictive condition. Through the examples, we can affirm that the policy is spatially decentralized and its complexity is bounded regardless of the number of agents.

Exploration and localization are two of capabilities necessary for mobile robots to navigate robustly in unknown environments. A robot needs to explore in order to learn the structure of the world, and a robot needs to know its

own location in order to make use of its acquired spatial information. However, a problem arises with the integration of exploration and localization simultaneously. The integration of a localization module into the exploration process based on SLAM techniques will be an interesting topic for a future research. We can also consider an extension of the Multi-SRT exploration method, where the robots constantly maintain a distributed network structure.

References

1. W. Burgard, M. Moors and F. Schneider, "Collaborative exploration of unknown environments with teams of mobile robots", *Plan-Based Control of Robotic Agents, LNCS*, Vol. 2466, (2002)
2. Y. Cao, A. Fukunaga and A. Kahng, "Cooperative mobile robotics: Antecedents and directions", *Autonomous Robots*, Vol. 4, (1997)1-23
3. G. Dudek, M. Jenkin, E. Milios and D. Wilkes, "A taxonomy for multi-agent robotics", *Autonomous Robots*, Vol. 3, (1996) 375-397
4. J. Espinoza L., A. Sánchez L. and M. Osorio L., "Exploring unknown environments with mobile robots using SRT-Radial", *IEEE Int. Conf. on Intelligent Robots and Systems*, (2007) 2089-2094
5. E. Frazzoli, L. Pallotino, V. G. Scordio and A. Bicchi, "Decentralized cooperative conflict resolution for multiple nonholonomic vehicles", *Proc. of the American Institute of Aeronautics and Astronautics Conf.*, (2005)
6. K. Hernández Guadarrama, "Estrategias de coordinación para la exploración con multi-agentes robóticos", *Master Thesis*, FCC-BUAP (in spanish), (2010)
7. J. Ko, B. Stewart, D. Fox, K. Konolige and B. Limketkai, "A practical, decision-theoretic approach to multi-robot mapping and exploration", *IEEE Int. Conf. on Intelligent Robots and systems*, (2003) 3232-3238
8. G. Oriolo, M. Vendittelli, L. Freda and G. Troso, "The SRT method: Randomized strategies for exploration", *IEEE Int. Conf. on Robotics and Automation*, (2004) 4688-4694
9. L. Pallottino, V. G. Scordio, A. Bicchi and E. Frazzoli, "Decentralized cooperative policy for conflict resolution in multi-vehicle systems", *IEEE Transactions on Robotics*, Vol. 23, No. 6, (2007) 1170-1183
10. R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun and H. Younes, "Coordination for multi-robot exploration and mapping", *17th Conf. of the American Association for Artificial Intelligence*, (2000) 852-858
11. A. Toriz Palacios, "Estrategias probabilisticas para la exploración cooperativa de robots móviles", *Master Thesis*, FCC-BUAP (in spanish), (2007)
12. A. Toriz P., A. Sánchez L. and M. A. Osorio, "Coordinated multi-robot exploration with SRT-Radial", *LNAI 5290, Springer-Verlag*, (2008) 402-411
13. A. Toriz P., A. Sánchez L. René Zapata and M. A. Osorio, "Building feature-based maps with b-splines for integrated exploration", *LNAI 6433, Springer-Verlag*, (2010) 562-571
14. B. Yamauchi, " Decentralized coordination for multirobot exploration", *Robotics and Autonomous Systems*, Vol. 29, (1999) 111-118
15. R. Zlot, A. Stenz, M. Dias and S. Thayer, "Multi-robot exploration controlled by a market economy", *IEEE Int. Conf. on Robotics and Automation*, (2002) 3016-3023

Texture Segmentation On a Local Binary Pattern Space

Gemma S. Parra-Dominguez, Raul E. Sanchez-Yanez, and
Victor Ayala-Ramirez

Universidad de Guanajuato DICIS Carretera Salamanca-Valle de Santiago Km 3.5+1.8 Km Comunidad de Palo Blanco, C.P. 36885 Salamanca, Gto.
gsparra@laviria.org, sanchezy@salamanca.ugto.mx, ayalav@salamanca.ugto.mx
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. A visual texture segmentation approach is proposed. Here, an input image is transformed into a local binary pattern space, making it a local indicator of texture units. Those texture units are then used to calculate local binary pattern difference histograms and local homogeneity values. Such values, computed from texture units, are a better indicator of texture presence than homogeneity values computed directly from gray levels. The local histograms improve the creation of classes and allow the possibility to discriminate between different textures. Then, our system performs both tasks, texture detection and texture discrimination trough a refinement of the detected textured regions. Experimental work shows good texture segmentation over Brodazt textures, along with good identification of non-textured regions. Texture segmentation is also performed on natural scenes with satisfactory results.

Key words: Texture detection, Texture segmentation, Local binary patterns, Difference histograms, Local homogeneity values.

1 Introduction

Visual texture is an important property of most objects, since it helps us to differentiate between elements in an image. Usually, texture is considered as an indicative of the gray level spatial distribution; and it is possible to describe it as the repetition of a certain pattern (texture element or texel) over a large region, larger than the pattern itself [1].

Texture analysis plays an important role in low level image analysis and understanding [2]. Usually, texture detection represents a processing prior to image classification or segmentation. Many texture analysis methods are based on gray level co-occurrence matrices, difference histograms, lineal transformations, Gabor filters, and Markov random fields [1]. Even though, most methods in texture analysis are orientated towards texture classification or segmentation, Karu *et al.* [3] argue that before applying any texture analysis to an image, it is necessary to identify if the input image has some textural characteristics, and if so, to locate where the texture is.

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 103-113

A texture characterization approach is the spatial gray level co-occurrence matrices (GLCM) introduced by Haralick [4]. Texture features such as energy, entropy and homogeneity can be computed from GLCM [4]. Each texture feature represents a specific property or phenomenon. In particular, homogeneity refers to how similar distributions of gray levels are. It is a second order statistic, with values between 0 and 1, where 0 stands for heterogeneous distributions, and 1 is for identical ones. Unser [5] proposed the use of first order statistics, computing sum and difference histograms, to approximate those features. Then, computation time is reduced and memory storage reminds manageable [5].

Different from GLCM approaches, there are non parametric methods such as the local binary patterns (LBP). A number of works about LBP is found nowadays. The LBP operator introduced by Ojala *et al.* [6] provides a robust way for describing pure local binary patterns in a texture. It is relatively invariant with respect to changes in illumination and image rotation, and it is computationally simple [7]. The use of LBP histograms as texture feature descriptors, has been extensively explored to achieve texture segmentation and texture classification [2, 7, 8, 9, 10]. However, none of those works have used an image in the LBP domain to compute textural features.

In this paper, an approach for texture segmentation is presented using an image transformation into an LBP space. Two textural features are then composed from it, local homogeneity values and LBP difference histograms. These allow the detection of texture presence and the discrimination between different types of texture. Although the most relevant activity is the transformation into an LBP space, four similar issues are repeated in the two stages of the system (detection and refinement): an splitting of the input image, a computation of feature vectors, a clustering process and a conformation of the output image.

The rest of the paper is organized as follows. In Section II, an introduction to the LBP space transformation is given. Also a brief review about difference histograms and homogeneity values computation is included. Section III describes the method explaining every step and assumed considerations. In Section IV, a performance evaluation is shown, also results and a brief discussion are presented. Finally, the conclusions of this study are given in Section V.

2 Image Transformation and Textural Features

The LBP transformation is a very simple process, just consisting in sums and comparisons. Having an input image I_{in} of size $K \times L$, it is scanned by a window of 3×3 pixels, and pixels inside the window are thresholded by the center pixel value. Values bigger or equals to the threshold are marked as 1; otherwise, they are marked as 0. As result, a binary neighborhood is obtained. Later, this binary neighborhood is multiplied by the weights given to the corresponding pixels. Finally, the values of the eight pixels are summed up to obtain a texture unit value associated to this neighborhood, as shown in Fig. 1 [6]. The transformed image will be of $(K-2) \times (L-2)$ size.

LBP texture units characterize a region, depending on its frequency over a determined zone. A distribution of texture units shows which of them are present and the number of times they appear in the region under inspection. Moreover, a distribution of texture unit differences, shows a relationship between them. When those texture units have similar values, the distribution of differences will tend to 0. Then, this distribution of differences, also called difference histogram, can characterize a region.

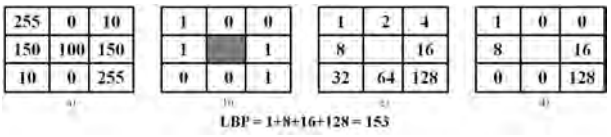


Fig. 1. An example of how LBP operator is applied to a pixel neighborhood[6].

Difference histograms and homogeneity values are calculated according to Unser's formulation, as explained in [5]. Then, having a normalized difference histogram $\hat{P}$ the local homogeneity in specific neighborhoods is computed using

$$h = \sum_{j} \frac{1}{1 + j2} \hat{P}(j) \tag{1}$$

where j is the difference value between two pixels according to a particular displacement and orientation [5].

As a feature, texture may have preferential directions over the image, and the inspection formulated should include most of them. Then, two displacement vectors $d_x = \{1,2\}$ and $d_y = \{1,2\}$ pixels, together with an angle vector $\Theta = \{0,45,90,135\}$ are used, seeking to detect texture in different directions. The average value of the 8 calculations in the Cartesian product $d \times \Theta$ is used in this work to calculate $\hat{P}$ and h.

3 Texture Segmentation Approach

The texture segmentation approach includes two stages, a detection of textured regions and a refinement of detected regions. In the first one, an image indicating the presence of texture is obtained; while in the second one, an image indicating the presence of different types of texture is reached. The main difference between both stages, is that more rigorous similarity measures between regions are used for refinement.

3.1 Detection of Textured Regions

A texture detection process is performed in order to determine if the image under test has some textured regions or not. If texture is identified on the input

image, a second process is then performed to achieve texture discrimination. The detection process performs four activities: an splitting of the input image, a computation of feature vectors, a clustering procedure, and finally, a conformation of the output image.

Splitting of the Input Image The texture detection system receives as input data an image I_{in} of $K \times L$ size in gray levels. This I_{in} is split into smaller images I_{a} using a partition window of size $M \times K$, where $M < K$ and $N < L$. This partition window is displaced according to a particular distance in each direction, horizontal and vertical. Then, a number of I_{a} is obtained from I_{in} depending on the partition window size and its displacement distance. This displacement distance determines the resolution of the system.

Computation of the Feature Vectors In general, the feature vectors are composed of a value of homogeneity, an LBP differences histogram and a key code of the region position. They are computed as follows. Every I_{a} is transformed into an LBP space, where a transformed I_{Ti} is obtained. Later, an examination over I_{Ti} is done in order to determine if I_{Ti} is an uniform region. I_{Ti} is divided in four different images I_{Tj} of the equal size. For every I_{Tj} an histogram W_{j} is computed. Then, the four W_{j} are compared using the cosine amplitude metric

$$r_{a,b} = \frac{\sum_{k=1}^{m} W_{ak} W_{bk}}{\sqrt{(\sum_{k=1}^{m} W_{ak}^2)(\sum_{k=1}^{m} W_{bk}^2)}} \tag{2}$$

where $r_{\text{a,b}}$ is the similarity value between two different histograms, W_{ak} and W_{bk} are the histograms with m values. The cosine metric amplitude is used to compare two histograms and to determine, in a range from 0 to 1, how similar they are. A value of 1 means identical histograms, while a value of 0 means dissimilar histograms. The smallest $r_{\text{a,b}}$ is divided by the biggest $r_{\text{a,b}}$ of the four W_{j}. From this division, a value bigger than 0.8 is expected; if it is reached, the four I_{Tj} are joined as I_{Ti}, if not, they are handled separately as I_{Tj}.

Unser's formulation is applied to obtain the normalized LBP difference histogram $\hat{P}$ and the local homogeneity value h of the region. $\hat{P}$ is an LBP difference histogram because it is computed from a transformed image. Then, there are two types of feature vectors, X_{i} computed from I_{Ti} and X_{j} computed from I_{Tj}

$$X_i = [Position_i, h_i, \hat{P}_i]$$
$$X_j = [Position_j, h_j, \hat{P}_j]$$

where, sub-indexes i and j stand for crops at different scale.

Clustering Procedure Once the feature vectors are computed, there are three possible cases to be performed by the clustering procedure. In case A only X_{i} is found, in case B both X_{i} and X_{j} exist, and in case C only X_{j} is available. Two revisions are made during clustering of the feature vectors. The first one groups

all vectors into different classes, while the second one, ensures the vectors are grouped into the best possible class.

The first revision begins for cases A and B using X_{i}, and for case C using X_{j}. The homogeneity value h_1 of the first vector is compared with the other homogeneity values h_n of the set. The difference between them is taken according to

$$d(n) = h_1 - h_n; (n = 1, ..., N) \tag{3}$$

and where the smallest $d(n)$ is found, both $\hat{P}$ are compared using (2). If a certain similarity percentage is reached, both vectors are grouped into one class.

In the next iteration, a different vector is inspected using the same method:

1. Comparison of homogeneity values: the vector under test is compared using (3) against the rest of vectors that have not been grouped, and against to all the groups previously created.
2. Similarity revision of histograms: the vector under test is compared using (2) against the vector, and against the group where the smallest $d(n)$ is found.
3. Merging of feature vectors: if the desired similarity value is reached, the inspected vector is grouped where the similarity is maximum. If it is not reached for any group or any vector, a new cluster containing only the inspected vector is created.

Before creating a new class, a similarity test is made using (2) and the average $\hat{P}$ of the clusters. If this new class is similar in certain percentage to an existing one, then both classes are grouped, if not, a new cluster is made. The goal is to avoid having repeated clusters. Each time a new vector is merged into a certain group, the average values of the group are updated. The process is repeated until all vectors are grouped into one and only one cluster.

The second revision occurs after all the vectors have been grouped. Here, the goal is to make sure that every vector is classified into the best possible cluster. This is achieved by comparing every $\hat{P}$ of each feature vector with the average $\hat{P}$ of each cluster, using (2). Then, every vector is stored in the group where the similarity is maximum. For case B, is here where X_{j} is assigned to one class. Since the average values of the clusters are updated during this revision, another similarity inspection is performed in order to avoid having repeated classes. Using the average $\hat{P}$ of the clusters and (2), the groups are considered as belonging to the same class if they are similar up to a certain percentage; higher values indicate it is necessary to merge the groups. When they are not similar enough, both groups remain.

During the clustering process, a threshold for homogeneity values is set. If any h is equal or bigger than the threshold, the region with this vector is considered as non-texture and the vector is not considered for a cluster creation. It is stored in an extra class. Also, during this process, a matrix containing which vector belongs to each class is created, using the original position key codes.

Conformation of the Output Image To create the output image I_{D} of the detection stage, the average h of the groups are sorted in descendent order;

leaving the most textured groups first, and the more homogeneous at the end. Then, every vector gets as gray level value x the number of its cluster, where $x = 1, 2, \ldots, X$. I_{D} will have the most textured region in black color, the textured regions in gray intensities, and the more homogeneous in lighted color. The regions considered as non-texture are painted in white.

3.2 Refinement of the Detected Regions

During the refinement, every textured region is inspected individually, following the methodology described before. Thus, if N classes were created, N inspections are performed. Each inspection includes an splitting of the input image, a computation of feature vectors and a clustering procedure. At the end, if more classes are created, a general revision is performed. Information about the final clusters is used to conform the output image. The cluster containing the non-texture regions is excluded from this inspection.

1. Splitting of I_{in} using I_{D} as reference: a number of images I_{b} is obtained, according to the size of the textured region under inspection, by using the same partition window and the same displacement distance, as before. The direction of this displacement is now determined by I_{D}.
2. Computation of the features vectors: after I_{b} is transformed into an LBP space image I_{Tk}, an uniformity examination over this region is not necessary. Then, feature vectors X_{k} are composed as described in Section 3.1, and, every X_{k} has the same scale.
3. Clustering procedure: as explained in Section 3.1, the clustering procedure is performed using X_{k}. The similarity percentages are more restrictive than those in the detection stage, seeking to discriminate among different types of texture.

In order to avoid having repeated classes, a general revision is performed after the N inspections. This is made by comparing every $\hat{P}_{\mathrm{k}}$ with the average $\hat{P}$ of every cluster. Then, X_{k} is stored in the group whose similarity is maximum. Again, the cluster containing the non-textured regions is excluded in this revision. A matrix containing which region belongs to each class is created, using the original position key codes of X_{k}.

For the creation of the output image I_{R} of the refinement stage, the same procedure described in Section 3.1 is followed. Later, to create the output image I_{out} of the system, two majority filters of size 3×3 pixels are applied to I_{R}, in order to eliminate any spurious regions created in the process. A majority filter assigns the most repeated values in the neighborhood to the pixel under inspection.

4 Experimental Work and Discussion

Before evaluation, four parameters were determined for the detection process: 1) Similarity between vectors of 0.850 (85%), 2) Similarity between clusters of 0.80

(80%), 3) Similarity between final clusters of 0.950 (95%), and 4) Homogeneity threshold of 0.875. For the refinement stage, a similarity between vectors of 0.950 (95%) and a similarity between clusters of 0.930 (93%) were used. Other considerations were assumed: partition windows of size 32×32 pixels to collect data with a displacement distance of 16 pixels, cosine amplitude as a resemblance metric between two histograms, and Unser's estimation to calculate local homogeneity values and local difference histograms.

A texture mosaic containing textured and non-textured regions was composed in order to evaluate the detection of homogeneous zones in an image. From Brodazt photographic album [11], texture D68 and texture D16 were selected. In Fig. 2 we can see the original image, its ground truth reference, and the results for texture detection, and for texture refinement before and after filtering. As we can see, the result is improved on each step of the approach, as expected. The homogeneous region is well detected, and both textured regions are segmented. However, texture 4 and texture 5 are mistakenly created. The errors in the frontiers between textures are due to the mix of information from different textured zones. The region can be assigned to the most present texture in the crop, or, it can be identified as another type of texture. In Table 1, a confusion matrix of segmentation results upon Texture Mosaic 1 is shown. Here, values are given in percentage values. A 91.70% of correct assignation was reached when comparing pixel correspondence between the output image and its ground truth reference.

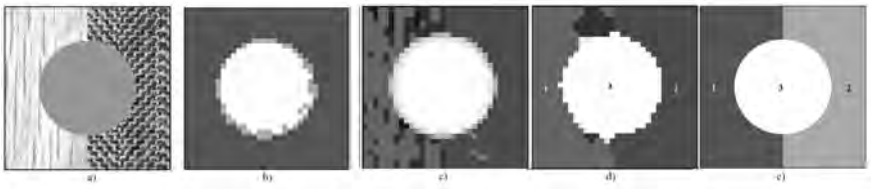


Fig. 2. Texture mosaic 1: a) I_{in}, b) I_{D}, c) I_{R}, d) I_{out}, and e) ground truth reference.

Table 1. Confusion matrix of texture mosaic 1

Class	1	2	3	TOTAL
1	29.20	0.0	0.0	29.20
2	0.29	35.89	0.34	36.52
3	2.38	0.69	26.62	29.69
4	3.91	0.0	0.0	3.91
5	0.68	0.0	0.0	0.68
TOTAL	36.46	36.58	26.97	100

Two more texture mosaics [12] were evaluated using the texture segmentation system. In Fig. 3, a texture mosaic consisting of two different types of texture is shown, together with its ground truth reference and the results of each step. Two textured regions were obtained as expected, and a 94.5% of correct assignation when comparing pixels correspondence was reached, see Table 2. In Fig. 4 a texture mosaic consisting of five different textures is shown. All textured regions were segmented, however, texture 3 is being confused with texture 2. Even tough a 69.30% of global performance was reached, more than 91.0% of texture 3 and texture 4 was correctly assigned when comparing pixels correspondence, see Table 3. If a different resolution is used, a displacement distance of 8 pixels, an improved result is obtained (refer to Fig. 5). Here, well defined elements are shown; a global performance of 66.45% was reached, and texture 4 is 99.20% correct assigned when comparing pixels correspondence.

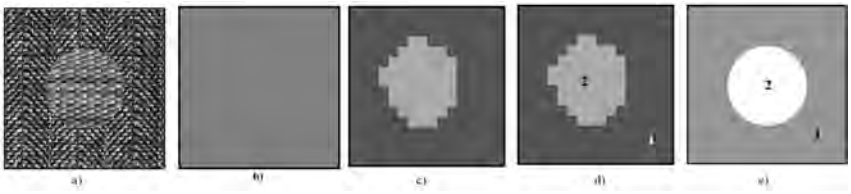


Fig. 3. Texture mosaic 2: a) I_{in}, b) I_{D}, c) I_{R}, d) I_{out}, and e) ground truth reference.

Table 2. Confusion matrix of texture mosaic 2

Class	1	2	TOTAL
1	77.29	2.40	79.69
2	3.10	17.21	20.31
TOTAL	80.39	19.61	100

A last evaluation was performed over a natural scene, using the image 108073 from the Berkeley database [13]. In Fig. 6 the original image and the evaluation results are shown. Three textured regions were detected (see Fig. 6b) and seven textured regions were segmented.

5 Conclusions

A texture segmentation approach is proposed. The methodology is divided in two tasks: a texture detection and a refinement of detected textured regions. Texture segmentation is efficiently performed using an input image transformed

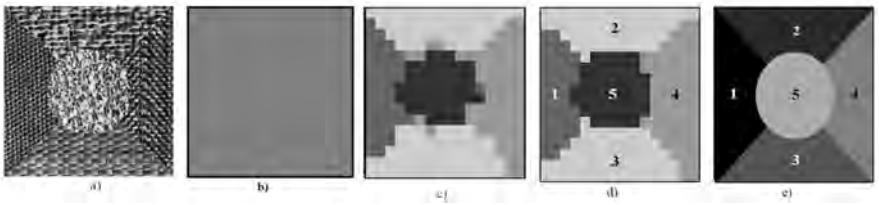


Fig. 4. Texture mosaic 3: a) I_{in}, b) I_{D}, c) I_{R}, d) I_{out}, and e) ground truth reference.

Table 3. Confusion matrix of texture mosaic 3

Class	1	2	3	4	5	TOTAL
1	15.63	0.0	0.0	0.0	0.0	15.63
2	3.19	18.69	18.46	0.71	1.53	42.58
3	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	1.05	1.31	19.19	2.28	23.83
5	1.48	0.69	0.01	0.0	15.80	17.97
TOTAL	20.29	20.42	19.78	19.90	19.61	100

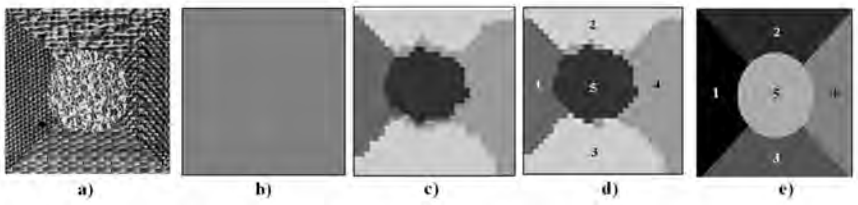


Fig. 5. Texture mosaic 2:a) I_{in}, b) I_{D}, c) I_{R}, d) I_{out}, and e) ground truth reference, using a displacement distance of 8 pixels.

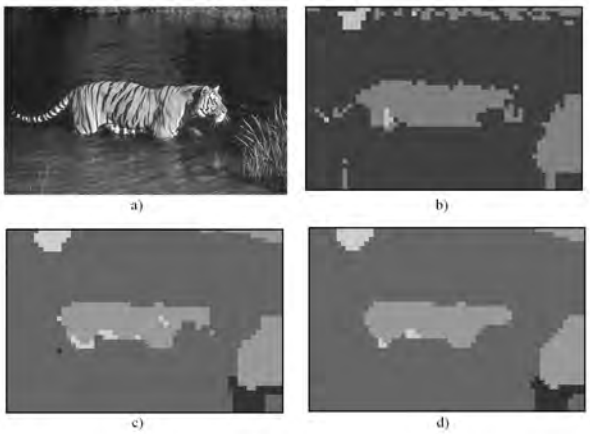


Fig. 6. Image 108073 [13]: a) I_{in}, b) I_{D}, c) I_{P}, and d) I_{out}.

into an LBP space; having then local indicators of texture units. Then, local homogeneity values indicate the presence of textured and non-textured regions; while LBP difference histograms improve the creation of classes and allow the discrimination between different textures. Texture segmentation is a refinement process after texture detection.

Experimental work shows good texture segmentation over Brodatz textures, along with good identification of non-textured regions. Texture segmentation is also performed on natural scenes with satisfactory results. Texture segmentation can be improved by decreasing the displacement distance of the partition window.

Even though the creation of prototypes and the clustering procedure are unsupervised processes, some parameters and considerations have to be determined empirically before texture segmentation. Automatic determination of them would be desirable in future works for performance improving and generalization of the system. Texture classification can be achieved by designing a learning process accordingly.

Acknowledgements

Gemma S. Parra-Dominguez gratefully acknowledges the Mexican National Council for Science and Technology (CONACyT) for the financial support through the scholarship grant number 302076/290564.

References

[1] M. Tuceryan and A. K. Jain, "Texture analysis," in *The Handbook of Pattern Recognition and Computer Vision*, L. F. P. C. H. Chen and P. S. P. Wang, Eds.

World Scientific Publishing, 1993, ch. 11, pp. 235–276.

[2] X. Qing, Y. Jie, and D. Siyi, "Texture segmentation using LBP embedded region competition," *Electronic Letters on Computer Vision and Image Analysis*, pp. 41–47, 2005.

[3] K. Karu, A. K. Jain, and R. M. Bolle, "Is there any texture in the image?" *Pattern Recognition*, vol. 29, no. 9, pp. 1437 – 1446, 1996.

[4] R. M. Haralick, K. Shanmugam, and I. Dinstein, "Textural features for image classification," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 3, no. 6, pp. 610 –621, nov. 1973.

[5] M. Unser, "Sum and difference histograms for texture classification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. PAMI-8, no. 1, pp. 118 –125, jan. 1986.

[6] T. Ojala, M. Pietikinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51 – 59, 1996.

[7] T. Maenp and M. Pietikinen, "Texture analysis with local binary patterns," in *The Handbook of Pattern Recognition and Computer Vision*, 3rd ed., C. H. Chen and P. S. P. Wang, Eds. World Scientific Publishing, 2005, ch. 1, pp. 197–216.

[8] X. Liu and D. Wang, "Image and texture segmentation using local spectral histograms," *Image Processing, IEEE Transactions on*, vol. 15, no. 10, pp. 3066 –3077, oct. 2006.

[9] M. Savelonas, D. Iakovidis, and D. Maroulis, "An lbp-based active contour algorithm for unsupervised texture segmentation," *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, vol. 2, pp. 279 –282, 2006.

[10] E. Tekeli, M. Cetin, and A. Ercil, "A local binary patterns and shape priors based texture segmentation method," *Signal Processing and Communications Applications, 2007. SIU 2007. IEEE 15th*, pp. 1 –4, jun. 2007.

[11] U. of Southern California. USC-SIPI image database. Http://sipi.usc.edu/database/database.cgi?volume=textures.

[12] R. Trigve. Trigve Randen. Http://www.ux.uis.no/ tranden/.

[13] P. Arbelaez, C. Fowlkes, and D. Martin. The Berkeley segmentation dataset and benchmark. Http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/.

On Geodesic Distance Computation: An Experimental Study

David Bautista-Villavicencio and Raúl Cruz-Barbosa

Universidad Tecnológica de la Mixteca, 69000, Huajuapan, Oaxaca, México
{dbautista,rcruz}@mixteco.utm.mx
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. The most common distance function used in the machine learning field is the Euclidean distance. It is due to its easily intuitive understanding and interpretation in the real world. However, it has been verified that for datasets representation presenting convoluted geometric properties (many foldings), the Euclidean distance is not the proper choice to measure the (dis)similarity between two elements of the data manifold. An alternative distance function that alleviates, in part, the previously mentioned problem is the geodesic distance, since it measures similarity along the embedded manifold, instead of doing it through the embedding space. Some problems of computing geodesic distances are related to computational time and storage restrictions about the graph representation and the shortest path algorithm to be used. Thus, the main objective of this paper is to show some characteristics about computational time and storage performance for computing the geodesic distance by using the basic Dijkstra algorithm and a full data matrix representation against some alternatives in order that researchers can select the suitable one depending on the available computational resources.

1 Introduction

Automation of the human being learning process is a complex task. When dividing the existing reality into different categories, we are seamlessly performing a classification task that can be improved over time through learning.

In the machine learning field, the task of unraveling the relationship between the observed data and their corresponding class labels can be seen as the modeling of the mapping between a set of data inputs and a set of discrete data targets. This is understood as supervised learning.

Unfortunately, in many real applications class labels are either completely or partially unavailable. The first case scenario is that of unsupervised learning, where the most common task to be performed is that of data clustering. The second case, semi-supervised learning, is less frequently considered but far more common than what one might expect: quite often, only a reduced number of class labels is readily available and even that can be difficult and/or expensive to obtain.

The distance functions have an important role in the machine learning field, mainly in unsupervised and semi-supervised learning. For clustering tasks, the

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 115-124

used distance can help to discover the involved groups by measuring how close are the data points to the groups prototypes. In the case of dimensionality reduction, specially manifold learning methods use a distance for finding the shortest path between two data points in the data manifold. For semi-supervised classification tasks, a distance function is used, for instance, for sharing or propagating the class label of labeled elements in the dataset to the closer unlabeled elements.

The most commonly used distance function in machine learning is the Euclidean distance. This has widely been used due to its easily intuitive understanding and interpretation in the real world. Besides, the simplicity of its computation makes it very suitable when we are in the process of selecting a distance function. However, it has been verified that for datasets representation presenting convoluted geometric properties (many foldings), the Euclidean distance is not the proper choice to measure the (dis)similarity between two elements of the data manifold [1–3]. In these cases, the Euclidean distance can be an inexact measure of the proximity between data. This problem becomes more complicated when we work with sample data that resides in a high dimensionality space and we have not additional information about their intrinsic geometry. This situation is the main motivation of this paper, since many datasets involving the previously mentioned properties are presented in real world applications as biomedicine, bioinformatics or web mining. Thus, the previous scenario encourages researchers for modeling this kind of data with an alternative distance function.

An alternative distance function that alleviates, in part, the previously mentioned problem is the geodesic distance, since it measures similarity along the embedded manifold, instead of doing it through the embedding space. Unlike Euclidean distance, geodesic distance follows the geometry of the manifold where data reside. In this way, it may help to avoid some of the distortions (such as breaches of topology preservation) that the use of a standard metric such as the Euclidean distance may introduce when learning the manifold, due to its excessive folding (that is, undesired manifold curvature effects).

Machine learning methods using geodesic distances can be categorized, according to their main task, as unsupervised and semi-supervised learning methods. Within the unsupervised learning methods are found the pioneering works of [1] y [4], where the main task to perform is non-linear dimensionality reduction. Other methods used for this task are [3] and [5]. Also, it can be found some variants of these as in [2]. For clustering and visualization tasks, the geodesic distance has been used in [6] and [7]. On the other hand, the first semi-supervised methods used for classification task were reported in [8] and [9]. These methods, as well as many others that involve the geodesic distance, are known as graph-based methods. Some methods of this type but from different nature are, for example, those based on Support Vector Machines [10], Self-Organizing Maps [11] and Generative models [12].

Most of the graph-based methods, previously mentioned, compute the data point pairwise distance of a graph using the basic Dijkstra algorithm and a full data matrix representation for finding the shortest path between them. The main

problems of this solution involve computational restrictions related to computational time and storage. Thus, the objective of this paper is to show some characteristics about computational time and storage performance for computing the geodesic distance by using the basic Dijkstra algorithm and a full data matrix representation against some alternatives in order that researchers can select the suitable one depending on the available computational resources.

The rest of the paper is organized as follows. In section II, the geodesic distance procedure and some alternatives of the involved modules in it are presented. The experimental results using several UCI datasets are shown in section III. Finally, the conclusions and future work are outlined in section IV.

2 Geodesic distances

At the present time, in some fields of computer science, such as machine learning, the use of alternative metrics like geodesic distance are common and widely used [1, 4, 5]. These methods use the geodesic distance as a basis for generating the data manifold. This metric favours similarity along the manifold, which may help to avoid some of the distortions that the use of a standard metric such as the Euclidean distance may introduce when learning the manifold. In doing so, it can avoid the breaches of topology preservation that may occur due to excessive folding.

The otherwise computationally intractable geodesic metric can be approximated by graph distances [13], so that instead of finding the minimum arc-length between two data points lying on a manifold, we would set to find the shortest path between them, where such path is built by connecting the closest successive data points. In this paper, this is done using the K-rule, which allows connecting the K-nearest neighbours (another alternative is the ϵ-rule, which allows connecting data points $\mathbf{x}$ and $\mathbf{y}$ whenever $\|\mathbf{x} - \mathbf{y}\| < \epsilon$, for some $\epsilon > 0$). A weighted graph is then constructed by using the data and the set of allowed connections. The data are the vertices, the allowed connections are the edges, and the edge labels are the Euclidean distances between the corresponding vertices. If the resulting graph is disconnected, some edges are added using a minimum spanning tree procedure in order to connect it. Finally, the distance matrix of the weighted undirected graph is obtained by repeatedly applying Dijkstra's algorithm [14], which computes the shortest path between all data samples. For illustration, this process is shown in Fig. 1.

2.1 Alternatives modules for graph distance computation

From Fig. 1, it can be shown that there are different alternatives for some of the involved modules in the geodesic distance computation. Some crucial characteristics, about computational time and storage constraints, for computing this distance are the graph representation of the dataset and the shortest path algorithm to be used. On the one hand, two alternatives for graph representation are: adjacency matrix and adjacency list. The former consists in a n by n matrix

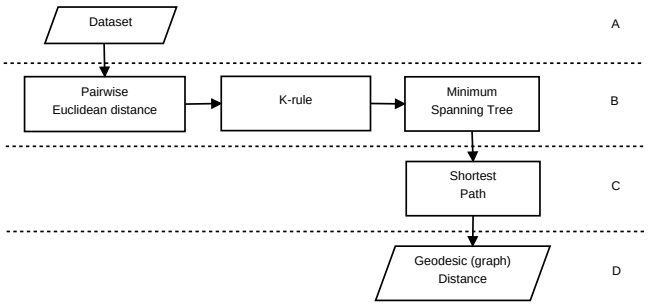


Fig. 1. Graph distance procedure scheme. Stage (A) represents the input data. Stage (B) is for building the weighted, undirected, connected graph. Stage (C) is for computing the geodesic (graph) distance, which is returned in stage (D).

structure, where n is the number of vertices in the graph. If there is an edge from a vertex i to a vertex j, then the element a_{ij} is 1, otherwise it is 0. This kind of structure provide faster access for some applications but can consume huge amounts of memory. The latter considers that each vertex has a list of which vertices it is adjacent to. This structure is often preferred for sparse graphs as it has smaller memory requirements.

On the other hand, three options (of several) for the shortest path algorithm are: (basic) Dijkstra, Dijkstra using a Fibonacci heap and Floyd-Warshall. All of them assumes the graph is a weighted, connected graph. The Dijkstra's algorithm is as follows.

Require: A source node x and a weighted, connected graph $G = (V, E)$, where V is a finite set of vertices (nodes), and E is a collection of edges that connect pairs of vertices.

Ensure: The shortest path between a source node x and every other node in V.

1. $V_{new} = x$, where $x \in V$ is a source node
2. $E_{new} = \emptyset$

repeat

(3a) Choose edge (u, v) from E with minimal weight such that $u \in V_{new}$ and $v \in V \setminus V_{new}$

(3b) Update path's length and add v to V_{new} and (u, v) to E_{new}.

until $V_{new} = V$

It its widely known that the time complexity of the simplest implementation, using the Big-O notation, for this algorithm is $O(|V|^2)$.

For some applications where the obtained graph is a sparse graph, Dijkstra's algorithm can save memory resources by storing the graph in the form of adjacency list and using a Fibonacci heap (F-heap) as a priority queue to implement extracting minimum efficiently. In this way, it can improve the running time of the algorithm 2.1 to $O(|E| + |V|log|V|)$, where the main improvement is made in the step (3a).

A Fibonacci heap is a binary tree which has the property that for every subtree, the root is the minimum item. This data structure is widely used as priority queue [15]. The priority queues are used to keep a dynamic list of different priorities jobs. A F-heap allows several operations as, for instance, ***Insert()*** adds a new job to the queue and ***ExtractMin()*** extracts the highest priority task.

Another approach for computing the shortest path is given by the Floyd-Warshall algorithm, which is an example of dynamic programming. Here, it finds the lengths of the shortest paths between all pairs of vertices. This algorithm is stated as follows.

Require: A weighted, connected graph $G = (V, E)$, where V is a finite set of n vertices (nodes), and E is a collection of edges that connect pairs of vertices.
Ensure: The shortest paths among the nodes in V.
Initialize the $path$ matrix, where $path[i][j] = weight(i, j)$, where $weight(i, j)$ returns the weight of the edge from i to j
for $k = 1...n$ **do**
  for $i = 1...n$ **do**
    for $j = 1...n$ **do**
      $path[i][j] = \min(path[i][j],$
        $path[i][k] + path[k][j])$
    end for
  end for
end for

Unlike Dijkstra's algorithm which assumes all weights are positive, this algorithm can deal with positive or negative edge weights. The complexity for this algorithm is $O(|V|^3)$.

3 Experiments

The goal of the experiments is twofold. Firstly, we aim to experimentally assess which combination of graph representations and shortest path algorithms produce the best computational time and storage performance for computing the geodesic distance of datasets with increasing number of items. Secondly, we aim to evaluate and compare the performance of the best combination (for graph representation and shortest path) found in the previous experiment using the C++ language (gcc compiler) and the Matlab®(R2009a) software. Since Matlab is widely used in the machine learning field and the computer science community, we select it for the comparison.

The process for computing geodesic distances is shown in Fig. 1 and explained in section 2. The experiments are carried out setting the K parameter equal to 10, in order to get a connected graph after the K-rule is applied. After that, the K parameter is set to 1 for showing the time performance of geodesic distance computation with an unconnected and sparse graph. The experiments are performed on a dual-processor 2.3 Ghz BE-2400 desk PC with 2.7Gb RAM.

3.1 Results and discussion

Five datasets, taken from the UCI machine learning repository [16], of increasing number of items were used for the experiments that follow. The first one is *Ecoli*, consisting on 336 7-dimensional points belonging to 8 classes representing protein location sites. The second dataset, German-credit-data (numerical version) herein called *German*, consists of 1000 24-dimensional data points belonging to good or bad credit risks. The third dataset is called *Segmentation*, which is formed by 2310 19-dimensional items representing several measurements of image characteristics belonging to seven different classes. The fourth dataset, *Pageblocks*, involves block measurements of distinct documents corresponding to five classes. It consists of 5473 items described by 10 attributes. The fifth set, herein called *Pendigits*, consists of 10992 16-dimensional items corresponding to (x, y) tablet coordinate information measurements, which belong to ten digits (classes).

The time performance results for computing geodesic (graph) distances, using $K = 10$, are shown in Table 1. Here, a combination of Adjacency matrix for graph representation and basic Dijkstra for shortest path algorithm outperformed the other combinations, except for *Pageblocks*, whereas the memory is not exceeded. This is due to the faster access to elements in an adjacency matrix when basic Dijkstra's algorithm required them. It is interesting to notice how the time performance for the adjacency list representation and Dijkstra is better for large datasets. Using Dijkstra, the time proportion between the adjacency matrix and list is decreased when the number of items is increased. It means that the graph storage by means of an adjacency list is crucial for large datasets. This effect is pronounced for the large *Pendigits* set, where the matrix representation can not deal with it due to operating system (it dedicates approximately 700 Mb for each process) memory restrictions. For large datasets, as *Pendigits*, it can be observed the best combination is for adjacency list and Dijkstra using Fibonacci heaps. Moreover, using the list representation, if time results are compared for Dijkstra and Dijkstra using F-heaps the time proportion is decreased when number of items is increased and this difference becomes better for Dijkstra implemented with F-heaps. This tendency is similar for the matrix representation. Thus, it can be inferred that for large and very large datasets the best time performance for computing geodesic distances should be using an adjacency list (or matrix, when storage restrictions are discarded) representation and Dijkstra using F-heaps. The opposite occurs for Floyd-Warshall algorithm independently from the graph representation. Its performance is noticeable only for small sets.

Now, the K parameter for the K-rule is set to 1, in order to show the computational time and storage performance when the procedure is dealing with an unconnected and sparse graph. The corresponding results are shown in Table 2. In general, it is clear how the modified minimum spanning tree procedure to connect the graph influences the time results. However, the results tendency observed from Table 1 are kept. In addition, it can be inferred from Tables 1 and

Table 1. Computational time performance results for graph distances computation (assuming a connected graph by setting $K = 10$) using several UCI datasets and different settings. '–' symbol means exceeded memory.

Dataset (# items)	**Shortest path**	**Representation**	**Time (s)**
Ecoli (336)	Dijkstra	Adjacency Matrix	0.43
	Dijkstra+F-heaps	Adjacency Matrix	1.19
	Floyd-Warshall	Adjacency Matrix	0.53
	Dijkstra	Adjacency List	0.67
	Dijkstra+F-heaps	Adjacency List	1.59
	Floyd-Warshall	Adjacency List	0.42
German (1000)	Dijkstra	Adjacency Matrix	12.43
	Dijkstra+F-heaps	Adjacency Matrix	25.03
	Floyd-Warshall	Adjacency Matrix	23.67
	Dijkstra	Adjacency List	16.18
	Dijkstra+F-heaps	Adjacency List	38.39
	Floyd-Warshall	Adjacency List	18.71
Segmentation (2310)	Dijkstra	Adjacency Matrix	185.57
	Dijkstra+F-heaps	Adjacency Matrix	297.31
	Floyd-Warshall	Adjacency Matrix	347.16
	Dijkstra	Adjacency List	229.83
	Dijkstra+F-heaps	Adjacency List	511.59
	Floyd-Warshall	Adjacency List	292.89
Pageblocks (5473)	Dijkstra	Adjacency Matrix	3621.90
	Dijkstra+F-heaps	Adjacency Matrix	4031.93
	Floyd-Warshall	Adjacency Matrix	18369.84
	Dijkstra	Adjacency List	3585.92
	Dijkstra+F-heaps	Adjacency List	8039.92
	Floyd-Warshall	Adjacency List	10409.90
Pendigits (10992)	Dijkstra	Adjacency Matrix	– –
	Dijkstra+F-heaps	Adjacency Matrix	– –
	Floyd-Warshall	Adjacency Matrix	– –
	Dijkstra	Adjacency List	124363.18
	Dijkstra+F-heaps	Adjacency List	66105.34
	Floyd-Warshall	Adjacency List	204604.99

2 that the larger the dataset, the less affected the Dijkstra+F-heaps connection algorithm is.

Finally, the time performance of our previous implementations, in C++, for computing geodesic distances using a matrix representation and basic Dijkstra algorithm is compared with Matlab software using the same settings. The K parameter is set to 10. The results are shown in Table 3. Overall, the time performance results for the C++ implementation are better than for Matlab. In fact, any result from Tables 1 and 2 is better than the respectively obtained by Matlab. Besides, the time performance is exponentially increased when the number of items are increased using the Matlab implementation. Also, it is noticeable that Matlab can not deal with medium size sets as *Pageblocks*. Since an adjacency matrix representation is used, the memory restriction tendency is the same as in Tables 1 and 2 but is more restrictive for Matlab.

4 Conclusion

In this paper, a procedure for geodesic distance computation was carried out. Different alternatives for graph representation and shortest path algorithm, in-

Table 2. Computational time performance results for graph distances computation (assuming an unconnected, sparse graph by setting $K = 1$) using several UCI datasets and different settings. '–' symbol means exceeded memory.

Dataset (# items)	**Shortest path**	**Representation**	**Time (s)**
Ecoli (336)	Dijkstra	Adjacency Matrix	0.47
	Dijkstra+F-heaps	Adjacency Matrix	1.21
	Floyd-Warshall	Adjacency Matrix	0.6
	Dijkstra	Adjacency List	0.67
	Dijkstra+F-heaps	Adjacency List	1.57
	Floyd-Warshall	Adjacency List	0.44
German (1000)	Dijkstra	Adjacency Matrix	12.85
	Dijkstra+F-heaps	Adjacency Matrix	25.72
	Floyd-Warshall	Adjacency Matrix	23.32
	Dijkstra	Adjacency List	16.18
	Dijkstra+F-heaps	Adjacency List	37.89
	Floyd-Warshall	Adjacency List	19.27
Segmentation (2310)	Dijkstra	Adjacency Matrix	186.55
	Dijkstra+F-heaps	Adjacency Matrix	294.22
	Floyd-Warshall	Adjacency Matrix	345.38
	Dijkstra	Adjacency List	228.47
	Dijkstra+F-heaps	Adjacency List	507.53
	Floyd-Warshall	Adjacency List	192.38
Pageblocks (5473)	Dijkstra	Adjacency Matrix	3483.08
	Dijkstra+F-heaps	Adjacency Matrix	3955.05
	Floyd-Warshall	Adjacency Matrix	10867.04
	Dijkstra	Adjacency List	5549.91
	Dijkstra+F-heaps	Adjacency List	7678.91
	Floyd-Warshall	Adjacency List	10179.90
Pendigits (10992)	Dijkstra	Adjacency Matrix	––
	Dijkstra+F-heaps	Adjacency Matrix	––
	Floyd-Warshall	Adjacency Matrix	––
	Dijkstra	Adjacency List	131085.17
	Dijkstra+F-heaps	Adjacency List	67312.69
	Floyd-Warshall	Adjacency List	193720.78

Table 3. C++ vs. Matlab time performance results for graph distances computation using Dijkstra's algorithm and an adjacency matrix

Dataset (# items)	**Language**	**Time (s)**
Ecoli (336)	C++	0.43
	Matlab	8.60
German (1000)	C++	12.43
	Matlab	220.98
Segmentation (2310)	C++	185.57
	Matlab	2479.50
Pageblocks (5473)	C++	3621.90
	Matlab	––
Pendigits (10992)	C++	––
	Matlab	––

volved in this procedure, were assessed using different UCI datasets with increasing number of items. Experimental results have shown that the use an adjacency matrix for storing the corresponding graph and Dijkstra's algorithm is recommendable for computing geodesic distances of small and medium datasets. When the number of items are increased forward larger datasets the use of an adjacency list for graph representation/storage becomes crucial in this computation. Furthermore, it is shown that for large sets the use of list representation and Dijkstra using Fibonacci heaps produce better time performance than any other of the analyzed graph representation and shortest path algorithms.

Also, a comparison of a C++ and Matlab implementation for geodesic distances computation was evaluated. The experimental results have shown that the C++ implementation for this procedure is much faster than the corresponding one in Matlab. Moreover, the computational storage (memory) constraint is more restrictive for Matlab than for the C++ implementation.

As future work, it is envisaged the insertion of the obtained geodesic distance computation module into manifold learning methods for dimensionality reduction.

Acknowledgments. Authors gratefully acknowledge funding from the Mexican SEP (PROMEP program) research project PROMEP/103.5/10/5058.

References

1. Tenenbaum, J.B., de Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science **290** (2000) 2319–2323
2. de Silva, V., Tenenbaum, J.: Global versus local methods in nonlinear dimensionality reduction. In Becker, S., Thrun, S., Obermayer, K., eds.: Advances in Neural Information Processing Systems. Volume 15., The MIT Press (2003)
3. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation **15**(6) (2003) 1373–1396
4. Roweis, S.T., Lawrence, K.S.: Nonlinear dimensionality reduction by locally linear embedding. Science (290) (2000) 2323–2326
5. Lee, J.A., Lendasse, A., Verleysen, M.: Curvilinear distance analysis versus isomap. In: Proceedings of European Symposium on Artificial Neural Networks (ESANN). (2002) 185–192
6. Archambeau, C., Verleysen, M.: Manifold constrained finite gaussian mixtures. In Cabestany, J., Prieto, A., Sandoval, D.F., eds.: Proceedings of IWANN. Volume LNCS 3512., Springer-Verlag (2005) 820–828
7. Cruz-Barbosa, R., Vellido, A.: Geodesic Generative Topographic Mapping. In Geffner, H., Prada, R., Alexandre, I., David, N., eds.: Proceedings of the 11th Ibero-American Conference on Artificial Intelligence (IBERAMIA 2008). Volume 5290 of LNAI., Springer (2008) 113–122
8. Zhu, X., Ghahramani, Z.: Learning from labeled and unlabeled data with label propagation. Technical report, CMU-CALD-02-107, Carnegie Mellon University (2002)
9. Belkin, M., Niyogi, P.: Using manifold structure for partially labelled classification. In: Advances in Neural Information Processing Systems (NIPS). Volume 15., MIT Press (2003)

10. Wu, Z., Li, C.H., Zhu, J., Huang, J.: A semi-supervised SVM for manifold learning. In: Proceedings of the 18th International Conference on Pattern Recognition, IEEE Computer Society (2006)
11. Herrmann, L., Ultsch, A.: Label propagation for semi-supervised learning in self-organizing maps. In: Proceedings of the 6th WSOM 2007. (2007)
12. Cruz-Barbosa, R., Vellido, A.: Semi-supervised geodesic generative topographic mapping. Pattern Recognition Letters **31**(3) (2010) 202–209
13. Bernstein, M., de Silva, V., Langford, J.C., Tenenbaum, J.B.: Graph approximations to geodesics on embedded manifolds. Technical report, Stanford University, CA, U.S.A. (2000)
14. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische Mathematik **1** (1959) 269–271
15. Fredman, M.L., Tarjan, R.E.: Fibonacci heaps and their uses in improved network optimization algorithms. J. ACM **34**(3) (1987) 596–615
16. Asuncion, A., Newman, D.: UCI machine learning repository [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. University of California, Irvine, School of Information and Computer Sciences (2007)

On Leukocytes Classification: a Comparative Study

Verónica Rodríguez-López and Raúl Cruz-Barbosa

Computer Science Institute
Universidad Tecnológica de la Mixteca
69000, Huajuapan, Oaxaca, México
{veromix,rcruz}@mixteco.utm.mx

(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. Leukocytes classification is a complex task, where the main problems are due to morphological diversity between cells of the same type and similar features found in different types of cells. In this paper a comparative study, in terms of classification accuracy, of Bayesian networks and neural networks for leukocyte classification is presented. The design of two Bayesian network models based on the expert's knowledge and data, a naive Bayes model and a multilayer perceptron neural network are presented. The experimental results have shown that a simple naive Bayes model is a suitable classifier for this task.

Keywords: Bayesian networks; neural networks; classification; leukocytes.

1 Introduction

White blood cells, or leukocytes, are cells of the immune system involved in defending the body against infection. There are five types of leukocytes that normally appear in blood: neutrophils, basophils, eosinophils, lymphocytes, and monocytes [7].

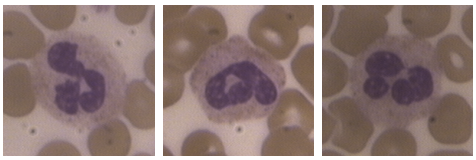
One of the most common requested test in a hematology laboratory is a complete blood count (CBC). As part of the CBC, a white blood cell count and a differential white blood cell count are done. The former measures the total number of white blood cells in a volume of blood given. The latter consists of a blood examination to determine the presence and the number of different types of white blood cells. The total number and the proportion of each type of leukocytes are associated with a person's health status [6, 3].

Leukocytes can be counted by either manual or automated hematology analyzers. The manual leukocytes count is a time consuming task, and highly dependent on lab technician skills who performs the differential analysis. Human classification errors are the main source of misclassification in the manual counts, where the main problem is the scarcity of cell samples (usually, sample sizes range from 100 to 200). On the other hand, automated hematology analyzers classify cell populations using both electrical and optical techniques. These machines decrease the time of performing routine examinations and at the same

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 125-135

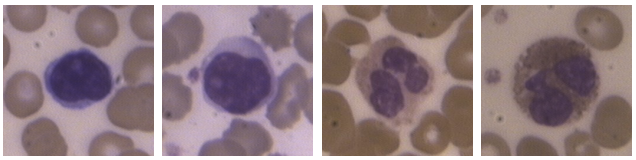
time increase cells classification accuracy. However, these analyzers are unable to accurately identify and classify all types of cells and are, particularly, insensitive to abnormal or immature cells. For this reason, most tests performed by these equipments will require a review of a skilled lab technician for definitive cell type identification [7].

To help lab technicians on leukocytes identification, many computational systems based on digital image processing and pattern recognition techniques have been developed. Despite several systems have reported a good performance [5, 9, 14], automation of leukocytes recognition is not an easy task. There are two main problems in this process. Firstly, morphological diversity is found between cells of the same type (e.g. neutrophil morphology). Secondly, similar features as shape and texture are found in different types of cell. These problems are illustrated in Figs. 1 and 2, respectively.



(a) (b) (c)

Fig. 1. Morphological diversity of neutrophil cell is presented in (a), (b) and (c).



(a) (b) (c) (d)

Fig. 2. Similar features between different types of leukocytes: (a) and (b) cells correspond to a lymphocyte and monocyte cell, respectively; (c) and (d) correspond to a neutrophil and eosinophil cell, respectively.

In this work, we compare the performance, in terms of classification accuracy, of Bayesian networks and neural networks for discrimination of five types of leukocytes. On the one hand, Bayesian networks have demonstrated to be useful as both a classifier and a powerful tool for knowledge representation and inference under conditions of uncertainty [8]. On the other hand, neural networks are a promising alternative to various conventional classification methods. These networks are data driven self-adaptive methods and they can adjust themselves

to data without any explicit specification of functional or distributional form for the underlying model [16].

This paper is organized as follows. In Sect. 2, a brief description about Bayesian networks and neural networks is presented. The description of the Bayesian network models and neural network model design for leukocytes classification and the corresponding results are shown in Sect. 3. Finally, the conclusions are presented in Sect. 4.

2 Background

2.1 Bayesian networks

Bayesian networks (BN), also known as belief networks, belong to the probabilistic graphical models family. These graphical structures are used for knowledge representation of uncertain domains and when they work with statistical techniques together present several advantages for data analysis [8].

A formal definition of a BN is as follows. A Bayesian network model, or simply a Bayesian network, is a pair (D, P), where D is a directed acyclic graph (DAG), $P = \{p(x_1|\pi_1), ..., p(x_n|\pi_n)\}$ is a set of n conditional probability distributions, one for each variable, and Π_i is the set of parents of node X_i in D [4]. The set P defines the associated joint probability distribution as

$$p(x_1, x_2, ..., x_n) = \prod_{i=1}^{n} p(x_i|\pi_i) \tag{1}$$

The simplest form of a Bayesian network is the naive Bayes model, in which the root node of a tree-like structure corresponds to a class variable. Also, this node is the only one parent for each attribute. The key assumption of the naive Bayes model is that all attributes are independent given the value of the class variable. Using this assumption, the conditional probability distribution for the class variable is very easy to calculate.

The naive Bayes assumption is helpful when we face high dimensionality input spaces. It is also useful when input vectors contain both discrete and continuous variables, since each one can be represented separately using appropriate models (e.g., Bernoulli distributions for binary observations or Gaussians for real-valued variables) [2].

Naive Bayes has been used as a simple and effective classifier in the Pattern Recognition field. It has two advantages over many other classifiers. Firstly, it is easy to construct, as the structure is given a priori. Secondly, a very efficient classification process is obtained.

2.2 Neural networks

Artificial neural networks, or simply neural networks, theory is an attempt at modeling the information processing capabilities of nervous systems. In mathematical terms, a neural network model is defined as a directed graph with the following properties:

1. A state variable n_i is associated with each node (neuron) i.
2. A real-valued weight w_{ik} is associated with each link (ik) between two nodes i and k.
3. A real-valued bias θ_i is associated with each node i.
4. A transfer function $f_i(n_k, w_{ik}, \theta_i, (k \neq i))$ is defined for each node i, which determines the state of the node as a function of its bias, weights (of its incoming links) and states of the nodes connected to it.

The transfer function usually takes the form $f(\sum_k w_{ik} n_k - \theta_i)$, where $f(\cdot)$ is a discontinuos step function or its smoothly increasing generalization known as sigmoidal function. Nodes without links toward them are called input neurons; output neurons are those with no link leading away from them [11].

The multilayer feedforward neural network, or equivalently referred to as multilayer perceptrons (MLP), is a very popular model in neural networks. A MLP has a layered structure: an input layer consisting of sensory nodes, one or more hidden layers of computational nodes, and an output layer that calculates the outputs of the network. The most common algorithm for training a MLP is named Backpropagation. In this algorithm the information is only propagated in the forward direction and there are no feedback loops. Even it does not have feed back connections, errors are back propagated during training. That is, the computations are passed forward from the input to the output layer, then the calculated errors are propagated back in the opposite direction to update the weights in order to obtain a better performance of the model [15].

3 Experiments

3.1 Experimental design and settings

The main objectives of the experiments are twofold. Firstly, we aim to explore the use of Bayesian network models for classifying all types (neutrophils, basophils, eosinophils, lymphocytes, and monocytes) of leukocytes. Secondly, a performance comparative study of Bayesian network models and neural network models is proposed.

Our experiments were carried out as follows. Initially, two tree structure Bayesian networks (TBN) models, which consider the expert's knowledge and medical literature, were designed. Then, we built a naive Bayes model and a MLP neural network model using the same features identified for the proposed Bayesian networks.

For the first experiment, the TBN-A and TBN-B models were designed. For simplicity, although perhaps there are other more suitables topologies, we used a tree structure in these models. For both models, we proposed a leukocyte classification node as the main one.

In the TBN-A model, we aimed to use some characteristics that experts take into account for the classification process. In accordance with the expert's knowledge these important characteristics are shape, size, and texture of nucleus as well as size and texture of cytoplasm. These characteristics are incorporated

into the model as discrete latent variables (or discrete latent nodes) and are connected with the (classification) principal node. Furthermore, for the Bayesian network structure building we placed some observable nodes (which are linked to the latent variables) representing the description or measurements of the corresponding features (see Fig. 3). These measurements are obtained by application of digital image processing techniques. The observable nodes are continuous variables that have a normal distribution. The description of the incorporated knowledge into the TBN-A model is presented as follows.

The first characteristic considered into the TBN-A model is the shape of the nucleus. The nucleus shape of lymphocytes is round, and the monocytes shape have a great reniform or horseshoe-shaped nucleus. The nucleus of neutrophils have from 2 to 5 lobules, it can present S, C, or glass shapes. The nucleus of eosinophils have 2 lobules and usually it is glass shaped. The nucleus of basophils is bi- or tri-lobed, but it is hard to see because of the number of granules which hide it [3, 7, 6]. This knowledge about the shape of nucleus is encoded into the nucleus shape node. The estimation of this shape is obtained by means of region descriptors, particularly, the compactness, dispersion, and the first Hu moment [12] were used. These descriptors were included into the TBN-A model as compactness, dispersion, and MH1 nodes.

Since nucleus size is more relevant than cytoplasm size for leukocytes identification, only the nucleus size is considered for the TBN-A model. Then, the nucleus size is measured as the ratio of number of pixels that belong to the corresponding region to the total number of pixels of the cell (nucleus and cytoplasm pixels). This nucleus size information is included into the nucleus size node, which was linked with the nucleus shape node due to the relationship between these two features.

The cytoplasm texture is an important characteristic of leukocytes, since it allows to group the cells by the presence or absence of granules in their cytoplasm [7]. The granulocyte type cells are neutrophils, basophils, and eosinophils. The agranulocyte cells are lymphocytes and monocytes. In order to get information about the cytoplasm texture, the energy descriptor [12] is used. This knowledge about the cytoplasm texture and its corresponding descriptor are captured with the cytoplasm texture and energyC nodes.

The texture of nucleus is another important characteristic of leukocytes that is reported in medical literature [7, 6]. For this reason, we included this knowledge into the TBN-A model in a similar way as the cytoplasm texture is.

The colour of cytoplasm is the last feature of leukocytes taken into account for the TBN-A model. The granulocyte leukocytes are characterized by the presence of differently staining colour granules in their cytoplasm: neutrophils have pink colour granules, eosinophils have orange granules, and basophils have dark purple granules. For the agranulocyte cases, the cytoplasm colour for lymphocytes is light blue and for monocytes is greyish blue [3, 6]. The colour descriptor is obtained through the average intensity value using the RGB space. The knowledge about the colour is encoded into the cytoplasm colour, Rvalue, Gvalue, and

Bvalue nodes. In summary, the topology of the TBN-A model is shown in Fig. 3.

In the second part of the first experiment, we explored the possibility to find a tree type Bayesian network model with a minimum set of nodes, which performs leukocyte classification with an acceptable degree of accuracy. A definition of the new model was found by modifying the TBN-A model. The modification is as follows. Analyzing the TBN-A model, we observed that the cytoplasm colour node is a redundant node because it does not encode uncertain information. For this reason, the cytoplasm colour node is removed. Since either cytoplasm or nucleus texture is described by one measurement we decided to remove the cytoplasm and nucleus texture nodes in the TBN-A model. We hypothesize that the energy's nodes are enough to consider the texture information. Following the previous observations, we defined the second Bayesian network model, namely TBN-B. The topology of the TBN-B model is presented in Fig. 4.

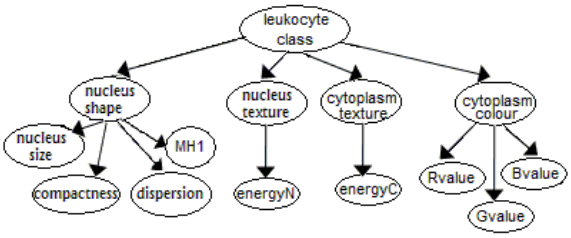


Fig. 3. Topology of the TBN-A model for leukocytes classification.

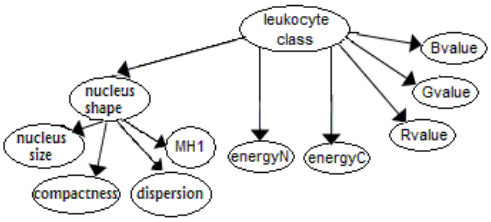


Fig. 4. Topology of the TBN-B model for leukocytes classification.

For the second experiment, the naive Bayes and the neural network models are built. To build both models, we used the most important features previously identified in the tree type Bayesian network models designed for leukocytes classification. These features are image descriptors (which are encoded in the observable or leaf nodes) used in the TBN-A and the TBN-B models.

Although a naive Bayes model does not consider domain knowledge and real dependence relationships between features, is a simpler model than other types of BN and has a good performance in classification problems. Analyzing the TBN-A and TBN-B models, we observed that there are no dependence relationships between the observable nodes, thus we hypothesize that a naive Bayes model could perform as well as a proposed BN model for leukocytes classification. Our naives Bayes model, namely NB, is showed in Fig. 5.

For the case of the MLP model, we built a neural network with a 9-N-5 topology, i.e. 9 input nodes, a hidden layer of N (where N will be determined in a range from 15 to 100 units) neurons, and a final output layer with 5 neurons providing the predicted leukocyte class. The input variables used in the first layer are the same input variables as in the NB model and the output layer corresponds with the five types of leukocytes (see Fig. 6). For training the constructed MLP, two faster algorithms than gradient descent are used: an heuristic-based and a numerical optimization-based method, namely, Resilient Backpropagation [13] and Scaled Conjugated Gradient [10], respectively.

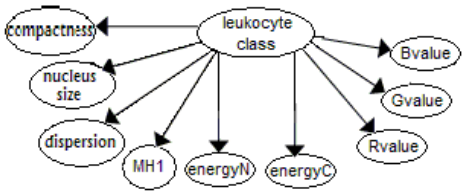


Fig. 5. Topology of the NB model for leukocytes classification.

3.2 Experimental results and discussion

In order to compare the performance, in terms of classification accuracy, of the Bayesian network and the neural network models, we used a set of 190 leukocytes colour images with a resolution of 256×256 pixels. The images were obtained by using a microscope that has an in-built CCD camera with a resolution of 640×480 pixels. The manual selection and cut of leukocytes region were applied to all images. For the nucleus and cytoplasm segmentation, we used a free software developed by Zoltan Kato [1]. The image set is formed by 8 basophils, 72 neutrophils, 9 eosinophils, 31 monocytes, and 70 lymphocytes. All images were previously classified by a human expert.

The classification performance of the designed Bayesian networks was evaluated by five-fold cross-validation. The parameters of the corresponding models

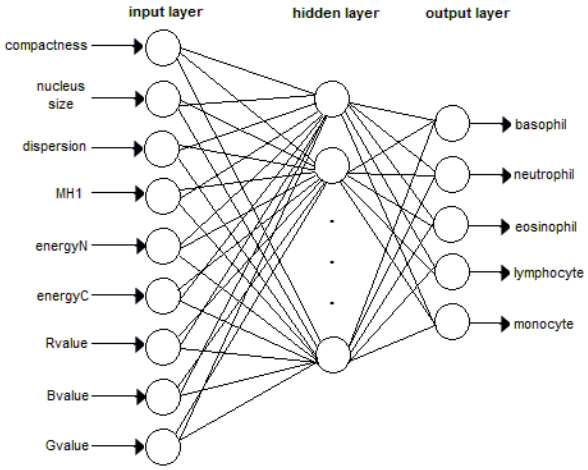


Fig. 6. Topology of the MLP model for leukocytes classification.

were obtained by using maximum likelihood estimation from complete data [8]. These models were tested using the Hugin Lite 7.3®software. The MLP model was trained and tested using the Matlab®software. As mentioned in the previous section, Resilient Backpropagation (RBP) and Scaled Conjugated Gradient (SCGBP) algorithms were used for training these neural network models. Here, the image set was divided into three sets: 60% for training set, 20% for validation, and 20% for independent test set.

The classification accuracy results for the proposed Bayesian network models and the MLP model, for the test set, are shown in Table 1. The results are better for the MLP (using 20 and 15 neurons in the hidden layer for RBP and SCGBP, respectively) and the NB models, in this case the models without domain knowledge. The domain knowledge was considered into the models with lower perfomance (the TBN-A and the TBN-B models). It seems that the domain knowledge implies dependencies between variables at same level, which were not considered into the proposed tree Bayesian network models. On the other hand, the results from Table 1 compare favourably with those classifiers that consider less types of leukocytes than the ones used here. For example, the classifiers presented in [9] and [5] consider only the most common types of leukocytes. In [9], they only classify four types (neutrophils, eosinophils, lymphocytes, and monocytes) of leukocytes achieving 86% of classification accuracy. Besides in [5], they classify neutrophils, eosinophils, and lymphocytes with 84% of accuracy.

The classification accuracy results for each type of leukocyte are presented in Table 2. Here, it can be observed that both Bayesian and neural network models can deal with the classification of all types of leukocytes, including basophils

and eosinophils, which are, usually, imbalanced classes (they appear less often in blood cells). Also, the TBN-B network has achieved better results than the TBN-A model as we expected. Although the classification results for NB and MLP variants are the same (as shown in Table 1), it can be observed, from Table 2, these results are better balanced (for each class) in NB than in MLP variants.

In general, the experimental results have shown that the simplest Bayesian network, NB model, is more suitable for leukocytes classification. Even though the results for NB and MLP models are similar, the NB model is simpler, easily implementable and faster than a neural network classifier. Furthermore, the construction of neural networks is a complex task because there are no principled methods for network parameters selection. In contrast, a naive Bayes model has an easy and fast construction procedure without parameters tuning process.

On the other hand, despite the tree Bayesian network models do not seem to be ideal for leukocytes classification, their design helped us to select the feature set used in NB and MLP models. Moreover, it should be emphasized that these models are simple classifiers that can be easily understood and verified by experts.

Table 1. Classification accuracy results for TBN-A, TBN-B, NB, MLP-RBP, and MLP-SCGBP models.

Classifier model	**classif. acc.**
TBN-A	89.5%
TBN-B	90.5%
NB	94.7%
MLP-RBP	94.7%
MLP-SCGBP	94.7%

4 Conclusions

In this paper, two tree Bayesian network models, a naive Bayes model and a multilayer perceptron neural network model (trained with two different algorithms) were tested for leukocytes classification. Despite the analyzed data set has no enough images of some types of leukocytes (imbalanced classes), all proposed classifiers have achieved a good performance, which are comparable with those reported in literature. Our proposed models, particularly, naive Bayes, can classify all types of leukocytes, including the less frequent types, with a high degree of accuracy.

The experimental results have shown that the naive Bayes and the MLP models outperformed the tree Bayesian network models. Although this performance is similar for NB and MLP models, the results suggest that simple naive Bayes model should be preferred over the complex MLP model for leukocytes classification.

Table 2. Classification accuracy results for each type of leukocyte of the TBN-A, TBN-B, NB, MLP-RBP, and MLP-SCGBP models.

Classifier model	type of leukocyte	classif. acc.
TBN-A	basophils	93.3%
	neutrophils	95.3%
	eosinophils	83.3%
	monocytes	61.0%
	lymphocytes	88.0%
TBN-B	basophils	93.3%
	neutrophils	95.3%
	eosinophils	83.3%
	monocytes	82.7%
	lymphocytes	89.5%
NB	basophils	100%
	neutrophils	95.3%
	eosinophils	83.3%
	monocytes	95.5%
	lymphocytes	95.8%
MLP-RBP	basophils	50%
	neutrophils	100%
	eosinophils	100%
	monocytes	100%
	lymphocytes	100%
MLP-SCGBP	basophils	100%
	neutrophils	100%
	eosinophils	100%
	monocytes	66.7%
	lymphocytes	92.3%

As future work, the construction of other types of Bayesian network models such as tree- and Bayesian network-augmented naive-Bayes (TAN and BAN) for leukocyte classification is considered. These models could then be compared with our proposed NB model and other kind of classifier (as support vector machines).

References

1. Berthod, M., Kato, Z., Yu, S., Zerubia, J.: Bayesian image classification using markov random fields. Image and Vision Computing (14), 285–295 (1996)
2. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
3. Carr, J.H., Rodak, B.F.: Clinical Hematology Atlas. Saunders, 2nd. edn. (2004)
4. Castillo, E., Gutierrez, J.M., Hadi, A.S.: Experts systems and Probabilistic Networks Models. Springer-Verlag (1997)
5. Colunga, M.C., Siordia, O.S., Maybank, S.J.: Leukocyte recognition using EM-algorithm. In: Aguirre, A.H., Borja, R.M., García, C.A.R. (eds.) MICAI '09: Proceedings of the 8th Mexican International Conference on Artificial Intelligence. pp. 545–555. Springer-Verlag (2009)

6. Estridge, B.H., Reynolds, A.P., Walters, N.J.: Basic Medical Laboratory Techniques. Delmar Cengage Learning, 4th. edn. (1999)
7. Greer, J.P., Foerster, J., Rodgers, G.M., Paraskevas, F., Glader, B., Arber, D.A., Means, R.T.: Wintrobe's Clinical Hematology, vol. 1. Lippincott Williams & Wilkins, 12th. edn. (2009)
8. Heckerman, D.: A tutorial on learning with bayesian networks. Tech. rep., Microsoft Research (1996)
9. Mircic, S., Jorgovanovic, N.: Automatic classification of leukocytes. Journal of Automatic Control 16(1), 29–32 (2006)
10. Moller, M.F.: A scaled conjugate gradient algorithm for fast supervised learning. Neural Networks 6(4), 525–533 (1993)
11. Muller, B., Reinhardt, J., Strickland, M.T.: Neural networks: An Introduction. Springer, 2nd. edn. (1996)
12. Nixon, M.S., Aguado, A.S.: Feature Extraction & Image Processing. Academic Press, 2nd. edn. (2007)
13. Riedmiller, M., Braun, H.: A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: Proceedings of the IEEE International Conference on Neural Networks. pp. 586–591 (1993)
14. Rodrigues, P., Ferreira, M., Monteiro, J.: Segmentation and classification of leukocytes using neural networks: A generalization direction. In: Bhanu Prasad, S.M.P. (ed.) Speech, Audio, Image and Biomedical Signal Processing using Neural Networks, pp. 373–396. Springer Berlin / Heidelberg (2008)
15. Tanga, H., Tan, K., Zhang, Y.: Neural networks: computational models and applications. Springer (2007)
16. Zhang, G.P.: Neural networks for classification: A survey. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and. Reviews 30(4), 451–462 (2000)

A Generalised Semantics for Belief Updates —An Equivalence-based Approach

Jorge Hernández and Juan C. Acosta

Computer Systems
Autonomous University of Hidalgo, ESH
UAEH-ESH, Mexico
jhcjorge@gmail.com; jguadarrama@gmail.com
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. As suggested in the literature, revising and updating beliefs and knowledge bases is an important yet unsolved topic in knowledge representation and reasoning in Answer Set Programming (ASP) that requires a solid theoretical basis, particularly in current applications of Artificial Intelligence where an agent can work in an open dynamic environment with incomplete information. Various researchers have combined postulates and ASP as key components to set up their approaches. However, many of such proposals still present some shortcomings when dealing with persistence situations, redundant information, contradictions or they simply lack of further analysis of properties that should make them more accessible. In need to satisfy more general principles and a common frame of reference, this paper introduces a general framework for updates of logic programs, a properties characterisation and its equivalence with other variants. Rather than a sequence of updates of programs, this semantics consists in performing updates of epistemic states at the object level that meets well-accepted belief revision postulates and that follows the original AGM conception.

1 Introduction

One of the goals of Artificial Intelligence and in particular of commonsense reasoning is how to make an agent intelligent that may be autonomous and capable of acting in an open dynamic environment. As suggested in the logic-programming literature, such a goal requires a solid theoretical basis on knowledge representation and nonmonotonic reasoning, and in particular, in *knowledge updates*. Logic programming is a classical well-known mechanism to code and represent agents' knowledge by means of a set of clauses called logic program. Such a program might be called a knowledge base and we code it into a semantics called Answer Sets Programming [14] or ASP in short. However, logic programming has typically been static in the sense that it provides no mechanism to automatically make changes (belief revision or updates) to the knowledge base.

In particular, when updating knowledge one needs a way to avoid inconsistencies due to *potential contradictory information* upcoming from new evidence that is typically incomplete. Much work has been done in the context of logic

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 137-148

programming based on a common ASP basis by satisfying certain properties and postulates: [1, 6, 12, 13, 21, 22, 23, 24]. However, despite the existence of several semantics for updates [3] and a vast analysis of general properties [13, 20], we claim we are still far from having a general one that can satisfy many existing and well-known principles to represent *"correct" dynamic knowledge*.

For instance, one of the missing and obvious properties in current semantics for updates is *persistence* that others don't manage well for several reasons, mainly for their approach on *sequences of updates* [3]. Such a problem has been introduced and overcome by Sakama and Inoue [22] by means of a semantics based on an extended particular version of abductive logic programming [15]. However, their semantics is strongly based on syntactical changes that has other problems (described later) and lacks of a proper characterisation of principles for updates (or belief revision), presumably due to their different goals of their referred approach.

With the aim to define a *general characterised semantics* and to succeed in the mentioned persistence situation (and in many others more), this paper consists of an alternate more-general approach, founded on generalised answer sets, and based upon further well-known principles for belief change (AGM-postulates [5]) that make it *syntax independent*, more intuitive and have generally-accepted properties.

Besides satisfying most belief revision postulates, this paper exhibits an approach that consists in performing *successive updates* so that the semantics can deal with the problem that, according to Sakama and Inoue, produces counter-intuitive interpretations in most approaches.

Partial results of this article have already appeared in preliminary versions (without proofs or other properties) in [2], as well as in the extended abstract in [4]

2 Problem Description

As introduced in previous paragraphs, once there existed a strong theoretical *basis for belief revision and updates*, a few authors in the field of logic programming proposed mechanisms for updates with a vast analysis, *taxonomy* and comparison of various known semantics [13, 22, 24]. Some others even suggested new principles, like [6, 7], with alternate logic-programming frameworks to ASP. However, owing to the different foundations each semantics has (even within ASP), yet some problems arise to meet a significant number of well-accepted *principles for updates* [3].

Although it is unlikely to come off with a semantics that may satisfy all of them [13, 20, 22], a more general one that fulfils most of widely-accepted principles is still necessary, which is not an easy task.

For instance, researchers studying updates of logic programs following principles for updates[1] (coded into eight well-known postulates for updates called

[1] Actually the principles are for belief revision rather than updates. We explain the difference later.

AGM theory [5]) —and in other principles around it[2]— boil down to the difficulty in satisfying many of them by means of a *non-monotonic framework* like ASP, owing to the *monotonic nature* of the postulates themselves —[9, 13, 20, 22]. Nonetheless, Osorio and Cuevas [20] achieved an interpretation of six of the original eight AGM postulates in terms of the *monotonic* (non-classical) $\mathcal{N}_2$-logic, for general "update" operators. They have chosen (the monotonic) $\mathcal{N}_2$-logic apparently because it is one of several that characterise ASP and includes two types of negation: negation-as-failure and strong negation. The authors' results in terms of a general semantics, however, seem to be inconclusive [20].

On the other hand, the nearest proposal (to the best of our knowledge) that seemed[3] to meet most of the existing principles is due to Sakama and Inoue [22], who have introduced and overcome an interesting *persistence situation* that others fail to represent well for different reasons, as pointed out by themselves and in [2]. In particular, the main feature of that interesting situation is that Sakama and Inoue's semantics is capable of maintaining a *knowledge base* (coded into a logic program) throughout its own evolution, and that is also the main motivation behind A. Guadarrama [2] to propose his approach. They both lack, however, of a more-general belief-update characterisation, besides other problems.

Although their object-level approach makes it a good candidate to be considered by the belief-revision community, Sakama and Inoue's minimal-change principle is still *syntactic*: the *changes to a knowledge base* are to be minimal, as they themselves explain, and that has other problems. The main issue, however, in that they have no characterisation of their semantics with further general *belief-change principles*, arguing that they aren't applicable to nonmonotonic propositional theories in general [22].

In addition to that, a big disadvantage of *syntactical approaches*, as discussed in [13, 20] is that, in general, they do not satisfy the *structural properties* proposed in [20, 23], and Sakama and Inoue's approach is not an exception.

Regardless the polemic that approach might cause (especially in *planning* domains) and the deployment of extended-abduction properties in their article, the lack of further and more *general properties for belief change* makes it hard to compare with other alternatives for updates of logic programs. Indeed, their first goal (as they themselves explain) is the converse: to provide a mechanism of updates to characterise their *extended abduction framework* [22].

In need to define a *general semantics* and regardless the difference between belief revision and updates due to Katsuno and Mendelzon [18], this paper includes further results and a slightly different and fundamental approach from the preliminary alternate basic solution in [2, 4] within the same studied foundation of *Minimal Generalised Answer Sets* (MGAS hereafter, from Kakas and Mancarella [16]) and with an alternate approach to both Zacarías et al. [23] and A. Guadarrama [1], proposing a *simpler general formulation* that likewise per-

[2] For a nice analysis and compilation of such principles, see [13].

[3] To my knowledge, there is no evidence that their semantics satisfies most of them, although it does overcome most of the problems that other semantics present.

forms multiple updates, but at the object level rather than sequences of updates, which overcomes the kind of problems already described. Moreover, the simpler semantics meets the *structural properties for updates* proposed by [13, 20, 23], as well as the *satisfaction of five of the six most general belief revision postulates* and many other relevant properties.

3 Preliminaries

A main foundation of this proposal is the well-known AGM-postulates [5] in a particular interpretation and notation [4], followed by a brief basic background of Answer Sets and Generalised Answer Sets. Owing to space constraints, however, this paper excludes sections of three ASP's characterising logic systems (intuitionistic logic, Nelson's logic and $\mathcal{N}_2$), which are more evidence of the solid foundation earlier suggested. Finally, in this paper it is assumed that the reader is familiar with basic notions of AGM-theory, as well as logic programming and in particular with ASP.

3.1 Logic Programming and Answer Sets

The following formalism gives the description of ASP, which is identified with other names like *Stable Logic Programming* or *Stable Model Semantics* [14] and A-Prolog. Its formal language and some more notation are introduced as follows.

Definition 1 (ASP Language of logic programs, $\mathcal{L}_{\mathsf{ASP}}$). *In the following $\mathcal{L}_{ASP}$ is a language of* propositional logic *with* propositional symbols*: $a_0, a_1, \ldots$;* connectives*: "," (*conjunction*) and meta-connective ";";* disjunction*, denoted as "|"; $\leftarrow$ (*derivation*, also denoted as $\rightarrow$);* propositional constants $\bot$ *(*falsum*);* $\top$ *(*verum*); "¬" (*default negation *or* weak negation*, also denoted with the word* not*); "∼" (*strong negation*, equally denoted as "−");* auxiliary symbols*: "(", ")" (*parentheses*). The propositional symbols are called* atoms *too or* atomic propositions*. A* literal *is an atom or a* strong-negated atom*. A* rule *is an ordered pair $\mathsf{Head}(\rho) \leftarrow \mathsf{Body}(\rho)$.*

An intuitive meaning of *strong negation* "∼" in logic programs with respect to the default negation "¬" is the following: a rule $\rho_0 \leftarrow \neg\rho_1$ allows to derive ρ_0 when there is no *evidence* of ρ_1, while a rule like $\rho_0 \leftarrow \sim\rho_1$ derives ρ_0 only when there is an *evidence* for $\sim\rho_1$, i.e. when it can be proved that ρ_1 is false.

With the notation introduced in Definition 1, one may construct clauses of the following general form that are well known in the literature.

Definition 2 (EDLP). *An* extended disjunctive logic program *is a set of rules of form*

$$\ell_1 \vee \ell_2 \vee \ldots \vee \ell_l \leftarrow \ell_{l+1}, \ldots, \ell_m, \neg\ell_{m+1}, \ldots, \neg\ell_n \tag{1}$$

where ℓ_i is a literal and $0 \leq l \leq m \leq n$.

Naturally, an *extended logic program* (or ELP hereafter) is a finite set of rules of form (1) with $l = 1$; while an *integrity constraint* (also known in the literature as *strong constraint*) is a rule of form (1) with $l = 0$; while a *fact* is a rule of the same form with $l = m = n$. In particular, for a literal ℓ, the *complementary literal* is $\sim\ell$ and vice versa; for a set $\mathcal{M}$ of literals, $\sim\mathcal{M} = \{\sim\ell \mid \ell \in \mathcal{M}\}$, and $Lit_{\mathcal{M}}$ denotes the set $\mathcal{M} \cup \sim\mathcal{M}$; finally, a *signature* $\mathfrak{L}_{\Pi}$ is a finite set of literals occurring in Π. Additionally, given a set of literals $\mathcal{M} \subseteq \mathcal{A}$, the complement set $\overline{\mathcal{M}} = \mathcal{A} \setminus \mathcal{M}$.

Although we have introduced ASP as propositional (*ground*) programs, fixed *non-ground ASP-programs* of arbitrary *arity* are also considered in the same way than Dantsin et al. [10] do. Accordingly, non-ground ASP-programs with variables or constants as *arguments* are seen as a simplified expressions of larger ground (propositional) ones without variables, where each *ground program* Π is a set of its *ground rules* $\rho \in \Pi$. In addition, a ground rule is the set obtained by all possible substitutions of variables in ρ by constants occurring in Π [10].

Although ASP is a strong theoretical framework to represent knowledge, it can't model changes to that knowledge by itself, nor can it represent conflict situations amongst an agent and its dynamic environment. So, a more general and relaxed semantics is necessary, to choose potential models amongst *conflicting information*, which ought to reflect the general *principles* and *postulates* under consideration. In particular, a suitable framework to our purposes has been *Abductive Logic Programming* due to Kakas and Mancarella and is briefly presented in the following section.

3.2 Abductive Programs and MGAS

As one of the semantics to interpret abductive programs, *Minimal Generalised Answer Sets* (MGAS) provides a more general and flexible semantics than standard ASP, with a wide range of applications. This framework is briefly introduced in the following set of definitions.

Definition 3 (16). *An* abductive logic program *is a pair* $\langle\, \Pi, \mathcal{A}^* \,\rangle$ *where* Π *is an arbitrary program and* $\mathcal{A}^*$ *a set of literals, called abducibles.*

On the other hand, there already exists a semantics to interpret abductive programs, called *generalised answer sets* (GAS) due to Kakas and Mancarella.

Definition 4 (GAS, 16). *The expression* $\mathcal{M}(\Delta)$ *is a* generalised answer set *of the abductive program* $\langle\, \Pi, \mathcal{A}^* \,\rangle$ *if and only if* $\Delta \subseteq \mathcal{A}^*$ *and* $\mathcal{M}(\Delta)$ *is an answer set of* $\Pi \cup \{\alpha \leftarrow \top \mid \alpha \in \Delta\}$.

In case there are more than one generalised answer sets, a *preferred inclusion order* may be established:

Definition 5 (8). *Let* $\mathcal{M}(\Delta_1)$ *and* $\mathcal{M}(\Delta_2)$ *be generalised answer sets of* $\langle \Pi, \mathcal{A}^* \rangle$. *The relation* $\mathcal{M}(\Delta_1) \leq_{\mathcal{A}^*} \mathcal{M}(\Delta_2)$ *holds if and only if* $\Delta_1 \subseteq \Delta_2$.

Last, one can easily establish the *minimal generalised answer sets* from an abductive inclusion order with the following definition

Definition 6 (MGAS, 8). *Let $\mathcal{M}(\Delta)$ be a* minimal generalised answer set *(MGAS) of $\langle \Pi, \mathcal{A}^* \rangle$ if and only if $\mathcal{M}(\Delta)$ is a generalised answer set of $\langle \Pi, \mathcal{A}^* \rangle$ and it is minimal with respect to its* abductive inclusion order*.*

4 Updating Epistemic States

One of the main goals of this proposal is to meet most well-accepted *principles* for updates at the *object level* and in Minimal Generalised Answer Sets (MGAS), besides other relevant properties. The approach consists in setting up the needed models for the desired properties in an *iterated fashion*, rather than a sequence of updates, as earlier explained.

A first analysis of the problem at the object level, a solution, *justification, basic model-oriented properties* and *comparison with other semantics* are available in [2]. However, the semantics hasn't been characterised with more general principles, which is necessary both to avoid counterintuitive behavuour and to provide a common *frame of reference* to compare with other approaches. So, let us briefly introduce it, followed by a *characterisation of Belief Revision.*

The semantics is formally expressed with the following set of definitions, revised from [2] to make it simpler, precise, and to comply even more with the postulates, which is part of the *contribution of this paper* and the difference with the one in [2] and the extended abstract in [4]. So, let us start with some definitions.

An *α-relaxed rule* is a rule ρ that is *weakened* by a default-negated atom α in its body: $\mathsf{Head}(\rho) \leftarrow \mathsf{Body}(\rho) \cup \{\neg\alpha\}$. In addition, an *$\alpha$-relaxed program* is a set of α-relaxed rules. On the other hand, a *generalised program* of $\mathcal{A}^*$ is a set of rules of form $\{\ell \leftarrow \top \mid \ell \in \mathcal{A}^*\}$, where $\mathcal{A}^*$ is a given set of literals.

Accordingly, updating a program with another consists in transforming an ordered pair of programs into a single abductive program, as follows.

Definition 7 ($\bullet$-update Program). *Given an updating pair of extended logic programs, denoted as $\Pi_1 \bullet \Pi_2$, over a set of atoms $\mathcal{A}$; and a set of unique abducibles $\mathcal{A}^*$, such that $\mathcal{A} \cap \mathcal{A}^* = \emptyset$; and the α-relaxed program Π' from Π_1, such that $\alpha \in \mathcal{A}^*$; and the abductive program $\Pi_{\mathcal{A}^*} = \langle \Pi' \cup \Pi_2, \mathcal{A}^* \rangle$. Its* $\bullet$-update program *is $\Pi' \cup \Pi_2 \cup \Pi_G$, where Π_G is a generalised program of $\mathcal{M} \cap \mathcal{A}^*$ for some* minimal generalised answer set *$\mathcal{M}$ of $\Pi_{\mathcal{A}^*}$ and "$\bullet$" is the corresponding update operator.*

Obviously Definition 7 allows none or more *$\bullet$-update programs.* In addition to that, Corollary 2 below shows that the update is always consistent provided that Π_1 is also consistent. Before that, let us formalise another minor obvious property:

Corollary 1. *Let Π_G be a generalised program out of a minimal generalised answer set $\mathcal{M}$ from $\Pi_{\mathcal{A}^*}$ and $\mathcal{M}_1$ an answer set of Π_G. The following two statements hold:*

a) $\mathcal{M}_1 = \mathcal{M} \cap \mathcal{A}^*$.
b) $\mathcal{M}_1 \subseteq \mathcal{M}$.

Last but not least, the associated *models* $\mathcal{S}$ *of the new knowledge base* correspond to the answer sets of a $\bullet$-update program as follows.

Definition 8 ($\bullet$-update Answer Set). *Let* $\mathbf{\Pi}_\bullet = (\Pi_1 \bullet \Pi_2)$ *be an update pair over a set of atoms* $\mathcal{A}$. *Then,* $\mathcal{S} \subseteq \mathcal{A}$ *is a* $\bullet$-answer set of $\mathbf{\Pi}_\bullet$ *if and only if* $\mathcal{S} = \mathcal{S}' \cap \mathcal{A}$ *for some* minimal generalised answer set $\mathcal{S}'$ *of its* $\bullet$*-update program.*

Intuitively, this formulation establishes an order with respect to the *latest update* —which corresponds to Katsuno and Mendelzon [17, 18]'s first postulate (R∘1)— and with respect to a *minimal change* when choosing the most preferred model: MGAS.

5 $\bullet$-Properties

The following sets of properties of this simpler formulation are the main contribution of this current semantics for *iterated updates* of *epistemic states*. They are classified into a study of consistency issues, and the satisfaction itself of KM′-postulates.

Before the main results, there are two particular properties suggested much earlier in [23] that are necessary for the rest of them. Additionally, the reader should note that a statement like $\Pi_1 \equiv \Pi_2$ means that both Π_1 and Π_2 have the same answer sets —or alternately $\Pi_1 \equiv_{\mathsf{ASP}} \Pi_2$. By a slight abuse of notation, when establishing equivalence between updates, indeed it means that they have the same (or different) update answer sets. Finally, the two properties from the literature (ref. [19, 20]), interpreted in our own notation, are the following.

- $\bullet$-SP-8, Strong Consistency, SC: If $\Pi_1 \cup \Pi_2$ is *consistent*, then $\Pi_1 \bullet \Pi_2 \equiv \Pi_1 \cup \Pi_2$. The update coincides with the union when $\Pi_1 \cup \Pi_2$ is consistent.
- $\bullet$-SP-9, Weak Irrelevance of Syntax, WIS: Let Π, Π_1, and Π_2 be logic programs under the same language. If $T_{\mathcal{N}_2}(\Pi_1) \equiv_{\mathcal{N}_2} T_{\mathcal{N}_2}(\Pi_2)$ then $\Pi \bullet \Pi_1 \equiv \Pi \bullet \Pi_2$.

Theorem 1. *[[2]] Suppose that* Π, Π_1, Π_2 *and* Π_3 *are* ELP. *Operator* $\bullet$ *satisfies the properties* $\bullet$*-SP-8 and* $\bullet$*-SP-9.*

This preliminary result shall be helpful to simplify further properties in upcoming sections.

5.1 Dealing with Inconsistencies

As previously suggested, dealing with inconsistencies is something necessary not only when new information contradicts previous one, but also with an *originally-inconsistent knowledge base*. This section consists of a study of consistency-preservation and consistency-restoration as key properties of $\bullet$-operator.

In particular, *Weak-consistency View* guarantees consistency of the abductive program from an update pair. On the other hand, a *consistent abductive program* from a $\bullet$-update pair shall mean the abductive program with generalised answer sets.

Lemma 1 (Weak Consistency View). *Suppose Π_0 and Π_1 are ELP's and an updating pair $\Pi_0 \bullet \Pi_1$ with its corresponding abductive program $\Pi_{\mathcal{A}^*} = \langle \Pi' \cup \Pi_1, \mathcal{A}^* \rangle$. If Π_1 is consistent then $\Pi_{\mathcal{A}^*}$ is also consistent.*

Accordingly, the following result holds.

Corollary 2 (Consistency Preservation). *Suppose Π_0 and Π_1 are ELP's. The update $\Pi_0 \bullet \Pi_1$ is consistent if Π_1 is consistent.*

Proof. The proof is similar to the one in Lemma 1

This property is known in the literature as *Consistency Preservation* and by Sakama and Inoue as *Inconsistency Removal*. Note that the sole name of the latter confirms the *syntactical orientation* of their approach. Last, it's wort noticing that this property is equivalent to postulate ($\mathsf{R} \circ 3$) both Katsuno and Mendelzon's and Darwiche and Pearl's third postulate.

On the other hand, Π_1 inconsistent in Corollary 2 may lead to two possible situations: that the resulting update is either consistent or inconsistent, as shown in the following example.

Example 1 (Inconsistent Update). Suppose the update $\Pi_1 = \{a \leftarrow \neg a\}$, which has no answer sets, to an original *empty knowledge base* $\Pi_0 = \emptyset$. As a result, $\bot \models \Pi_0 \bullet \Pi_1$. Now suppose the same update to an initial knowledge base $\Pi_0' = \{a \leftarrow \top\}$. The $\bullet$-update answer set of such an update $\{a\} \models \Pi_0' \bullet \Pi_1$.

Corollary 2 also proves to be useful both when satisfying belief revision postulates and when *restoring consistency* from an originally inconsistent knowledge base, as explained below. On top of that, this property is a general case of Sakama and Inoue's *inconsistency removal* that makes syntactical changes to the original knowledge base.

Next, the following proposition follows directly from Corollary 2.

Proposition 1 (Consistency Restoration). *Suppose Π_0 is an ELP. The update $\Pi_0 \bullet \emptyset$ is consistent.*

Proof. The proof is similar to the one in Lemma 1

As described in this section, $\bullet$-operator guarantees *robustness of knowledge bases* in many situations where other alternate frameworks break down. Accordingly, the properties presented in this section shall be part of a more general framework of principles and postulates.

Next, the core of this paper is the introduction of a particular interpretation of one of the latest formulations of the AGM-postulates (KM') and which of them are met by $\bullet$-operator.

5.2 Principles

One of the main goals of this paper is a formulation of a semantics for updates of logic programs that can meet as-many-as possible general properties. So, let us start this section with an interpretation and characterisation of Katsuno and Mendelzon's postulates (R ∘ 1)–(R ∘ 6), as a main foundation to this *revision* of *epistemic states*.

(RG ∘ 1) $\Pi_1 \subseteq \Pi \circ \Pi_1$.
(RG ∘ 2) If $\Pi \cup \Pi_1$ is consistent, then $\Pi \circ \Pi_1 \equiv_{\mathsf{ASP}} \Pi \cup \Pi_1$.
(RG ∘ 3) If Π_1 is consistent, then $\Pi \circ \Pi_1$ is also consistent.
(RG ∘ 4) If $\Pi_x = \Pi_y$ and $\Pi_1 \equiv_{\mathcal{N}_2} \Pi_2$ then $\Pi_x \circ \Pi_1 \equiv_{\mathsf{ASP}} \Pi_y \circ \Pi_2$.
(RG ∘ 4′) If $\Pi_1 \equiv_{\mathcal{N}_2} \Pi_2$ then $\Pi \circ \Pi_1 \equiv_{\mathsf{ASP}} \Pi \circ \Pi_2$.
(RG ∘ 5) $\Pi \circ (\Pi_1 \cup \Pi_2) \subseteq (\Pi \circ \Pi_1) \cup \Pi_2$.
(RG ∘ 6) If $(\Pi \circ \Pi_1) \cup \Pi_2$ is consistent, then $(\Pi \circ \Pi_1) \cup \Pi_2 \subseteq \Pi \circ (\Pi_1 \cup \Pi_2)$.

An immediate result is the following main theorem of this paper certifying that •-operator satisfies five of these six belief revision postulates.

Theorem 2 (RG ∘ -properties). *Suppose that Π, Π_1 and Π_2 are ELP. Update operator "•" satisfies properties (RG ∘ 1)–(RG ∘ 4) and (RG ∘ 6).*

Proof. (RG ∘ 1) $\Pi_1 \subseteq \Pi \bullet \Pi_1$.
By Definition 7, $\Pi_1 \subseteq \Pi' \cup \Pi_G \cup \Pi_1$ that clearly satisfies the objective.
(RG ∘ 2) If $\Pi \cup \Pi_1$ is consistent, then $\Pi \bullet \Pi_1 \equiv \Pi \cup \Pi_1$.
This postulate corresponds to Strong-Consistency property and satisfied by Theorem 1.
(RG ∘ 3) If Π_1 is consistent, then $\Pi \bullet \Pi_1$ is also consistent.
This postulate is equivalent to Corollary 2.
(RG ∘ 4′) If $\Pi_1 \equiv_{\mathcal{N}_2} \Pi_2$ then $\Pi \bullet \Pi_1 \equiv \Pi \bullet \Pi_2$.
This postulate is equivalent to property •-SP-9 and satisfied by Theorem 1.
(RG ∘ 6) If $(\Pi \bullet \Pi_1) \cup \Pi_2$ is consistent, then $(\Pi \bullet \Pi_1) \cup \Pi_2 \subseteq \Pi \bullet (\Pi_1 \cup \Pi_2)$.
Suppose $(\Pi \bullet \Pi_1) \cup \Pi_2$ is consistent. Then, the abductive program of $\Pi \bullet \Pi_1$ is $\langle\, \Pi' \cup \Pi_1, \mathcal{A}^* \,\rangle$ with its respective MGAS's $\mathcal{M}(\Delta_1)$ that is an answer set of $\Pi' \cup \Pi_1 \cup \{\alpha \leftarrow \top | \alpha \in \Delta_1\}$ where $\Delta_1 \subseteq \mathcal{A}^*$ and its corresponding generalised program Π_{G_1}. By Definition 7, the update $\Pi \bullet (\Pi_1 \cup \Pi_2)$ has the abductive program $\langle\, \Pi' \cup (\Pi_1 \cup \Pi_2), \mathcal{A}^* \,\rangle$ with its MGAS's $\mathcal{M}(\Delta_2)$ that is an answer set of $\Pi' \cup \Pi_1 \cup \Pi_2 \cup \{\alpha \leftarrow \top \mid \alpha \in \Delta_2\}$ where $\Delta_2 \subseteq \mathcal{A}^*$ and its corresponding generalised program Π_{G_2}. Because Π_2 is consistent with the update $\Pi \bullet \Pi_1$, the number of abducibles in Δ_1 never change, and it's easy to verify that Δ_2 contains at least the same abducibles than Δ_1, $\Delta_1 \subseteq \Delta_2 \subseteq \mathcal{A}^*$ and thus $\Pi_{G_1} \subseteq \Pi_{G_2}$. In consequence, the respective •-update programs of each pair are $\Pi' \cup \Pi_1 \cup \Pi_2 \cup \Pi_{G_1}$ and $\Pi' \cup \Pi_1 \cup \Pi_2 \cup \Pi_{G_2}$, where $\Pi' \cup \Pi_1 \cup \Pi_2 \cup \Pi_{G_1} \subseteq \Pi' \cup \Pi_1 \cup \Pi_2 \cup \Pi_{G_2}$ as required. Therefore, $(\Pi \bullet \Pi_1) \cup \Pi_2 \subseteq \Pi \bullet (\Pi_1 \cup \Pi_2)$.

Nevertheless, postulate (RG ∘ 5) does not hold. As a counterexample, consider the following programs: $\Pi = \{a \leftarrow \top; {\sim}b \leftarrow \top; {\sim}c \leftarrow \top\}; \Pi_1 = \{b \leftarrow \top\}; \Pi_2 = \{c \leftarrow \top\}$. Such an update inverts the direction of the relation.

5.3 Discussion

This section is an introduction to new general properties characterising •-operator that go from the structural properties, most of them inherited from its equivalent counterpart in [23], to more general ones encoded in our particular interpretation of belief revision postulates. The satisfaction of AGM-postulates in ASP is something new and important, provided that other current approaches either don't meet most of them or have discarded them for considering that their *monotonic nature* is incompatible with non-monotonic frameworks like ASP, as previously discussed in Section 2.

Another issue other approaches have is when updating in a sequence rather than an iterated fashion, which leads to counterintuitive results, especially in the persistence situation and when new updates arise afterwards. By following the original AGM paradigm, we also claim that the iterative approach has other more natural properties than its sequenced counterpart.

The section is also a *study of inconsistencies* not only due to new information that contradicts current knowledge, but also from both an *originally-inconsistent knowledge base*, as well as *new originally-inconsistent observations* that not necessarily contradict current beliefs. The former is something that may be considered a key feature of belief revision. However, as one of the main goals of this paper is to provide a strong general framework to correctly represent knowledge, and making a strict distinction with belief update theory might result controversial.

On the other hand, dealing with *originally-inconsistent observations* might seem counterintuitive to some researches, but it does not mean that observing such contradictions may not be possible in a *changing environment*. Take for example two *concurrent observations* that contradict each other, updating a current knowledge base in, say, a problem of Ambient Intelligence when a sensor fails and another one contradicts it. Another example is an *observation that is inconsistent* due to a typo or another kind of human error. Traditionally, those problems are left to future debugging, but with a tendency to model even-more *autonomous entities*, tolerating inconsistencies is not only reasonable, but also necessary to preserve a knowledge base from collapse.

Bibliography

[1] A. Guadarrama, J. C. 2007a. Implementing knowledge update sequences. In *MICAI 2007: Advances in Artificial Intelligence*, A. Gelbukh and A. Kuri Morales, Eds. LNCS, vol. 4827. Springer-Verlag, Aguascalientes, Mexico, 260–270. 2, 3

[2] A. Guadarrama, J. C. 2007b. Maintaining knowledge bases at the object level. In *Special Session of the 6th International MICAI Conference*, A. Gelbukh and A. F. Kuri Morales, Eds. IEEE Computer Society, Aguascalientes, Mexico, 3–13. ISBN: 978-0-7695-3124-3. 2, 3, 6, 7

[3] A. Guadarrama, J. C. 2007c. A road map of updating in ASP. ISSN: 1860-8477 IfI-07-16, Institute für Informatik, TU-Clausthal, Clausthal, Germany. December. 35pp. 2

[4] A. Guadarrama, J. C. 2008. AGM postulates in answer sets. In *LANMR'08 Fourth Latin American Workshop on Non-monotonic Reasoning*, M. Osorio and I. Olmos, Eds. ISSN 1613-0073, vol. 408. CEUR, Benemérita Universidad Autónoma de Puebla, Puebla, México, 3pp. 2, 3, 4, 6

[5] Alchourrón, C. E., Gärdenfors, P., and Makinson, D. 1985. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic 50,* 2 (June), 510–530. 2, 3, 4

[6] Alferes, J. J., Banti, F., Brogi, A., and Leite, J. A. 2005. The refined extension principle for semantics of dynamic logic programming. *Studia Logica 79,* 1, 7–32. 2

[7] Alferes, J. J., Leite, J. A., Pereira, L. M., Przymusinska, H., and Przymusinski, T. C. 1999. Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming 45,* 1–3, 43–70. 2

[8] Balduccini, M. and Gelfond, M. 2003. Logic programs with consistency-restoring rules. In *Proceedings of the AAAI Spring 2003 Symposium.* AAAI Press, Palo Alto, California, 9–18. 5, 6

[9] Brewka, G. 2001. Declarative representation of revision strategies. *Journal of Applied Non-classical Logics 11,* 1–2, 151–167. 3

[10] Dantsin, E., Eiter, T., Gottlob, G., and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Computing Surveys 33,* 3 (September), 374–425. 5

[11] Darwiche, A. and Pearl, J. 1994. On the logic of iterated belief revision. In *Proceedings of the fifth Conference on Theoretical Aspects of Reasoning about Knowledge*, R. Fagin, Ed. Morgan Kaufmann, Pacific Grove, CA, 5–23. 8

[12] Delgrande, J. P., Schaub, T., Tompits, H., and Woltran, S. 2008. Belief revision of logic programs under answer set semantics. In *KR*. 411–421. 2

[13] Eiter, T., Fink, M., Sabbatini, G., and Tompits, H. 2002. On properties of update sequences based on causal rejection. *Theory and Practice of Logic Programming 2,* 6, 711–767. 2, 3, 4

[14] Gelfond, M. and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In *Logic Programming, Proceedings of the Fifth International Conference and Symposium ICLP/SLP*, R. A. Kowalski and K. A. Bowen, Eds. MIT Press, Seattle, Washington, 1070–1080. 1, 4

[15] Inoue, K. and Sakama, C. 1995. Abductive framework for nonmonotonic theory change. In *the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*. Morgan Kaufmann Publishers, Montreal, Canada, 204–210. 2

[16] Kakas, A. C. and Mancarella, P. 1990. Generalized Stable Models: A semantics for abduction. In *ECAI*. Stockholm, Sweden, 385–391. 3, 5

[17] Katsuno, H. and Mendelzon, A. O. 1989. A unified view of propositional knowledge base updates. In *the 11th International Joint Conference on Artificial Intelligence, IJCAI-89*, N. S. Sridharan, Ed. Morgan Kaufmann, Detroit, Michigan, USA, 1413–1419. 7

[18] Katsuno, H. and Mendelzon, A. O. 1991a. On the difference between updating a knowledge base and revising it. In *KR'91*. Morgan Kaufmann Publishers, Cambridge, Massachusetts, USA, 387–394. 3, 7

[19] Katsuno, H. and Mendelzon, A. O. 1991b. Propositional knowledge base revision and minimal change. *Artificial Intelligence 52,* 3, 263–294. 7, 8, 9

[20] Osorio, M. and Cuevas, V. 2007. Updates in answer set programming: An approach based on basic structural properties. *Journal of Theory and Practice of Logic Programming 7,* 4, 451–479. 2, 3, 4, 7

[21] Osorio, M. and Zacarías, F. 2004. On updates of logic programs: A properties-based approach. In *FoIKS*. Springer, Wilhelminenburg Castle, Austria, 231–241. 2

[22] Sakama, C. and Inoue, K. 2003. An abductive framework for computing knowledge base updates. *Theory and Practice of Logic Programming 3,* 6, 671–715. 2, 3, 8

[23] Zacarías, F., Osorio, M., A. Guadarrama, J. C., and Dix, J. 2005. Updates in Answer Set Programming Based on Structural Properties. In *7th International Symposium on Logical Formalizations of Commonsense Reasoning*, S. McIlraith, P. Peppas, and M. Thielscher, Eds. Fakultät Informatik, ISSN 1430-211X, Corfu, Greece, 213–219. 2, 3, 4, 7, 10

[24] Zhang, Y. and Foo, N. 2005. A unified framework for representing logic program updates. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-2005)*, M. M. Veloso and S. Kambhampati, Eds. AAAI Press / The MIT Press, Pittsburgh, Pennsylvania, USA, 707–713. 2

Generating CNC Code From a Domain Specific Language *

Gustavo Arroyo[1], J. Guadalupe Ramos[2], and Josep Silva[3]

[1] Centro Interdisciplinario de Investigación y Docencia en Educación Técnica,
Av. Universidad 282, CP 76000, Santiago de Querétaro, México,
garroyo@ciidet.edu.mx,
[2] Instituto Tecnológico de La Piedad,
Av. Tecnológico 2000, CP 59310, La Piedad, Michoacán, México
jgramos@pricemining.com
[3] Universidad Politécnica de Valencia,
Camino de Vera s/n, C.P. 46022, Valencia, España
jsilva@dsic.upv.es
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. Computerized Numeric Control (CNC) is an industrial language for the manufacturing of products. CNC programs are series of code that consist of assembler-like instructions and, consequently, they are low-level programs that require specialized developers in order to gain productivity in programs writing. In this work we introduce a domain specific language for the generation of CNC programs. The DSL itself has been developed in Curry, a declarative functional-logic language. Our DSL includes a set of functions that encapsulate CNC instructions raising the abstraction level, and therefore, improving productivity. It is designed in such a way that non-expert users can write CNC programs. We show how the use of a DSL allows us to perform the requirements capture of CNC systems and to reduce the gap between the requirements and the prototype. Finally, from our domain specific language we generate real CNC code in order to produce real world applications.

Keywords: CNC Programming; Domain Specific Languages

1 Introduction

The implementation of programs by employing computer programming languages is a technical task which is frequently delegated to programmers. Each programming language is founded on a set of technical features that influence the writing style of programs. Moreover, a program is profusely composed by technical sentences whose behavior strongly depends on a particular paradigm.

* This work has been partially supported by the Spanish *Ministerio de Ciencia e Innovación* under grant TIN2008-06622-C03-02, by the *Generalitat Valenciana* under grant ACOMP/2010/042, by the *Universidad Politécnica de Valencia* (Program PAID-06-08), by SES-ANUIES and by the Mexican *Dirección General de Educación Superior Tecnológica* under grant 2369.09-P.

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 151-161

Hence, a programmer is a person with a solid preparation in the technical aspects of a particular programming language.

Historically, rising the abstraction level of programming languages has been a common goal in all paradigms. This change of abstraction level allows the hiding of difficult and cumbersome details closer to the machine, and it allows non-expert programmers to construct solutions. A simple method to rise the abstraction level of programs is to produce interfaces that hide low-level instructions that are grouped into modules composing the so-called libraries of the language. A more sophisticated method consists in developing a new programming language that encloses principles and abstractions in a consistent way that is inspired on the target domain of application. These programming languages are known as *Domain Specific Languages* (DSL's) [6] and they provide a powerful solution to construct abstractions of higher level. A DSL incorporates the most common abstractions of a domain and it offers combinators which permit to construct programs and to produce interactions among abstractions.

Orthogonally, when companies develop large (or even medium) new software systems, it is relatively common to discover that the prototype does not fit the requirements which were contracted (by the customer) at the beginning of the project. This produces a critical situation that implies additional costs and time, and that could be avoided by performing an adequate requirements capture.

Indeed the design of domain specific languages is considered an activity for requirements capture [1]. However, although some DSL's are good for resembling the abstractions of the domain, sometimes they do not produce the final code of the tool, and thus they are only useful as languages for specification.

In this work we focus on the development of a domain specific language for Computer Numerical Control (CNC). Nowadays, CNC machines have become the basis of many industrial processes. CNC machines include robots, production lines, and all those machines that are controlled by digital devices. Typically, CNC machines have a *Machine Control Unit* (MCU) which inputs a CNC program and controls the behavior and movements of all the components of the machine. We consider that rising the abstraction level of the CNC language will allow us (1) to be able to develop more friendly CNC programs since we use abstractions of higher level, (2) to be more productive since we propose a library of functions, each of them encapsulating many simple CNC instructions, and (3) to demonstrate that DSL's are useful in order to produce real code and hence they permit to capture executable requirements.

We present our language as a set of functions that encapsulate CNC instructions, and that can generate real CNC code in order to produce real world applications.

As the host language of our DSL, we propose Curry [5], a multi-paradigm functional-logic language. Our choice relies on the fact that Curry is a high-level language which provides a very convenient framework to produce and (formally) analyze and verify programs. Moreover, it has many modern features such as, e.g., lazy evaluation (which allows us to cope with infinite data structures), higher-order constructs (i.e. the use of functions as first-class citizens, which al-

lows us to easily define complex combinators), type classes for arranging together types of the same kind, etc. Part of this work was inspired on [9] where authors present a DSL approach for routers specification in Curry. Some ideas in our designs are based on [3].

2 CNC: A brief review

Currently, as stated in standard ISO 6983 [4], CNC programs interpreted by MCUs are formed by an assembler-like code which is divided into single instructions called *G-codes* (see an example in Figure 1).

One of the main problems of CNC programming is the lack of portability. In general, each manufacturer of CNC machines introduces some extension to the standard G-codes due to the wide variety of functions and tools that CNC machines provide. Thus, when trying to reuse a CNC program, programmers have to first tune it for the MCU of their specific CNC machines. For example, even though both CNC machines HASS VF-0 and DM2016 are milling machines, the G-codes they accept are different because they belong to different manufacturers [3] (e.g., the former is newer and is able to carry out a wider spectrum of tasks).

CNC programming is a hard task because G-codes represent a low-level language without control statements, procedures, and many other advantages of modern high-level languages. In order to provide portability to CNC programs and to raise the abstraction level of the language, there have been several proposals of intermediate languages, such as APL [8] and OMAC [7], from which G-codes can be automatically generated with compilers and post-processors.

Computer numerical control is the process of having a computer controlling the operation of a machine [11].

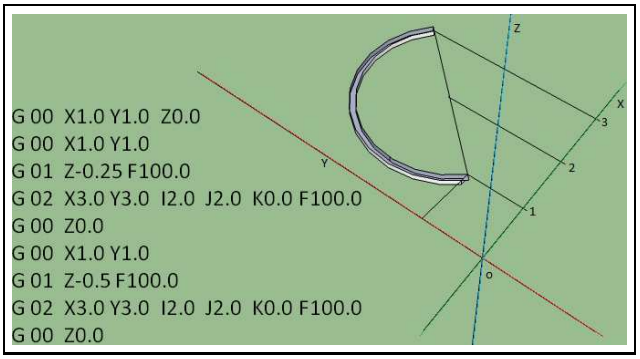


Fig. 1. Simple CNC Program

A CNC program is a series of blocks containing one or more instructions, written in assembly-like format [10]. These blocks are executed in sequential order, step by step. Each instruction has a special meaning, as they get translated

into a specific order for the machine. They usually begin with a letter indicating the type of activity the machine is intended to do, like F for feed rate, S for spindle speed, and X, Y and Z for axes motion. For any given CNC machine type, there are about 40-50 instructions that can be used on a regular basis.

2.1 G and M Codes

G words, commonly called G codes, are major address codes for preparatory functions, which involves tool movement and material removal. These include rapid moves, lineal and circular feed moves, and canned cycles. M words, commonly called M codes, are major address codes for miscellaneous functions that perform various instructions not involving actual tool dimensional movement. These include spindle on and off, tool changes, coolant on and off, and other similar related functions. Most G and M-codes have been standardized, but some of them still have a different meaning for particular controllers. As mentioned earlier, a CNC program is a series of blocks, where each block can contain several instructions. For instance,

```
N0030 G01 X3.0 Y1.7
```

is a block with one instruction, indicating the machine to do a movement (linear interpolation) in the X and Y axes. Figure 1 shows an example of a simple CNC program for cutting a half circle by means of linear and circular interpolation G codes (G 01 and G 02).

2.2 An example

In this section, we illustrate a CNC programming example. Due to the lack of space, it is not possible to show a real example of a complete CNC program—explaining the meaning of all involved G and M codes. Instead, we consider a simple CNC milling machine which can move the turret chuck in the X, Y and Z axes. The machine also handles absolute and incremental positioning of the turret chuck.

A CNC program for this machine consists of a header and a body. The header is optional and is usually a short comment, whilst the body is a list of blocks, where each block is identified by a number (`Nnnnn`). This number can be optional, and can contain either one or more instructions or a comment, where comments are always parenthesized. An instruction will contain one of the CNC codes shown in Figure 2.

A CNC program for cutting a circular arc from (1,1) to (3,3) with center in (2,2) is as follows:

```
N0010 (cutting a circular arc XY plane)
N0020 G90
N0030 G00 X1.0 Y1.0 Z0.0
N0040 G01 Z − 0.25 F50.0
N0050 G02 X3.0 Y3.0 I2.0 J2.0 K0.0 F50.0
N0070 G00 X1.0 Y1.0
N0080 G01 Z − 0.5 F50.0
N0090 G02 X3.0 Y3.0 I2.0 J2.0 K0.0 F50.0
N0095 G00 Z0.0
```

In this example, the block `N0010` denotes a comment; `N0020` instructs the CNC machine that absolute positioning is being used; `N0030` moves the turret

`G01`: moves the turret chuck along the XYZ axes. It can be followed by X, Y and Z codes, it represents a linear interpolation.
`G02`: moves the turret chuck along circular span lying in a plane parallel to one of the three principal planes of reference. It represents a clock wise circular interpolation.
`G90`: indicates that absolute positioning is being used.
`G91`: indicates that incremental positioning is being used.
`X(-)nn`: used to move the turret chuck along the X axis.
`Y(-)nn`: used to move the turret chuck along the Y axis.
`Z(-)nn`: used to move the turret chuck along the Z axis.

where `nn` is a number whose meaning depends on the type of positioning we are handling:

- In the case of *absolute* positioning, the number indicates the new absolute position in the corresponding axis, where (0,0,0) is a given reference point over the table.
- In the case of *incremental* positioning, the number indicates the number of units in the current axis that the tool is being shifted.

Fig. 2. Instructions set for a simple CNC milling machine

chuck at (1,1,0), 0 in the Z axis represents the surface of the piece to be machined; `N0040` moves the turret chuck 0.25 mm under the table in the Z axis, it makes a hole in (1,1); `N0050` starts the circular interpolation from the last position to the end position (3,3). The I,J,K words define the center of the arc. F represents the feed motion; `N0060` moves up the turret chuck; `N0070` moves the turret chuck at (1,1) position (the beginning of the arc); `N0080` moves down the turret chuck, this time 0.5 millimeters down in the Z axis; `N0090` starts again the circular arc 0.5 mm under the table in the Z axis; finally `N0095` places the turret chuck at Z0.0. So we get a circular arc cut 0.5 mm depth (see Figure 1).

3 A DSL for CNC programming embedded in Curry

In this section we introduce a domain specific language for CNC code generation, which is based on the standard ISO 6983 [4]. We focus particularly in design aspects (taking into account that the DSL is developed in Curry), explaining the main functions developed and showing examples in order to illustrate the applicability of our DSL.

3.1 The functional-logic language Curry

Curry is a functional-logic language that inherits the best properties of the most important declarative programming paradigms, i.e., logic and functional programming paradigms. Curry combines features of both paradigms like: nested expressions, lazy evaluation, higher-order functions from the functional programming and logical variables, partial data structures, built-in search from the logic

programming. Curry also has another powerful properties such as concurrent programming and besides, compared with functional programming: search, computing with partial information; compared with logic programming: more efficient evaluation due to the deterministic evaluation of functions. We refer the reader to [5] for an introduction to Curry.

3.2 Using Curry as the host language of the DSL

The Curry data structure that we define to hold a CNC program is shown in Figure 3.

```
data CNCprogram = Header Body
data Header  = Maybe Comment
type Comment = String
data Maybe a = Nothing | Just a
type Body    = [Command]
type Command = [Instruction]
data Instruction = N Int | G String | X Float | Y Float | Z Float
| U Float | V Float | W Float | P Float | Q Float | R Float | A Float
| B Float | C Float | I Float | J Float | K Float | F Float | S Float
| T Float | M String
```

Fig. 3. Curry data structure for representing a CNC program

In our setting, a CNC program is compound of a header and a body [2]. A `Header` is an optional comment—a `Comment` is a String—, when it is missing the `Nothing` constructor is used. A `Body` is a set of blocks, each block being a `Command` or a set of instructions. Finally, an `Instruction` is defined as one of the possible codes defined in the ISO 6983 standard [4].

For instance, we invoke the function `DrawHole` as follows:

```
DrawHole (1.0,1.0,0.0,1.5,10.0,0.5)
```

This command drills a hole in (1,1,0) with a 1.5 units depth, and a 0.5 units wide tool. It produces the following code:

```
[(G "00"), (X 1.0), (Y 1.0), (Z 0.0)]
[(G "01"), (Z (−0.25)), (F 10.0)]
[(G "01"), (Z (−0.5)), (F 10.0)]
[(G "01"), (Z (−0.75)), (F 10.0)]
[(G "01"), (Z (−1.0)), (F 10.0)]
[(G "01"), (Z (−1.25)), (F 10.0)]
[(G "01"), (Z (−1.5)), (F 10.0)]
```

The code moves the turret chuck to (1,1,0), then makes a hole with linear interpolation -0.25 units in Z axis to feed speed 10.0 units per minute.

Roughly analyzing the `DrawHole` function, it starts with the *InitPOS* procedure, it calls the *writefile* function definition which takes the *GotoXYZ* function as parameter. Note that, because Curry is a higher-order language, it can accept functions as arguments. *GotoXYZ* yields the first CNC command. Then it executes the *Perforate* procedure; because the width of the tool is less than the depth of the hole, the program control executes *FAUX_DrawHole* producing the

rest of CNC commands until the depth is reached (details in the URL indicated at section 4).

It is worth to note that function *writefile* appends each CNC command in a file each time it is executed. Auxiliary functions are shown in Figure 4.

```
-- erasefile cleans CNC file
erasefile :: IO()
erasefile = writeFile "CNCCapture.txt" "%\n"
-- Go to X, Y, Z
GotoXYZ :: (Float, Float, Float) -> Command GotoXYZ
(x,y,z) = [G "00", X x, Y y, Z z]
-- Drill Z with a feedrate
DrawZ :: (Float, Float) -> Command
DrawZ (z,feedrate) = [G "01", Z (0-.z), F feedrate]
-- writefile writes CNC commands in a text file
writefile :: Command -> IO()
writefile(NewCommand) =appendFile "CNCCapture.txt" (show(NewCommand)++"\n")
```

Fig. 4. `writefile` and other auxiliar functions

3.3 Functions of the DSL

In this section we specify the function definitions of our DSL that is currently under development and we show the CNC code that it generates from a list of simple parameters. The intention of such functions is to simplify the way in which programming is performed for a piece of machining. Generally all the parts that take a manufacturing process can be formed from the combination of simple geometric figures, such as linear, circular and parabolic functions.

From this idea we are designing a DSL whose functions generate CNC code based on the ISO 6983 standard [4]. These DSL instructions produce: linear and circular cuts, make simple geometric figures (rectangles, circles), and canned (circular or rectangular). In the following we describe some representative functions and their parameters.

Configuring functions. These functions are useful for specifying general parameters during the programming process. For instance, they indicate whether the positioning is absolute or relative, whether they require measures in inches or millimeters and so on. Some Curry definitions of this functions are shown in the following chart:

```
—Selecting the kind of movement
Movement :: (String “Absolute” | “Relative”) –> IO()

—Selecting the kind of compensation
Compensation :: (String “Center” | “Left” | “Right”) –> IO()

—Selecting measurement unit
Unit :: (String “Millimeters” | “Inches”) –> IO()
```

Function for making a hole in a specific point

```
DrawHole :: (Float,Float,Float,Float,Float,Float) –> IO()
DrawHole (InitX,InitY,InitZ,Height,Feedrate,dx)
```

where *InitX, InitY, InitZ* represent the initial position of the CNC machine tool, *Height* is the depth of the drilling, *Feedrate* is the speed rate of the tool and *dx* represents the width of the cutting tool.

Function for lineal cutting

```
DrawLine :: (Float, Float, Float, Float, Float, Float, Float, Float) –> IO()
DrawLine (InitX, InitY, InitZ, EndX, EndY, Height, Feedrate, dx)
```

where *InitX, InitY, InitZ* indicate the initial position of the CNC machine tool, *EndX, EndY* represent the final position for the cutting, *Height* is the depth of the drilling, *Feedrate* is the cutting speed for the CNC machine tool and *dx* represents the width of the cutting tool.

Function for making an arch of a circle

```
DrawArc :: (Float, Float, Float, Float, Float, Float, Float, Float, Float, String, Float,
            Float) –> IO
DrawArc (Initx, Inity, Initz, Endx, Endy, i, j, k, Height, TSpin, Feedrate, dx)
```

where *InitX, InitY, InitZ* indicate the initial position of the CNC machine tool, *Endx, Endy* represent the X,Y final coordinates of the arch, *i, j, k* represent the coordinates of the circle, *Height* represents the depth of the drilling, *TSpin* represents the orientation of the spin, *Feedrate* represents the cutting speed of the CNC machine tool and *dx* represents the width of the cutting tool.

Function for making a rectangular canned

```
DrawBox :: (Float, Float, Float,Float, Float, Float, Float,Float) –> IO()
DrawBox (Initx, Inity, Initz, Length, Width, Height, Feed, dx)
```

where *InitX, InitY, InitZ* indicate the initial position of the CNC machine tool, *Length* represents the length of the box, *Width* represents the width of the box, *Height* represents the depth of the drilling, *Feedrate* represents the cutting speed and *dx* represents the width of the cutting tool.

Function for making a circular canned

```
DrawCylinder :: (Float, Float, Float, Float, Float, Float, Float)->IO()
DrawCylinder (i, j, k, Height, Radius, Feedrate, dx)
```

where *i, j, k* indicate the center of the circle, *Height* represents the depth of the drilling, *Feedrate* represents the cutting speed and *dx* represents the width of the cutting tool.

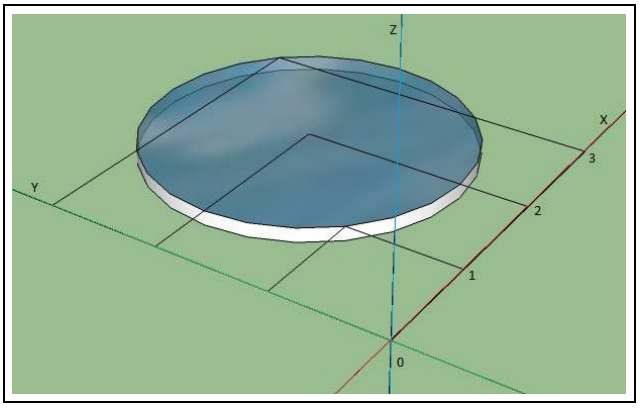


Fig. 5. Circular canned

3.4 Using DSL Functions

The following DSL Curry function

```
DrawCylinder (2.0,2.0,0.0,0.25,1.0,0.5)
```

makes a circular canned (see Figure 5). It defines a center of the circular sector in (2,2,0) with a 0.25 units depth with 1.0 units radius and a tool 0.5 units wide. This DSL function produces de following CNC code:

```
[(G "00"),(X 2.0),(Y 2.0),(Z 0.0)]
[(G "01"),(Z (-0.25)),(F 40.0)]
[(G "01"),(X 2.5),(F 120.0)]
[(G "03"),(X 2.5),(I 2.0),(J 2.0),(K 0.0),(F 120.0)]
[(G "01"),(X 3.0),(F 120.0)]
[(G "03"),(X 3.0),(I 2.0),(J 2.0),(K 0.0),(F 120.0)]
[(G "00"),(X 2.0),(Y 2.0),(Z 0.0)]
```

`G 03` makes a circular interpolation counter clock wise, because the tool is 0.5 wide with a hole in the center and just two `G 03` commands are enough to make the circular canned.

Another example is shown in Figure 1, where CNC code to produce a circular arc can be generated with a single call to the following DSL function:

```
DrawArc (1.0,1.0,1.0,3.0,3.0,2.0,2.0,0.0,0.5,"CW",100.0,0.5)
```

It generates the following CNC code:

```
[(G "00"),(X 1.0),(Y 1.0),(Z 1.0)]
[(G "00"),(X 1.0),(Y 1.0)]
[(G "01"),(Z (-0.25)),(F 100.0)]
[(G "02"),(X 3.0),(Y 3.0),(I 2.0),(J 2.0),(K 0.0),(F 100.0)]
[(G "00"),(Z 1.0)]
[(G "00"),(X 1.0),(Y 1.0)]
[(G "01"),(Z (-0.5)),(F 100.0)]
[(G "02"),(X 3.0),(Y 3.0),(I 2.0),(J 2.0),(K 0.0),(F 100.0)]
[(G "00"),(Z 1.0)]
```

Clearly, the DSL improves the readability of programs, and reduces the size of the code. In this example, using the DSL allows us to perform all the work with a single DSL instruction. Figure 6 shows the Curry code that implements the `DrawArc` DSL function. Observe that an important advantage of the DSL is that we can re-implement all the functions for another specific CNC machine but keeping the same signature. This means that the DSL provides a portability that allows us to use the same DSL programs in different CNC machines.

```
-- DrawArc draws an arc where i,j,k are the origin of the radio
-- TSpin:  CW=ClockWise CCW=CounterClockWise
DrawArc :: (Float, Float, Float, Float, Float,Float,Float,Float,Float,
            String, Float, Float) -> IO()
DrawArc (Initx, Inity, Initz, Endx, Endy, i, j, k, Height, TSpin,
          Feedrate, dx) = do InitPOS
                             Perforate
where
 InitPOS = writefile(GotoXYZ(Initx, Inity, Initz))
 Perforate =  if (Height<dx) then do
     writefile(DrawZ(Height,Feedrate))
     writefile([Spin(TSpin),X Endx, Y Endy, I i, J j, K k, F Feedrate])
-- if tool width is less than depth
       else do FAUX_DrawArc(Initx,Inity,Initz,Endx,
               Endy,i,j,k,Height,TSpin,Feedrate,(0.5*.dx),(0.5*.dx))
```

Fig. 6. The Curry code of the DSL function `DrawArc`

4 Conclusions

In this work we introduced a DSL developed in Curry as a high-level language for the design of CNC programs. We have shown that using the DSL provides important advantages over pure CNC programming.

The DSL provides two main advantages in the development of CNC programs. The first advantage is the possibility of programming at a high abstraction level. This improves the productivity, i.e., to provide more commands to the CNC machine (more CNC code) with less instructions and readability. This goal has been reached since the DSL functions encapsulate procedures that automatically produce CNC code for a specific and non-simple task.

The second advantage is reducing the gap between user requirements and the developed prototypes. This situation is a common trouble in the software engineering field because the analyst that captures the requirements and the programmer that programs the prototype are different persons. The use of DSLs during the analysis allows the analyst to produce rapid prototypes without the need of being a CNC programmer. This can be done thanks to the higher ab-

straction level of the DSL that allows requirements to become the so-called executable requirements.

Preliminary experiments are encouraging and point out the usefulness of our approach. However, there is plenty of work to be done, such as augmenting our library with other useful functions for making geometric figures, defining functions for other CNC machines (lathes, milling machines, etc), defining a graphical environment for simplifying the task of designing CNC programs, etc. Some other examples and the source code of our Curry DSL library is publicly available at the following URL: http://www.ciidet.edu.mx/Sitio_DSL/CNC_DSL.htm

Acknowledgment

We are grateful to Antonio Ávalos Olguín, Professor of the Technological Institute of Querétaro for his comments and support in the validation of CNC programming. Also to Marco Antonio Llanes Rodríguez for his valuable cooperation in the development of the Curry library.

References

1. C. Wohlin A. Aurum. *Engineering and Managing Software Requirements*. Springer-Verlag, 2005.
2. G. Arroyo. Diseño de un Lenguaje de Especificación para CNC. IP Report (in Spanish), DSIC-UPV, 2004.
3. G. Arroyo, C. Ochoa, J. Silva, and G. Vidal. Towards CNC Programming Using Haskell. In Lemaitre Christian, Reyes Carlos A., and Gonzalez Jesus A., editors, *Advances in Artificial Intelligence-IBERAMIA 2004*, pages 386–395. Springer LNCS 3315, 2004.
4. International Standardization for Organizations. Technical committee: ISO 6983-1/TC 184/SC 1. Numerical control of machines – Program format and definition of address words, September 1982.
5. M. Hanus (ed.). Curry: An Integrated Functional Logic Language. Available at http://www-ps.informatik.uni-kiel.de/currywiki/, 2010.
6. P. Hudak. Modular Domain Specific Languages and Tools. In *Procedings of Fifth International Conference on Software Reuse*, pages 134–142. IEEE Computer Society Press, 1998.
7. J. Michaloski, S. Birla, C.J. Yen, R. Igou, and G. Weinert. An Open System Framework for Component-Based CNC Machines. *ACM Computing Surveys*, 32(23), 2000.
8. T.P. Otto. An apl compiler. In International Conference on APL, editor, *Proceedings of the international conference on APL-Berlin-2000 conference*, pages 186–193. ACM Press - New York, NY, USA, 2000.
9. J. G. Ramos, J. Silva, and G. Vidal. An Embedded Language Approach to Router Specification in Curry. In *SOFSEM*, pages 277–288, 2004.
10. W. Seames. *CNC: Concepts and Programming*. Delmar Learning, 1994.
11. M. Weck, J. Wolf, and D. Kiritsis. STEP-NC The STEP Compliant NC Programming Interface: Evaluation and Improvement of the Modern Interface. In *Proc. of the ISM Project Forum 2001*, 2001.

PSO-Designed Operators for Image Edge Detection

Victor Ayala-Ramirez, Ezequiel Martinez-Ayala, and Raul E. Sanchez-Yanez

Universidad de Guanajuato, DICIS, Department of Electronics Engineering,
Carr. Salamanca-Valle, Km. 3.5+1.8, 36700, Salamanca, Mexico
E-mail: {ayalav,sanchezy}@salamanca.ugto.mx,ezequiel@laviria.org
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. We present here a method that automatically synthesize complex spatial operators from the evolution of a particle swarm and a thresholding value to perform edge detection on images. The proposed approach optimizes the values of the support of a complex spatial filter in order to emulate the response of a given edge operator on a training image. We present the results of using both a generic Canny edge operator and a Sobel edge response on a training image and the evaluation of the evolved edge operator on a set of test images. We measure the performance of edge operators using a Precision-Recall metric. First experiments show that the evolved filters are qualitatively good and that PSO can be used to emulate image processing tasks. We present both qualitative and quantitative results supporting this.

Keywords: Particle Swarm Optimization; Complex spatial filter; Edge detection;

1 Introduction

Particle Swarm Optimization (PSO) is an evolutionary computation technique originally proposed by Kennedy and Eberhardt in 1995 [3] that is useful for nonlinear function optimization in discrete and continuous spaces. PSO technique is inspired from the social behavior of schools of fishes and flocks of birds. In particular, the technique emulates how each organism moves in coordination with all their companions when they look for food or for a refuge.

There are some works in the literature where PSO has been used to perform image processing and computer vision tasks. For example, Ye *et al.* [8] have proposed to use PSO as an alternative to methods for threshold selection techniques, like the one by Otsu. Some other authors (e.g. Wei *et al.* [7], Tang *et al.*, Zhang and Liu and Zheng *et al.*) propose the utilization of PSO in image segmentation tasks. Zheng *et al.* [10] applyy a PSO-based method to the segmentation of CT and MRI images. Zhang and Liu [9] dealt with the problems arising in underwater applications. Tang *et al.* [2] address the problem of multilevel thresholding by posing it in terms of PSO.

Even edge detection has been addressed by some authors. For example, Alipoor *et al.* [1] have proposed a method to synthesize a $2D$ spatial filter as in

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 163-170

this work. Nevertheless, they address the problem by using a real-valued mask. Another approach to edge detection has been presented by Setayesh *et al.* [6]. They propose to perform a PSO on homogeneity measures and a uniformity factor of a contour curve.

In this work, we propose to simultaneously design a complex-valued edge operator and to optimize the thresholding level in order to emulate the response of a known edge operator on a training set of images. For this purpose, we show how to emulate a Canny edge operator and a Sobel edge operator. The fitness of each particle is evaluated by using the F metric (proposed by Martin *et al.* [5]). Our experiments over a set of testing images from the Berkeley database [4] show the usefulness of the proposed method.

Rest of this work is organized as follows: in Section 2, we review the edge detection problem in terms of a PSO-based approach. Section 3 presents the application of the method to the design of two optimal edge operators. Performance evaluation of the implemented system is addressed in Section 4 as a demonstration of the validity of the proposed approach. Finally, Section 5 presents our conclusions about this work and the main aspects to be covered in future work.

2 Formulation

2.1 Edge detection

Edge detection is an essential task for image processing. This task is typically performed by convolving a $2D$ spatial operator M (defined over a given spatial support) to the input image I. The resulting image O can be expressed as:

$$O = I \otimes M \tag{1}$$

with $\otimes$ being the convolution operator. The edge detection operator M is typically characterized by a set of coeffiecients that are the spatial weights used to perform the $2D$ convolution operation. When the edge detection operation must provide a binary output, a thresholding step is needed. This process requires to set a threshold value T to decide if a pixel in the result image is considered as an edge or not.

2.2 PSO for finding an optimal edge detection operator

The objective of this work is to find the optimal parameters of an edge detection operator with respect to a training set of images $S = \{S_1, S_2, \ldots, S_m\}$. For each image S_k on the training sets S, we have a corresponding image $R_k \in R = \{R_1, R_2, \ldots, R_m\}$, that is the expected result of the edge detection process on image S_k. The elements in R can be obtained by the application of a reference operator to the images in the training set, or by any other arbitrary process, such as, the generation of the images by an expert.

In the case of binary edge detection tasks, the PSO algorithm will optimize the spatial filter coefficients of the mask M and the thresholding value T. We

$$M = \begin{bmatrix} m_1 & m_2 & m_3 \\ m_8 & m_9 & m_4 \\ m_7 & m_6 & m_5 \end{bmatrix}$$

Fig. 1. Spatial localization of the weighting coefficients for the mask M.

consider a complex spatial mask M with complex coefficients $m_i \in C,\ i = \{1, \ldots, 9\}$. The localization of the spatial coefficients in the mask is shown in Figure 1. The thresholding value T is a real number $T \in [0, I_{max}]$ with I_{max} being the maximal intensity level hat can be present in an image.

Each particle P represents then a set of parameters for the mask coefficients and for the thresholding level. For a given particle P_j, a feasible solution of the optimization problem, we can obtain an output image O_j when a training image I is applied. O_j is computed as follows:

$$O_j = T(I \otimes M) \tag{2}$$

$$O_j(x, y) = \begin{cases} 1 & \text{if } (I \otimes M)(x, y) > T \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

$O_j(x, y)$ being the pixel at spatial position (x, y) of the output image O_j.

Let O_k^P be the result of applying the edge operation process defined by the parameter set encoded by the particle P to the training image S_k. That is.

$$O_k^P = T_P(S_k \otimes M_P) \tag{4}$$

$$O_k^P(x, y) = \begin{cases} 1 & \text{if } (S_k \otimes M_P)(x, y) > T_P \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The fitness score $f(P)$ of a particle P is defined as:

$$f(P) = \sum_{k=1}^{m} F(O_k^P, R_k) \tag{6}$$

with $F(A, B)$ being a similarity metric between images A and B. We will describe below what is the metric $F(\cdot)$ used.

The optimal particle P^* will encode a parameter set consisting of an optimal mask M^* and an optimal threshold value T^*, such that:

$$P = P^* \iff f(P^*) = \max_P \sum_{k=1}^{m} F(O_k^P, R_k) \tag{7}$$

The optimal edge detection process defined by the optimal particle P^* genes can then be used to detect edeges on any input image.

2.3 Similarity Metric Between Images

Martin *et al.* [5] have proposed the F-metric to measure similarity between two edge images A and B. This methodology takes an image, let us say A, as the reference image for the comparison. In this image, the F metric only considers the edge pixels present on it. For the compared image B, we count three quantities:

TP the number of true positive edge pixels. That is, the number of edge pixels that appear both in A and B.
FP the number of false positive edge pixels. That is, the number of edge pixels that appear in B but that are not present in A.
FN the number of false negative edge pixels. That is, the number of edge pixels that appear in A but that are not present in B.

These values are used to compute two figures of merit P (Precision) and R (Recall), accordingly to:

$$P = \frac{TP}{TP + FP} \tag{8}$$

$$R = \frac{TP}{TP + FN} \tag{9}$$

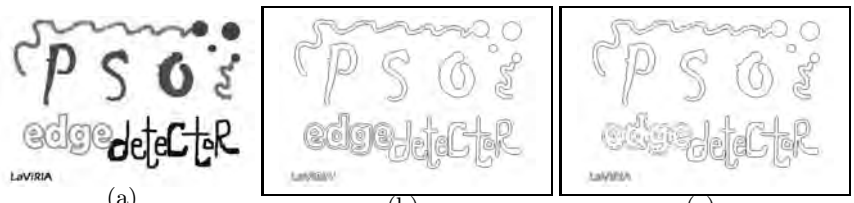
The F similarity metric combines P and R values in a single figure of merit, that takes values $F = [0, 1]$. A value 1 represents that both images are identical, and a score of 0 implies no matching edge pixels in both images. F value is computed as follows:

$$F = \frac{1}{2}\frac{PR}{P + R} \tag{10}$$

3 Implementation

We have run a PSO algorithm that uses as genes the complex coefficients of the spatial filter mask and the thresholding level. We have developed two test cases, both of them use a single training image S_1 (shown in Figure 2), but changing the process to obtain the image to be used as reference:

I A Canny-emulation filter, where a Matlab Canny operator is applied to S_1 in order to get a reference image, namely R_1^1.
II A Sobel-emulation filter, where a Matlab Sobel operator is applied to S_1 in order to get a reference image, namely R_1^2.

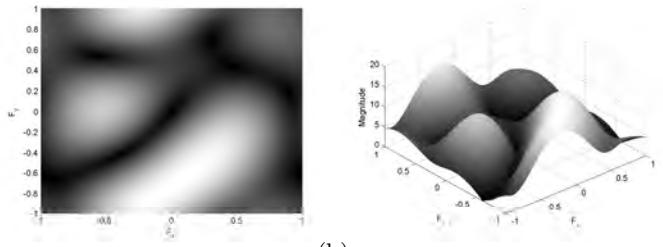


(a) (b) (c)

Fig. 2. (a) Original training image S_1, (b) the training references R_1^1 used to evolve the edge operator in case I, and (c) the training references R_1^2 used to evolve the edge operator in case II.

$$B_1^* = \begin{bmatrix} -1.9 + j\,1.0 & 2.7 - j\,1.4 & 2.5 - j\,0.7 \\ -1.1 + j\,1.0 & -0.8 + j\,2.5 & 1.6 + j\,2.8 \\ -2.9 - j\,2.0 & -0.8 - j\,2.3 & 0.9 - j\,1.0 \end{bmatrix}$$

(a)



(b)

Fig. 3. Optimal edge operator found in case I.

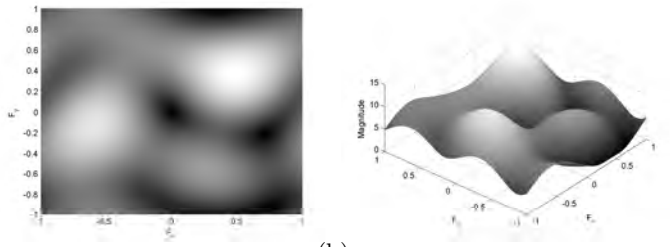
For each of these cases, an optimal edge filter, namely B_1^* and B_2^* were obtained for the case I and case II, respectively.

The coefficients of the spatial filter B_1^* are shown in Figure 3(a). Figure 3(b) is a graph of the frequency response of B_1^*. The optimal thresholding constant found by the PSO algorithm for this case was $T = 70.0$.

In the case II, the optimal set of coefficients of the spatial filter B_2^* are shown in Figure 4(a). In Figure 4(b), we can see a graph of the frequency response of B_2^*. The optimal thresholding constant found by the PSO algorithm for this case was $T = 47.0$.

$$B_2^* = \begin{bmatrix} 0.4 + j\,2.2 & 0.6 + j\,3.0 & 2.1 - j\,1.6 \\ -2.2 + j\,2.7 & 0.3 - j\,2.8 & 1.6 - j\,1.3 \\ 0.5 + j\,1.2 & -1.8 - j\,2.9 & -1.3 - j\,0.6 \end{bmatrix}$$

(a)



(b)

Fig. 4. Optimal edge operator found in case II.

4 Test and Results

We have applied the optimal edge detectors found in both cases (case I and case II) to a set I of test images, $I = \{I_1, \ldots, I_5\}$ selected from the Berkeley database. We have also used two sets of ground-truth images: U^1 (obtained from the application of a Matlab Canny edge operator) for test case I and U^2 (obtained from the application of a Matlab Sobel edge operator) for test case II.

Table 1 shows a qualitative summary of the results of the application of the optimal B_1^* and B_2^* edge operators and the expected results of using an optimal Matlab edge operator. As we can see there, results of the PSO-designed edge detectors are similar in appearance to the ground truth images. In order to compare quantitatively the results, Table 2 shows the F measures between images in the second and third columns in Table Qualitative and the fourth and fifth columns of the same table. We can observe that F measures are better for the Spbel like edge operator. That could be explained because Canny is not typically computed as a spatial filter. Conversely, Sobel was originally proposed as a spatial filter

These results were all obtained using 20 particles and 20 generations for the particle swarm evolution. Given this, we expect to improve F measures if we augment any of these PSO configuration parameters in further experiments.

5 Conclusions and Perspectives

We have presented an approach to obtain custom edge detectors designed using a PSO algorithm. The PSO algorithm optimizes simultaneously the parameters

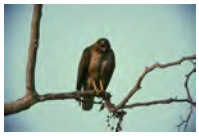
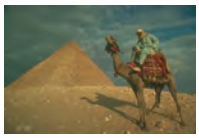
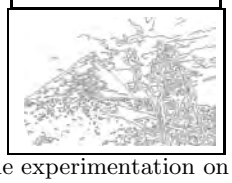
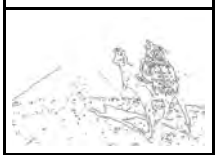
k	I_k	$I_k \otimes B_1^*$	U_k^1	$I_n \otimes B_2^*$	U_k^2
1					
2					
3					
4					
5					

Table 1. Qualitative results of the experimentation on a set of test images.

Test Image	$F(I_k, U_k^1)$	$F(I_k, U_k^2)$
I_1	0.399	0.602
I_2	0.139	0.580
I_3	0.115	0.560
I_4	0.185	0.504
I_5	0.220	0.585

Table 2. Results of the comparison between results images and their respective ground truth using the F metrics.

of a spatial filter and a thresholding step. We have shown the usefulness of the approach to emulate the response of a Canny and a Sobel edge operators. In both cases, qualitative and quantitative tests have been performed to validate our approach. Preliminary results show to be promising and we expect to work in the near future in tuning the PSO to improve its performance evaluation. The approach presented here could be easily extended for other image processing and computer tasks.

Acknowledgments

Ezequiel Martinez-Ayala gratefully acknowledges Mexico's CONACYT for the financial support through the scholarship 329514/229785.

References

1. Alipoor, M., Imandoost, S., Haddadnia, J.: Designing edge detection filters using Particle Swarm Optimization. In: Proc of the 18th Iranian Conference on Electrical Engineering (ICEE). pp. 548 –552 (May 2010)
2. Hongmei, T., Cuixia, W., Liying, H., Xia, W.: Image segmentation based on improved PSO. In: Proc. of the 2010 Int. Conf. On Computer and Communication Technologies in Agriculture Engineering (CCTAE). vol. 3, pp. 191 –194 (2010)
3. Kennedy, J., Eberhart., R.: Swarm Intelligence. Morgan Kaufmann Publishers, Inc., San Francisco, CA (2001)
4. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. 8th Int'l Conf. Computer Vision. vol. 2, pp. 416–423 (July 2001)
5. Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Trans. on. Pattern Analysis and Machine Intelligence 26(5), 530–549 (2004)
6. Setayesh, M., Zhang, M., Johnston, M.: A new homogeneity-based approach to edge detection using PSO. In: Proc. of 24th Int. Conf. on Image and Vision Computing IVCNZ '09. pp. 231 –236. New Zealand. (Nov 2009)
7. Wei, K., Zhang, T., Shen, X., Liu, J.: An improved threshold selection algorithm based on Particle Swarm Optimization for image segmentation. In: Proc. of 3rd Int. Conf. onNatural Computation, 2007. ICNC 2007. vol. 5, pp. 591 –594 (Aug 2007)
8. Ye, Z., Chen, H., Liu, W., Zhang, J.: Automatic threshold selection based on Particle Swarm Optimization algorithm. In: Proc. of the 2008 Int. Conf. on Intelligent Computation Technology and Automation (ICICTA). vol. 1, pp. 36 –39 (Oct 2008)
9. Zhang, R., Liu, J.: Underwater image segmentation with maximum entropy based on Particle Swarm Optimization (PSO). In: First Int. Multi-Symposiums on Computer and Computational Sciences, 2006. IMSCCS '06. vol. 2, pp. 360 –636 (2006)
10. Zheng, L., Pan, Q., Li, G., Liang, J.: Improvement of grayscale image segmentation based on PSO algorithm. In: Proc. of the Fourth International Conf. on Computer Sciences and Convergence Information Technology, ICCIT '09. pp. 442 –446 (Nov 2009)

Computational System Analysis and Detection of Diabetes Mellitus

Marina I. Ramos-Martínez and Raúl Santiago Montero

Instituto Tecnológico de León,
Av. Tecnológico S/N, 37000, León, Gto., Mexico
{rsantiago66@gmail.com}
http://www.posgrado.itlleon.edu.mx
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. *Diabetes mellitus type 2 (DM2) is the leading cause of death in Mexico and is characterized by hyperglycemia (high glucose levels in the blood). Due to the high cost means to control DM2 in patients containing the disease, is created a computer system for early detection, which uses a pattern recognition method (KNN) to make a diagnosis on admission of a new data patient. The computer system can determine a diagnosis of a new patient with a faster way, plus it helps keep a more organized and easy access to information of each individual.*

Keywords: Pattern analysis, Pattern classification, KNN, Diabetes Mellitus TII

1 Introduction

The quality of health care defined by Donabedian as "the degree to which the most desirable means used to achieve the highest possible improvements in health." To ensure quality, there must be two inseparable elements, namely the system design and performance monitoring. Preventive medicine units should consider using a computer system, the result is to help in the capture and retrieval of patient information [1].

Measure and report the health of a population is crucial for anyone concerned about providing quality services to the population. A surveillance system allowing for timely information that facilitates making decisions or make recommendations for short, medium or long term, objective and scientific bases for the purpose of preventing or controlling health problems like diabetes mellitus type 2 (DM2), known for its high impact on health services utilization.

DM2, particularly when not controlled, can represent a heavy economic burden for the individual and society. Thus, depending on the country, estimates suggest that diabetes may represent between 5 and 14 % of health care expenditure to control the disease. [1] In Mexico, the DM2 is the leading cause of death in Fig. 1 can display the global position he occupied Mexico in the years studied. This is because people do not have a previous diagnosis or information sufficient to prevent it. The Guanajuato state ranks third in prevalence to diabetes at the national level (Fig. 2). [3]

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 171-176

Fig. 1. Position of Diabetes Mortality in Mexico. [2]

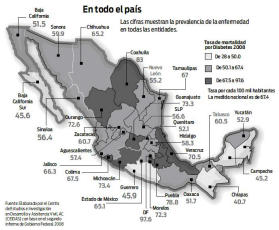


Fig. 2. Prevalence of Diabetes in Mexico. [5]

We developed a computer system capable of making a timely diagnosis of type 2 diabetes using an artificial intelligence technique (KNN) on databases of diabetic and nondiabetic patients. With the implementation of KNN, we can determine whether a new patient has diabetes mellitus or not.

The epidemic of diabetes mellitus, is the leading cause of death in Mexico, with an upward trend for three years to add more than 60 000 deaths and 400 000 new cases annually, with a greater number of deaths among women.

Significantly, the World Health Organization (WHO) has recognized this disease as a global threat, since it is estimated that more than 180 million people with diabetes worldwide, with the likelihood that this figure will increase to more double by 2030. [3]

2 Methodology

The new patient data are captured, including: File No., age, sex, body mass index, waist circumference, history of hypertension and diabetes mellitus, fasting glucose, systolic and diastolic blood pressure. The captured data representing the points of a vector. The system takes the values of 4 records in the database, two records of diabetic patients and two non-diabetic patients. Taking into account the physiological and biological characteristics of man and woman are different, if the new patient is female will be drawn only records of female patients and male otherwise. These data also represent the points of a vector. The system then calculates the distance between the new vector and the other 4 vectors by KNN. The shortest distance is chosen, if it is closest to the vector with DM2 positive, then the system concludes that the new patient is diabetic, otherwise it is not diabetic.

The KNN is described in the following steps:

1. First, each pattern in the training set is classified using k neighbors of the other training patterns set.
2. If the classification obtained is different from the original, the model is excluded from the training set. Thus, a new, smaller set of training is obtained.
3. The test patterns are classified using the 1-NN rule and the new training set derived in step 2. [4]

The criteria for defining whether a subject has diabetes are:

1. Fasting capillary glucose > 126 mg / ml or blood glucose levels at any time of day > 200 mg / ml.
2. Systolic blood pressure > 140mmHg or diastolic pressure > 90mmHg.
3. Body mass index > 30 kg/m2, obese, (25 to 29.9 kg/m2 overweight), ($\leqslant$ 24.9 kg/m2, normal weight).
4. The abdominal obesity was identified as the waist circumference of men was greater than 102 cm and women greater than 88 cm. [2]

3 Results

In Tables 1, 2 and 3 use the following nomenclature to describe the characteristics used in the system to determine if a new patient is diabetic or not.

- Sex (1 for women, 2 men's) 1 female Indeed, 2 male
- dm = (history of diabetes mellitus 1 = yes, 2 = no
- ht = (A history of hypertension, 1 = yes, 2 = no)
- gld = (blood glucose
- sis = systolic pressure
- dias = diastolic pressure
- imc = body mass index
- abd = waist circumference

The system resulted in the same diagnosis that the doctor previously performed, so that we can say that the system is effective. The databases used by the KNN are shown in Table 1 and 2.

Table 1 and 2 shows the databases previously assessed by the doctor that were used by the KNN to determine the diagnosis of diabetes in new patients.

Table 3 shows the database with information of new clients with which the system was tested, which were also previously evaluated by the doctor to perform the comparison of results, both the doctor and the system.

The system resulted in the same diagnosis that the doctor previously performed, so that we can say that the system is effective.

Table 1. Database nondiabetic

N°	Edad	sexo	dm	ht	imc	abd	sis	dias	gld
1	17	1	2	2	22.5	75	100	65	84
2	18	1	2	1	20	75	97	60	85
3	19	2	2	2	20	84	110	64	80
4	18	2	2	2	24	85	117	95	91
5	19	1	2	2	22	70	94	64	111
6	18	2	2	2	30.5	98	136	80	120
7	19	1	2	2	19.6	77	107	58	81
8	19	2	1	2	23	87	111	72	96
9	18	1	2	2	17.6	102	64	71	94
10	18	1	2	2	19.5	75	93	60	85
11	18	2	1	2	24.1	89	108	58	106
12	19	1	1	1	23.2	84	101	69	91
13	19	2	1	1	21.2	93	102	56	80
14	18	1	1	1	18	71	97	55	117
15	18	2	1	1	22	83	117	57	80
16	18	2	2	2	24	90	107	60	92
19	19	2	2	2	24.3	85	124	60	95
20	18	2	1	2	23	89	124	69	101

Table 2. Database diabetic patients

Edad	sexo	imc	abd	dm	ht	sis	dias	gld
51	1	42.18	95	2	2	120	80	170
36	1	32	90	2	1	110	70	140
36	1	30.11	80	2	2	130	90	135
42	2	31.2	96	2	2	120	70	146
77	2	28	85	2	2	140	80	154

Table 3. Database Tests and results

N°	Age	sex	dm	ht	imc	abd	sis	dias	gld	Medical Diagnosis	Diagnosis System
1	18	1	2	2	19	67	116	86	91	No diabético	No diabético
2	19	1	2	2	21	76	102	68	76	No diabético	No diabético
3	19	1	2	2	29	91	83	60	86	No diabético	No diabético
4	19	1	2	2	22	70	93	54	80	No diabético	No diabético
5	17	2	2	2	27	91	110	69	80	No diabético	No diabético
6	17	1	2	2	26	87	100	71	78	No diabético	No diabético
7	19	2	2	2	22	74	106	53	81	No diabético	No diabético
8	18	2	2	2	26	87	116	62	79	No diabético	No diabético
9	18	1	2	2	23	76	96	63	89	No diabético	No diabético
10	18	1	2	2	24	87	110	70	93	No diabético	No diabético
11	19	1	2	2	22	81	112	63	87	No diabético	No diabético
12	19	1	1	2	26	88	101	57	81	No diabético	No diabético
13	19	2	2	1	27	89	107	65	75	No diabético	No diabético
14	19	2	2	2	19	73	106	57	86	No diabético	No diabético
15	18	1	2	2	19	73	99	59	97	No diabético	No diabético
16	18	1	2	1	25	93	108	68	83	No diabético	No diabético
17	18	1	1	2	20	71	107	84	106	No diabético	No diabético
18	18	1	2	2	21	71	83	48	114	No diabético	No diabético
19	17	1	2	2	21.5	73	103	64	80	No diabético	No diabético
20	18	2	1	1	27	90	128	61	84	No diabético	No diabético
21	32	2	1	2	40.7	82	123	79	324	Diabético	Diabético
22	45	1	2	2	30.2	77	89	80	234	Diabético	Diabético
23	24	1	1	1	38	103	113	93	200	Diabético	Diabético
24	29	1	2	1	37.2	81	117	90	115	Diabético	Diabético
25	18	1	2	2	29	87	111	80	194	Diabético	Diabético
26	34	1	1	2	46	82	99	94	113	Diabético	Diabético
27	56	2	1	2	33.3	87	101	88	187	Diabético	Diabético
28	89	2	1	1	41.9	83	91	99	169	Diabético	Diabético
29	34	1	2	2	35	78	87	87	192	Diabético	Diabético
30	23	1	1	2	41	74	116	74	178	Diabético	Diabético
31	67	1	1	2	50	93	149	75	182	Diabético	Diabético
32	45	1	2	2	32.5	70	137	86	128	Diabético	Diabético
33	67	1	1	2	47	85	154	91	190	Diabético	Diabético
34	56	1	2	1	35	84	132	96	120	Diabético	Diabético
35	37	1	1	2	49	98	109	84	177	Diabético	Diabético

4 Conclusion

The proposed computer system allow the availability of reliable and permanent, and to store the data of each patient and have quick access to them. The system helps the different characters of institutional health care team providing information useful to have a positive impact on the pharmacological control of the disease, prevent acute complications and late complications delay. 35 tests were made which had a result equal to diagnosis by the doctor, so you can say that the AI technique (KNN) used is very efficient. In the medical field rules established are used for diagnosis. AI can help facilitate the work to create programs that are fed by these rules, to make faster diagnoses. We plan to continue forward with this project. Once the first phase will collect more patient data to bring the system to test a larger population.

Acknowledgment

This work was, in part, supported by the CONACyT (Master scholarship 365927) and the Instituto Tecnológico de León. The authors would like to thank... more thanks here

References

1. Donabedian, A.; Garantía y monitoria de la calidad de la atención médica; Cuernavaca, Morelos: Instituto Nacional de Salud Pública (1992);(Perspectivas en Salud Publica; p.p.9-12.
2. World Health Organization Department of Non-communicable Disease Surveillance; Definition, Diagnosis and Classification of Diabetes Mellitus and its Complications, Report of a Who Consultation, Part 1: Diagnosis and Classification of Diabetes Mellitus; World Health Organization, Geneva, (1999); 59 p.p.
3. Mejia, José Gerardo; Ssa presenta Norma Oficial contra diabetes mellitus; EL UNIVERSAL, México, D.F. 20 de octubre del 2009.
4. Marques de Sá, J.P., Pattern Recognition, Springer, p.p. 113-116.
5. Fuentes, Mario; La cuestión social en México. Nacional: 27. 11 de noviembre del 2008. Ceidas

Increasing the Reliability of a Network via the Number of Edge Covers

Guillermo De Ita, Yolanda Moyao, Meliza Contreras, Pedro Bello

Faculty of Computer Science, Universidad Autónoma de Puebla
{deita,ymoyao,mcontreras,pbello}@cs.buap.mx
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. Counting the number of edge covers on graphs, problem denoted as #Edge_Covers, is a #P-complete problem. Knowing the number of edge covers is useful for estimating the relevance of the lines in a communication network, which is an important measure in the reliability analysis of a network. In this paper, we propose a method for increasing the degree of reliability of a network for adding strategely new lines and then to increase the density of edge cover sets of the network. Regarding to the most common physical topologies of a network; Bus, Stars, Trees and Rings, we present efficient algorithms for solving the #Edge_Covers problem for these topologies.

Keywords: Edge Covers problem; communication network; reliability analysis;

1. Introduction

A computer network can be modeled via an undirected graph $G = (V, E)$ where the vertices of the graph V represent sites and the edges of the graph E stand for the links between the sites. There are many types of networks varying in their performance, definitions and therefore with different concepts of reliability [3].

In practice each site or link can fail, if an element (either node or edge) fails, we say that it is *down*, otherwise we say that it is *up*. The problem of checking the connectivity of the network is known to be NP-hard [1, 7]. In our work, we assume that sites (vertices) are perfect, but links may independently fail with similar known probabilities.

We show how to calculate the number of edge cover sets of the network in order to determine a degree of reliability of the network as well as for computing the relevance of the lines which is an important estimation of the 'strategic' value of each line in a network.

Given an initial network $N = (V, E)$ as an undirected Graph, we define the reliability of N in terms of the number of edge covers that N has. And we present, how to estimate the reliability of N when new communication lines are aggregated to N. To add a new line to a network N allow us to increase the *connectivity density* of the original network. Then, our procedure permit us to search for the best two nodes for adding a new line to N and maximizes so the *connectivity density* of N.

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 179-188

Our proposal is based on counting the number of edge covers of a graph (the #Edge_Covers problem) [5, 6], and for solving this late problem, we have developed a novel method for counting edge covers of a graph for the most common topologies of a network; bus, stars, trees and rings. We show that the #Edge_Covers problem is solved in linear time on the size (number of nodes plus number of edges) of any graph without intersecting cycles (any pair of cycles with common edges).

1.1. Preliminaries

We model a communication network by an undirected graph $G = (V, E)$ (i.e. finite, loopless and without parallel edges), where V represents the set of vertices (also known as nodes) and E represents the set of edges. Sometimes $V(G)$ and $E(G)$ are utilized to emphasize the graph G. A vertex and an incident edge are said to *cover* each other.

The neighborhood of a vertex $v \in V$ is the set $N(v) = \{w \in V : \{v, w\} \in E\}$ and its degree, denoted as $\delta(v)$, is the number of neighbors that it has. The cardinality of a set A will be denoted as $|A|$. Thus, $\delta(v) = |N(v)|$, and we say that a vertex v is *pendant* if its neighborhood contains only one vertex; an edge e is *pendant* if one of its endpoints is a pendant vertex [2]. The degree of the graph G is $\Delta(G) = max\{\delta(x) : x \in V\}$.

Given a graph $G = (V, E)$, $S = (V', E')$ is a subgraph of G if $V' \subseteq V$ and E' contains edges $\{v, w\} \in E$ such that $v, w \in V'$. If E' contains every edge $\{v, w\} \in E$ where $v, w \in V'$ then S is called the *subgraph of G induced by S* and is denoted by $G\|S$. $G - S$ denotes the graph $G\|(V - V')$. $G - v$ for any $v \in V$ denotes the induced subgraph $G\|(V - \{v\})$, and $G - e$ for any $e \in E$ denotes the subgraph of G with the set of nodes V and set of edges $E - \{e\}$.

A *connected component* of G is a maximal induced subgraph of G, that is, a connected component is not a proper subgraph of any other connected subgraph of G. Notice that, in a connected component, for every pair of its vertices u, v, there is a path from u to v. A tree graph is an acyclic connected graph.

An edge cover for a graph $G = (V, E)$ is a subset of edges $\mathcal{E} \subseteq E$ that covers all nodes of G, that is, for each $u \in V$ there is a $v \in V$ such that $e = \{u, v\} \in \mathcal{E}$. Let $C\mathcal{E}(G) = \{\mathcal{E} \subseteq E : \mathcal{E}$ is an edge cover of $G\}$ be the set of edge covers that G has. Let $NE(G) = |C\mathcal{E}(G)|$ be the number of edge covers of G. Computing the number $NE(G)$ for any input graph is known as the #Edge_Covers problem.

The #Twice-SAT problem consists in counting the number of models of a Boolean formula F where variable of F appears at most two times (with any one of its two signs) in F. #Edge_Covers is a #P-complete problem which has been proved via the reduction from #Twice-SAT to #Edge_Covers [2, 8].

2. Estimating the Relevance on Communication Lines

Complex networks, modeled as large graphs, have received much attention during the last years. However, topological information on these networks is only

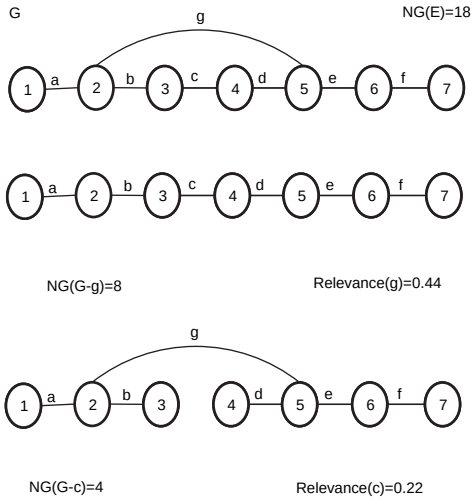


Fig. 1. Estimating the relevance of lines in a network

available through intricate measurement procedures [1]. Nodes and edges of a network could fail (become non-operational). If an element (either node or edge) fails, we say that it is *down*, otherwise we say it is *up*.

If we assume that the communication lines (edges) in a network G have the same failure probability and those failures are independent from each other, whenever an edge $e \in G$ fails we can estimate different classes of reliability of the network. For example, the 'relevance' of a line e in a network G can be estimated by the conditional probability $P_{e/G}$ which can be approximated by the fraction of the number of edge covers which are substracted when the edge e is removed (fails), that is, $P_{e/G} = 1 - \frac{NE(G-e)}{NE(G)}$.

Thus, $P_{e/G}$ gives us the strategic value of an edge e in a network G, i.e. for greater values of $P_{e/G}$ the line e is most relevant than the other lines in G, in order to maintain the connectivity of G in case of failures. $P_{e/G}$ is an proportion which indicates the density of edge covers of G with respect to the line e.

In general, as e is any edge of G then $P_{e/G}$ could be used for estimating the relevance of any edge of G, such as it is shown in Table 1. For example in figure 1, we show the effect over the number of edge covers of the network G when lines g or c of the network G fail. Thus, we are estimating the 'density' of each edge for covering all node of G when some edges fail.

In the table 1, we assume that one of the lines of the set $\{b, c, d, e, f\}$ down, and the the effect of that fails over the reliability of the network of the figure 1. So, in the table 1 we show the relevance of each one of those lines.

Given this measure of reliability and given a network $N = (V, E)$, a relevant practical problem is to determine two non-adjacent nodes in V for connecting such nodes via a new communication line for N and then to increase the density

Table 1. Relevance of each edge of G

Edges	$NE(G-e)$	$P_{e/G}$
g	8	.44
c	4	.22
b	7	.61
d	5	.72
e	7	.61

of edge covers of N. How to select such two nodes, is the objective of the following section.

2.1. Increasing the Density of the Edge Covers of a Networks

Assuming a network $N = (V, E)$, $n = |V|$ and $m = |E|$, there are $n*(n-1)-|E|$ options for selecting two non-adjacent nodes of N. If we know the value $NE(N)$ and adding to E a new line e then the value $NE(N \cup e)$ is a measure about how increase the number of edge covers in N.

We want to select the two non-adjacent nodes such that the edge e between such node permit us to maximize the value $NE(N \cup e)$ into the set of all possible new edges of N. And for this, we need first how to compute $NE(N \cup e)$ for any new edge e.

Let $e = \{u, v\} \notin E$ and $u, v \in V$. e is a new possible edge for N and we want to know the difference $NE(N \cup e) - NE(N)$. When an edge cover $\mathcal{E}$ of G is being built, we distinguish between two different states of a node u; we say that u is *free* when it has not still been covered by any edge of $\mathcal{E}$, otherwise the node is *covered*.

In order to compute $NE(N \cup e)$ we conform a new network $N_1 = (V_1, E_1)$ obtained from $N = (V, E)$ in the following way: $V_1 = V - \{u, v\}$ plus new virtual covered node: u_x for each edge in $E - e$ type $\{x, u\}$, and new virtual covered node: v_x for each incident edge type $\{x, v\}$. When we compute $NE(N)$ we will consider that the new virtual nodes u_x and v_x are covered nodes, that means, any adyacent edge to them could (or not) appear into the set of edge covers of N.

In order to form E_1, the edge e is deleted from $E(N)$ and for each edge type $e_j = \{x, u\} \in E(N)$ a new pendant edge $e'_j = \{x, u_j\}$ appears in E_1, or when $e_j = \{x, v\} \in E(N)$ it changes in E_1 as $e'_j = \{x, v_j\}$, u_j and v_j being one of the new covered nodes. Such edges e'_jś are pendant edges in N_1 as well as the new covered nodes u_x and v_x are pendant nodes, see figure 2. Furthermore, we can show that $NE(N \cup e) = NE(N_1)$.

Notice that in this case, all cycle of N containing any one of the original nodes u and v is not more a cycle in N_1. Notice also that the main problem here, consists in count the number of edge cover sets for different class of physical topologies networks. In the following chapter, we present linear time exact procedures for computing the number of edge cover sets of a network, for the most common physical topologies [9]. The degree of the graph is irrelevant in our methods.

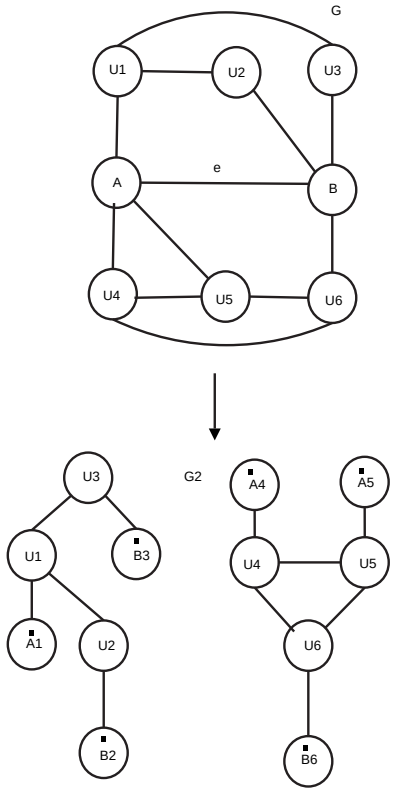


Fig. 2. Estimating the number of edge covers for adding a new line (e)

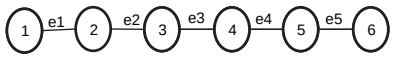
3. Linear time Procedures for Counting Edge Covers

$NE(G)$ for any graph G, including the case when G is a disconnected graph, is computed as: $NE(G) = \prod_{i=1}^{k} NE(G_i)$, where k is the cardinality of the set of connected components of G and each G_i represents an element of this set. The set of connected components of G can be computed in linear time [1].

The edges of G appearing in all edge cover sets are called *fixed edges*. We start designing procedures for counting edge covers, considering the most common topologies of a network.

3.1. Case A: The Bus Topology

Let $P_n = G = (V, E)$ be a linear bus (a path graph). We assume an order between vertices and edges in P_n, i.e. let $V = \{v_0, v_1, \ldots, v_n\}$ be the set of $n+1$ vertices and let $e_i = \{v_{i-1}, v_i\}$, $1 \leq i \leq n$ be the n edges of P_n.



$e_1 \quad e_2 \quad e_3 \quad e_4 \quad e_5$

$(1,0) \to (1,1) \to (2,1) \to (3,2) \mapsto (5,0) = 5$

Fig. 3. Counting edge covers on a bus

Let $G_i = (V_i, E_i)$, $i = 0, \ldots, n$ be the subgraphs induced by the first i nodes of V, i.e. $G_0 = (\{v_0\}, \emptyset), G_1 = (\{v_0, v_1\}, \{e_1\}), \ldots, G_n = P_n$. $G_i, i = 0, \ldots, n$ is the family of induced subgraphs of G formed by the first i nodes of V. Let $\mathcal{CE}(G_i)$ be the set of edge covers of each subgraph G_i, $i = 0, \ldots, n$.

Each edge $e_i, i = 1, \ldots, n$ in the bus has associated an ordered pair (α_i, β_i) of integer numbers where α_i carries the number of edge cover sets of $\mathcal{CE}(G_i)$ where the edge e_i appears in order to cover the node v_{i-1}, while β_i conveys the number of edge cover sets in $\mathcal{CE}(G_i)$ where the edge e_i does not appear.

By traversing P_n in depth-first search [1], each pair $(\alpha_i, \beta_i), i = 1, \ldots, n$ is computed in accordance with the type of edge that e_i is. P_n has two fixed edges: e_1 and e_n. The pair (1,0) is assigned to (α_1, β_1) because e_1 has to appear in all edge cover of P_n.

If we know the values $(\alpha_{i-1}, \beta_{i-1})$ for any $0 < i < n$, then we know the number of times where the edge e_{i-1} appears or does not appear into the set of edge covers of G_i. When the edge e_i is being visited, the vertex v_{i-1} has to be covered considering its two incident edges: e_{i-1} and e_i. Any edge cover of $\mathcal{CE}(G_i)$ containing the edge e_{i-1} (α_{i-1} cases) has already covered v_{i-1} then the ocurrence of e_i is optional. But for the edge covers where e_{i-1} does not appear (β_{i-1} cases) e_i must appear in order to cover v_{i-1}. This simple analysis shows that the number of edge covers where e_i appears is $\alpha_{i-1} + \beta_{i-1}$ and that just in α_{i-1} edge covers the edge e_i does not appear. Thus, we compute (α_i, β_i) associated with the edge e_i, applying the Fibonacci recurrence relation.

$$\alpha_i = \alpha_{i-1} + \beta_{i-1}; \quad \beta_i = \alpha_{i-1} \tag{1}$$

When the search arrives to the last edge e_n of the linear bus, we have already computed the pair $(\alpha_{n-1}, \beta_{n-1})$; since e_n is a fixed edge, it has to appear in all edge covers of P_n. We call $\alpha_n = \alpha_{n-1} + \beta_{n-1}$ and $\beta_n = 0$ *the recurrence for processing fixed edges* (RPFE).

The pair associated with e_n is $(\alpha_n, \beta_n) = (\alpha_{n-1} + \beta_{n-1}, 0)$. The sum of the elements of this pair (α_n, β_n) yields the number of edge covers: $NE(P_n) = \alpha_n + \beta_n$. Notice that $NE(P_n)$ is computed in linear time over the number of edges in P_n. In figure 3 we present an example where $\to$ denotes the application of recurrence (1), and $\mapsto$ denotes the application of RPFE.

Recall that each Fibonacci number F_i can be bounded from above and from below by $\phi^{i-2} \geq F_i \geq \phi^{i-1}, i \geq 1$, where $\phi = \frac{1}{2} \cdot (1 + \sqrt(5))$.

Theorem 1 *The number of edge cover sets of a path of n edges, is:*
$F_n = \texttt{ClosestInteger}\left[\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n\right]$.

proof: The series $(\alpha_i, \beta_i), i = 1, \ldots, n$ used for computing $NE(P_n)$, coincides with the Fibonacci numbers: $(F_1, F_0) \rightarrow (F_2, F_1) \rightarrow (F_3, F_2) \rightarrow \ldots \rightarrow (F_{n-1}, F_{n-2}) \mapsto (F_n, 0)$. Then, we infer that $(\alpha_i, \beta_i) = (F_i, F_{i-1})$ for $i = 1, \ldots, n-1$ and $\alpha_n = F_n, \beta_n = 0$. Thus, $NE(P_n) = \alpha_n + \beta_n = F_n$.

3.2. Case B: The Tree Topology

Let $T = (V, E)$ be a rooted tree. *Root-edges* in T are the edges with one endpoint in the root node; *leaf-edges* in T are the edges with one endpoint in a leaf node of T.

Given any intermediate node $v \in V$, we call a *child-edge* of v to the edge connecting v with any of its children nodes, and the edge connecting v with its father node is called the *father-edge* of v. $NE(T)$ is computed by traversing T in post-oder and associating (α_e, β_e) with each edge $e \in E$, except for the leaf edges.

Theorem 2 *If T is a tree without covered nodes and T' is the leaves-pruned tree of T with leaves labelled as covered nodes then $NE(T) = NE(T')$.*

proof: Suppose the tree T has leaf-edges $e_1, e_2, \ldots, e_k$. Since each leaf-edge covers a leaf-node of T and there is not other edge which covers such a leafnode then, every leaf-edge of T should be included in each set cover for T. Let $CE(T) = \{A_0 \cup \{e_1, e_2, \ldots, e_k\}, A_1 \cup \{e_1, e_2, \ldots, e_k\}, ..., A_n \cup \{e_1, e_2, \ldots, e_k\}\}$ be the set of edge covers for T where neither of $A_0, A_1, \ldots, A_n$ contains elements of the set $\{e_1, e_2, \ldots, e_k\}$. By removing from each element of the set $CE(T)$ the following set $e_1, e_2, \ldots, e_k$ we end up with $L = \{A_0, A_1, \ldots, A_n\}$, whose cardinality obviously coincides with the cardinality of $CE(T)$, that is $|CE(T)| = |L|$. It will be shown that $|L| = |CE(T')|$. It is obvious that each element of L is an edge cover for T' since each element of L covers all the vertexes of T, except the leaves. Now, suppose that A_i is a cover for T' then $A_i \cup \{e_1, e_2, ..., e_k\}$ should cover T for every $i = 1, \ldots, k$. Hence $|CE(T')| = |CE(T)| = |L|$ or $NE(T) = NE(T')$, which concludes the proof.

Procedure for computing #Edge Covers for Trees(T)

1. Reduce the input tree T to other tree T' by prunning all leaf nodes and leaf-edges from T, and by labeling as *covered* all father nodes of the original leaf nodes of T (see figure 4).
2. Traverse T' in post-order and associate a pair (α_e, β_e) with each edge e in T'. Such pairs are computed in the following way:
   (a) $(\alpha_e, \beta_e) = (1, 1)$ if e is a leaf-edge of T', since its children nodes have been covered.
   (b) if an internal node v is being visited and it has a set of child-edges, e.g. $u_1, u_2, ..., u_k$, as we have already visited all child-edges of v, then each pair $(\alpha_{u_j}, \beta_{u_j})$, $j = 1, \ldots, k$ has been computed. Assume α_u carries the number of different combinations of the child-edges of v for covering v,

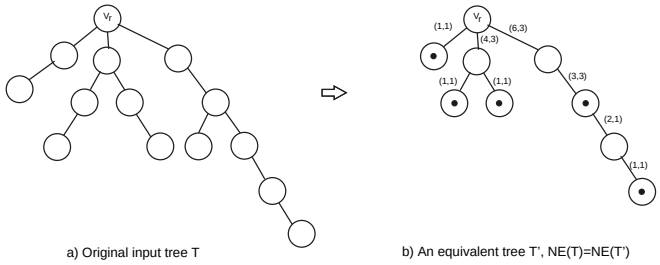


Fig. 4. Computing the number of edge covers for a tree

while β_u gives the number of combinations among the child-edges of v which do not cover v. The pair (α_u, β_u), which we assume represents an imaginary child-edge e_u of v, is computed as:

$$\alpha_u = \prod_{j=1}^{k}(\alpha_{u_j} + \beta_{u_j}) - \prod_{j=1}^{k}\beta_{u_j} \; ; \quad \beta_u = \prod_{j=1}^{k}\beta_{u_j} \tag{2}$$

The pair associated to the father-edge e_v of v is computed as:

$(\alpha_v, \beta_v) =$

$\begin{cases} (\alpha_u + \beta_u, \alpha_u) & \text{if } v \text{ is a free node or,} \\ (\alpha_u + \beta_u, \alpha_u + \beta_u) & \text{if } v \text{ is a covered node} \end{cases}$

This step is iterated until it computes the pairs (α_e, β_e) for all edge $e \in T'$. If there are more than one root-edges then one iteration more of this step is applied in order to obtain a final pair $(\alpha_{e_r}, \beta_{e_r})$ associated with just one root-edge e_r.

3. $NE(T)$ is computed in accordance with the status of the root node v_r of T; $NE(T) = \alpha_{e_r} + \beta_{e_r}$ if v_r is a covered node, otherwise $NE(T) = \alpha_{e_r}$.

The above procedure returns $NE(T)$ in time $O(n + m)$ which is the necessary time for traversing T in post-order. Notice that this case includes the star topology network.

Example 1 *Let T be the tree of figure 4a. T' is the reduced tree from T where its covered nodes are marked by a black point inside of the nodes (figure 4b). When T' is traversed in post-order a pair (α_e, β_e) is associated with each edge. The pairs for the child-edges of v_r, are: (1,1), (4,3) and (6,3). Those three edges are combined in only one edge e_r applying recurrence (2): $\alpha_{e_r} = (1+1)*(4+3)*(6+3) - 1*3*3 = 117$ and $\beta_{e_r} = 1*3*3 = 9$. Since v_r is the root node and it is free, then $NE(T) = \alpha_{e_r} = 117$*

3.3. Case C: The Ring Topology

Let $C_n = (V, E)$ be a simple ring with n edges. We assume an order over the nodes and edges of C_n given by $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_n\}, e_i =$

$\{v_i, v_{i+1}\}, i = 1, \ldots, n-1$, $e_n = \{v_n, v_1\}$. We call a *computing thread* or just a *thread* to the series $(\alpha_1, \beta_1) \rightarrow (\alpha_2, \beta_2) \rightarrow \cdots \rightarrow (\alpha_k, \beta_k)$ obtained for counting in incremental way, applying the recurrence (1), the number of edge covers of a path with k edges.

Let L_p be the thread used for computing the series of pairs associated to the n edges of C_n. $(\alpha_1, \beta_1) = (1, 1)$ is associated with e_1 since C_n has not fixed edges. Traversing in depth first search, the new pairs in L_p are computed applying the Fibonacci recurrence (1) since all nodes in C_n have degree two and they are free. After n applications of recurrence (1), the pair $(\alpha_n, \beta_n) = (F_{n+1}, F_n)$ is obtained, F_i being the i-th Fibonacci number.

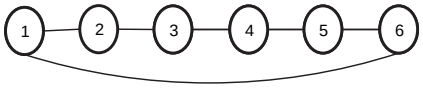
Let NC_n be the number of edge sets counted by L_p, i.e. $NC_n = \alpha_n + \beta_n = F_{n+2}$. L_p counted the edge sets where neither e_1 nor e_n appear, since $\beta_1 = 1$ and $\beta_n > 0$. Due to e_1 or e_n or both have to be included in the edge cover sets of C_n in order to cover v_1, we have to substract from NC_n the number of sets which not cover v_1.

Let Y be the number of edge sets which cover all nodes of C_n except v_1, then $NE(C_n) = NC_n - Y$. In order to compute Y a new thread $L'_p = (\alpha'_1, \beta'_1) \rightarrow \cdots \rightarrow (\alpha'_n, \beta'_n)$ is computed. L'_p begins with the pair $(\alpha'_1, \beta'_1) = (0, 1)$, i.e. it begins counting the edge sets where e_1 does not appear. After n applications of recurrence (1) the last pair (α'_n, β'_n) of L'_p obtains (F_{n-1}, F_{n-2}).

The number of edge sets where neither e_1 nor e_n appear is $\beta'_n = F_{n-2}$, hence $Y = F_{n-2}$. Finally, $NE(C_n) = NC_n - Y = F_{n+2} - F_{n-2}$. Then, we deduce the following theorem.

Theorem 3 *The number of edge cover sets of a simple cycle C_n with n edges, expressed in terms of Fibonacci numbers, is: $NE(C_n) = F_{n+2} - F_{n-2}$.*

With ' $\curvearrowleft$ ' we denote the binary operation $(\alpha_n, \beta_n - \beta'_n)$ between two pairs, and the result is assocciated with the last edge e_n of the ring C_n (fig. 4). Notice that the computation of $NE(C_n)$ is the order $O(n)$ since we compute the two threads: Lp and L'_p in parallel while the depth-first search is applied.



$$
\begin{array}{rl}
 & (\alpha_1, \beta_1) \rightarrow (\alpha_2, \beta_2) \rightarrow (\alpha_3, \beta_3) \rightarrow (\alpha_4, \beta_4) \rightarrow (\alpha_5, \beta_5) \rightarrow (\alpha_6, \beta_6) \\
Lp: & (1,1) \rightarrow (2,1) \rightarrow (3,2) \rightarrow (5,3) \rightarrow (8,5) \rightarrow (13,8) \\
Lp': & (0,1) \rightarrow (1,0) \rightarrow (1,1) \rightarrow (2,1) \rightarrow (3,2) \rightarrow (5,3) \\
 & \Rightarrow (13,8) \curvearrowleft (5,3) = (13,5)
\end{array}
$$

Fig. 5. Counting the number of edges covers for a ring

Example 2 *Let C_6 be the ring illustrated in figure 5. Applying the above theorem, we have that $NE(C_6) = F_{6+2} - F_{6-2} = F_8 - F_4 = 21 - 3 = 18$.*

The graphs which hold the topologies of the above cases (A) to (C) englobe the most common topologies of a communication network. The linear time procedures designed here can be included into a branch and bound algorithm which processes any kind of topology of a network [4].

4. Conclusion

We have determined different recurrence relations for counting the number of edge cover sets for different physical topologies of a network. If the topology of a network G is a bus, a star, a tree or a ring, then the number of edge covers of G can be computed in linear time on the size (number of nodes plus number of edges) of the network.

Knowing the number of edge covers of a network is helpful for estimating its reliability. For example, we have shown how to estimate the 'relevance' of any line e of the network based on the proportion of the number of edge covers where e does not appear with respect to the total number of edge covers in the network. Such proportion is an indicative of the density of edge covers of the newtwork with respect to the line e.

Furthermore, we present a method for selecting the best two non adjacent nodes on the network which allow us maximize the density of edge covers sets, adding a new line connecting such nodes in the network.

References

1. D. J. M. Garey., Computers and Intractability a Guide to the Theory of NP-Completeness, (1979).
2. M. D. R. Bubley., Graph orientations with no sink and an approximation for a hard case of #sat., *Proc. of the Eight Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 248–257, 1997.
3. Y. Shpungin., Combinatorial approach to reliability evaluation of network with unreliable nodes and unreliable edges., *PInt. Jour. of Computer Science Vol. 1*, 1(3):177–191, 2006.
4. R. Tarjan., Depth-First Search and Linear Graph Algorithms., *SIAM Journal on Computing*, 1:146–160, 1972.
5. S. P. Vadhan., The complexity of counting in sparse, regular, and planar graphs., *SIAM Journal on Computing*, 31(2):398–427, 2001.
6. E. M. V.E. Levit., The independence polynomial of a graph - a survey., *Holon Academic Inst. of Technology*, 2005.
7. B. Russ.,*Randomized Algorithms: Approximation, Generation, and Counting, Distinguished dissertations.*, 2001.
8. L. Kou, C. Stockmeyer, C. Wong., Covering edges by cliques with regard to keyword conflicts and intersection graphs., *Communications of the ACM*, pages 136–139, 1978.
9. G. De Ita, P. Bello, M. Contreras., Applying Fibonacci Recurrences for Counting Combinatorial Objects in Graphs., *Advances in Computer Science and Engineering Vol. 34*, :3–14, 2008.

Artificial Intelligence planning with p-stable semantics

Sergio Arzola[1], Claudia Zepeda[1], Mario Rossainz[1], and Mauricio Osorio[2]

[1]Benemérita Universidad Autónoma de Puebla,
Facultad de Ciencias de la Computación
[2]Universidad de las Américas, Puebla
sinrotulos@gmail.com,czepedac@gmail.com
rossainz@cs.buap.mx,osoriomauri@gmail.com
http://sites.google.com/site/gmlogyc
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. Our work is intended to model and solve artificial planning problems with logic based planning, using the novel semantics called *p-stable*, which is an alternative of stable semantics. It can be applied in a variety of tasks including robotics, process planning, updates, making evacuation plans and so on.

Keywords: planning, p-stable semantics, yale shooting problem.

1 Introduction

Currently, the Artificial Intelligence has more applications as well as has been taken relevance into processes and products of industries. One of the biggest branches of Artificial Intelligence is the study of planning, because resolution of planning problems has different applications in different areas too. For example, the robotics movement planning, creating evacuation plans, etcetera.
Planning in Artificial Intelligence is decision making about the actions to be taken.
Imagine an intelligent robot. The robot is a computational mechanism that takes input through its sensors and act with the effectors, which can be motors, lights, and so on. So the sensors allow the robot to perceive its environment and to build a representation of the world has perceived before as well as its immediate surroundings. Then the robot must act according to the representation of the world it has, that cames from its perception. The robot acts through its effectors which are devices that allow the robot to change the states of the environment interacting with it as changes the states of itself, like move from a place to another, move items, and so on. At an abstract level, a robot is a mechanism that maps its observations to actions which are obtained through sensors and performed by the effectors respectively. In this context. planning is the decision making, where gives a sequence of actions by a sequence of observations [13].
There are different approaches about planning done in different areas of artificial intelligence, however our focus here is into Logic-based Planning [7]. Furthermore, our proposal is using the p-stable semantics in order to model and solve

Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 189-197

planning problems.
The p-stable semantics is a novel semantics that came from an alternative for stable semantics [8]. There exists evidence about the applicably of p-stable in different domains. However, the purpose of this article is about the planning domain [14] [1] In this work, we are interested in two basic parts: the first one is the modeling in language $\mathcal{A}$, where it is intended to show how we can model easily a planning problem and the second one is how to translate it in a simple way from language $\mathcal{A}$ to p-stable semantics rules. In order to see what models we get, we use as resolver the last implementation of p-stable semantics [11].
This paper is structured as follows. In section 2 we introduce the general syntax of the logic programs used in this paper. We also provide the definition of stable and p-stable semantics. In section 3 we present the logic basic planning and the p-stable approach giving a basic example of code. Finally in section 4 we present the conclusions.

2 Background

In this section we summarize some basic concepts and definitions used to understand this paper.

2.1 Logic programs

A *signature* $\mathcal{L}$ is a finite set of elements that we call *atoms*, or *propositional symbols*. The language of a propositional logic has an alphabet consisting of *proposition symbols*: $p_0, p_1, \ldots$; *connectives*: $\wedge$, $\vee$, $\leftarrow$, $\neg$; and *auxiliary symbols*: (,). Where $\wedge$, $\vee$, $\leftarrow$ are 2-place connectives and $\neg$ is a 1-place connective. Formulas are built up as usual in logic. A *literal* is either an atom a, called *positive literal*; or the negation of an atom $\neg a$, called *negative literal*. The formula $F \equiv G$ is an abbreviation for $(F \leftarrow G) \wedge (G \leftarrow F)$. A *clause* is a formula of the form $H \leftarrow B$ (also written as $B \rightarrow H$), where H and B, arbitrary formulas in principle, are known as the *head* and *body* of the clause respectively. The body of a clause could be empty, in which case the clause is known as a *fact* and can be denoted just by: $H \leftarrow$. In the case when the head of a clause is empty, the clause is called a *constraint* and is denoted by: $\leftarrow B$. A *normal* clause is a clause of the form $H \leftarrow \mathcal{B}^+ \cup \neg\mathcal{B}^-$ where H consists of one atom, $\mathcal{B}^+$ is a conjunction of atoms $b_1 \wedge b_2 \wedge \ldots \wedge b_n$, and $\neg\mathcal{B}^-$ is a conjunction of negated atoms $\neg b_{n+1} \wedge \neg b_{n+2} \wedge \ldots \wedge \neg b_m$. $\mathcal{B}^+$, and $\mathcal{B}^-$ could be empty sets of atoms. A finite set of normal clauses P is a *normal program*.

Finally, we define $RED(P, M) = \{H \leftarrow B^+, \neg(B^- \cap M) \mid H \leftarrow B^+, \neg B^- \in P\}$. For any program P, the positive part of P, denoted by $POS(P)$ is the program consisting exclusively of those rules in P that do not have negated literals.

2.2 Stable and p-stable semantics

From now on, we assume that the reader is familiar with the notion of classical minimal model [10]. We give the definitions of the stable and p-stable semantics for normal programs.

Definition 1. *[12] Let P be a normal program and let $M \subseteq \mathcal{L}_P$. Let us put $P^M = POS(RED(P, M))$, then we say that M is a* stable model *of P if M is a minimal classical model of P^M.*

Definition 2. *[12] Let P be a normal program and M be a set of atoms. We say that M is a* p-stable model of P *if: (1) M is a classical model of P (i.e. a model in classical logic), and (2) the conjunction of the atoms in M is a logical consequence in classical logic of $RED(P, M)$ (denoted as $RED(P, M) \models M$).*

Example 1. Let P be the normal program $\{b \leftarrow \neg a, \quad a \leftarrow \neg b, \quad p \leftarrow \neg a \quad p \leftarrow \neg p\}$. We can verify that $M_1 = \{a, p\}$ and $M_2 = \{b, p\}$ model the rules of P. From the definition of the RED transformation we find $RED(P, M_1) = \{b \leftarrow \neg a, \quad a \leftarrow , \quad p \leftarrow \neg a, \quad p \leftarrow \neg p\}$, and $RED(P, M_2) = \{b \leftarrow, \quad a \leftarrow \neg b, \quad p \leftarrow, \quad p \leftarrow \neg p\}$. It is clear that $RED(P, M_1) \models M_1$ and $RED(P, M_2) \models M_2$. Hence M_1 and M_2 are *p-stable models* for P. It is easy to see that M_2 is stable model of P whereas M_1 is not.

The following theorem shows the relation between the stable and p-stable semantics for normal logic programs.

Theorem 1. *[12] Let P be a normal logic program and M be a set of atoms. If M is a stable model of P then M is a p-stable model of P.*

3 Planning based on p-stable semantics

In this section we present how we model planning into the p-stable semantics.

3.1 Logic-based Planning

In a planning problem, we are interested in looking for a sequence of actions that leads from a given initial state to a given goal state. There exist different action languages that are formal models used to model planning problems, such as $\mathcal{A}, \mathcal{B}$, or $\mathcal{C}$ [9]. A planning problem specified in one of these languages has a easy encoding in declarative logic languages based on p-stable semantics. In this Section we shall present a brief overview extracted from [4] about language $\mathcal{A}$, and the encoding of planning problems based on p-stable semantics.

3.2 Language $\mathcal{A}$

The alphabet of the language $\mathcal{A}$ consists of two nonempty disjoint sets of symbols **F** and **A**. They are called the set of fluents, and the set of actions. Intuitively, a fluent expresses the property of an object in a world, and forms part of the description of states of the world. A *fluent literal* is a fluent or a fluent preceded by $\sim$. A *state* σ is a set of fluents. We say a fluent f holds in a state σ if $f \in \sigma$. We say a fluent literal $\sim f$ holds in σ if $f \notin \sigma$. Actions when successfully executed change the state of the world. Situations are representations of the history of action execution. The situation $[a_n, \ldots, a_1]$ corresponds to the history where action a_1 is executed in the initial situation, followed by a_2, and so on until a_n. There is a simple relation between situations and states. In each situation s some fluents are true and some others are false, and this 'state of the world' is the state corresponding to the situation s.

The language $\mathcal{A}$ can be divided in three sub-languages: *Domain description language*, *Observation language*, and *Query language* [9,4].

Domain description language. It is used to express the transition between states due to actions. The domain description D consists of effect propositions of the following form: a **causes** f **if** $p_1, \ldots, p_n, \sim q_1, \ldots, \sim q_r$ where a is an action, $f, p_1, \ldots, p_n, q_1, \ldots, q_r$ are fluents. Intuitively, the above effect proposition means that if the fluent literals $p_1, \ldots, p_n, \sim q_1, \ldots, \sim q_r$ hold in the state corresponding to a situation s then in the state corresponding to the situation reached by executing a in s the fluent literal f must hold. The role of effect propositions is to define a transition function, Φ, from states and actions to states. The domain description part also can include *executability conditions*: **executable** a **if** $p_1, \ldots, p_n, \sim q_1, \ldots, \sim q_r$ where a is an action and, $p_1, \ldots, p_n, q_1, \ldots, q_r$ are fluents. Intuitively, it means that if the fluent literals $p_1, \ldots, p_n, \sim q_1, \ldots, \sim q_r$ hold in the state σ of a situation s, then the action a is executable in s.

Observation language. A set of observations O consists of value propositions of the form: **initially** f. Given a consistent domain description D the set of observations O is used to determine the states corresponding to the initial situation, referred to as *initial states* and denoted by σ_0.

Query language. We say a consistent domain description D in the presence of a set of observations O entails a query Q of the form f **after** $a_1, \ldots, a_m$ if for all initial states σ_0 corresponding to (D, O), the fluent literal f holds in the state $[a_m, \ldots, a_1]\sigma_0$. We denote this as $D \models_O Q$.

Hence, in order to model a planning problem using language $\mathcal{A}$, we must specify a triple (D, O, G) where D is a domain description, O is a set of observations, and G is a collection of fluent literals $G = \{g_1, \ldots, g_l\}$, which we will refer to as a goal. So, we require to find a sequence of actions $a_1, \ldots, a_n$ such that for all $1 \leq i \leq l$, $D \models_O g_i$ **after** $a_1, \ldots, a_n$. We then say that $a_1, \ldots, a_n$ is a *plan* for achieves goal G with respect to (D, O).

3.3 P-stable encoding of planning problems

We have described before, how to model planning problems using language $\mathcal{A}$. In this section we present a way to encode planning problems into p-stable semantics, since this semantics is new, there is no application made for planning purposes, but we can model the language $\mathcal{A}$ into rules of this semantics as following:

Vocabulary Fluents $f_1, \ldots, f_n$ can be declared by just making a rule of each fluent as a fact:
`fluent(`f_1`), ... , fluent(`f_n`).`
Similar as above, the actions $a_1, \ldots, a_n$ can be declared by presenting each action as a fact:
`action(`a_1`), ... , action(`a_n`).`

Encoding domain description. The propositions of the form: a **causes** f **if** $p_1, \ldots, p_n, \sim q_1, \ldots, \sim q_r$ can be declared by the following rule:
`holds(f, `T`+1)` $\leftarrow$ `occurs(a, T), holds(`$p_1, \ldots, p_n$`, T), not_holds(`$q_1, \ldots, q_r$`, T), time (T).`
Also prepositions of the following form:
executable a **if** $p_1, \ldots, p_n, \sim q_1, \ldots, \sim q_r$. can be declared by the following rule:
`occurs(a,T)` $\leftarrow$ `holds(`$p_1, \ldots, p_n$`, T), not_holds(`$q_1, \ldots, q_r$`, T), time (T).`
In order to encode the rules, we express them with holds and occurs, which means that the fluent is satisfied or the action occurs at time T respectively.

Encoding observation language. These prepositions represents the initial state of the problem. It can be declared by giving it every observation as a fact at time 0:
`holds(`f_a`,0), ..., holds(`f_m`,0).`
`not_holds(`f_n`,0), ..., not_holds(`f_z`,0).`

Encoding query language. These prepositions declare the goal, or the wished state at time N, it is represented by giving what state we want to have at time N by its fluents as follows:
`holds(`f_a`,N), ..., holds(`f_m`,N).`
`not_holds(`f_n`,N), ..., not_holds(`f_z`,N).`

In the following section we give a brief example of a planning problem modeled into language $\mathcal{A}$ and into p-stable semantics in order to clarify this.

3.4 The yale problem modeled and encoded

Here we present the yale shooting problem, that consists of the following scenario: A turkey is initially alive and a gun is initially unloaded, the goal is to kill the turkey. In order to do it, we need to load the gun first and then shoot. We are going to add, that the final state of the gun will be loaded.

We present this planning problem modeled using language $\mathcal{A}$ and encoded into p-stable semantics. Briefly we remark that an $\mathcal{A}$ model is based on a set of fluents, actions, executable conditions, an initial state and a goal.

In language $\mathcal{A}$ Here we show how to model the problem into language $\mathcal{A}$. We can represent this with two fluents, which are loaded and alive. Loaded means that the gun may be or not loaded. Alive represents the state of the turkey, that can be alive or not alive. So our set or fluents are: {`loaded, alive`}. Also we only need two actions, load the gun and shoot. Our set of actions will be: {`load, shoot`}.
The *domain description* propositions are the executable conditions and, what causes an action A. For example: the load can not shoot if it is not loaded. Representing it into language $\mathcal{A}$ will be as:

```
executable shoot if loaded.
```

The following conditions are very easy to understand:

```
executable load if not loaded.
shoot causes not alive if alive.
shoot causes not loaded if loaded.
load causes loaded if not loaded.
```

The *observation language*, which declare the initial state of the problem,is that the turkey is alive and the gun is not loaded, so our *initially* state would be: {`alive, not loaded`}.
Finally the *query language* propositions, that mean the *goal*, which are to kill the turkey and let the gun loaded. Our set will be formed by the fluents: {`not alive, loaded`}.

In p-stable semantics In this section we present its encoding based on p-stable semantics. In particular we use the new implementation for p-stable semantics [11].
Briefly, we mention that is close similar from smodels [2], which you can represent planning by describing each fluent and action into clauses.
By giving the model of the language $\mathcal{A}$ it is very easy to model it into p-stable semantics. The *fluents* and *actions* can be represented respectively as follows:

```
fluent(loaded).
fluent(alive).
action(load).
action(shoot).
```

The next rule specifies that T in time(T) is an integer, and it can be from zero to N, here we set N as number three.

```
time(0..3)
```

The following rules are the *domain description* problem, as it has modeled before in language $\mathcal{A}$, in p-stable semantics is also very easy to understand.

```
occurs(shoot, T) :- holds(loaded, T), time(T).
occurs(load, T) :- not_holds(loaded, T), time(T).
not_holds(alive, T+1) :- occurs(shoot, T), holds (alive, T), time(T).
not_holds(loaded, T+1) :- occurs(shoot, T), holds(loaded, T), time(T).
holds(loaded, T+1) :- occurs(load, T),
not_holds(loaded, T), time(T).
```

Because it is not allowed to have negative atoms at the head of a rule, we use the auxiliary form as not_holds, in order to avoid inconsistency. If a fluent F holds at time T, it can not be that F not holds at the same time T. Representing this into code:

```
holds(F, T):- not not_holds(F, T), time(T), fluent(F).
```

The following rules represent the *observation language*, that is the initial state of the problem. As it has been shown before, the turkey is alive and the gun is not loaded. In order to translate this into code, we express it into the fact that it holds or not at time 0, which it is the initial time.

```
holds(alive, 0).
not_holds(loaded, 0).
```

Finally the rules of the *query language*, indicates the goal state that we want to have at time N, where N represents the number of steps. Here we need only three steps.

```
not_holds(alive, 3).
holds(loaded, 3).
```

With these rules, we have modeled the yale shooting problem. Then we use a recent implementation of p-stable semantics [11] in order to have the p-stable models that satisfy the conditions mentioned before. These models are the plan, that is the sequence of actions that must be made in order to archive the goal. This implementation use the lparse syntaxis and it is executed as following:

```
lparse program.lp | ./PstableResolver -p 0
```

We only obtained one model, that it is the plan. Because it could not be found another plan that satisfies the rules shown before. We explain the plan into the following table:

Time	Action	State
0	load	alive, not loaded
1	shoot	alive, loaded
2	load	not alive, not loaded
3	-	not alive, loaded

It is easy to see, that in order to kill the turkey and the gun be loaded, we need to first load the gun, then shoot at the turkey and finally load the gun again.

Comparing the results obtained in this example with p-stable models and answer sets, they both obtain models in different ways according to its semantics. However, in [5] it is proved that for a normal program the p-stable models contain the answer sets, which means that p-stable semantics can bring more plans, than stable semantics does for normal programs.

4 Conclusion

Planning involves the representation of actions and world models, reasoning about the effects of actions, and so on. We show that p-stable semantics is a good way to model and solve planning problems, giving us with the p-stable models the plans that we need, in order to go from an initial state to a goal. It can be applied in a variety of tasks including robotics, process planning, autonomous agents and spacecraft mission control [3]. We have explained in this paper how to model a planning problem into language $\mathcal{A}$ and how to translate into p-stable semantics and how to encode it. For future work, we are interested in create an interface for the planning grounding like coala [6] from the potassco project , which works with answer set solving, but instead of stable semantics apply the implementation of p-stable semantics which has been shown before.

References

1. J. J. Alferes, F. Banti, and A. Brogi. A principled semantics for logic programs updates. In *Nonmonotonic Reasoning, Action, and Change (NRAC'03)*, 2003.
2. ASP_Solver. Web location of Smodels: http://www.tcs.hut.fi/software/smodels/.
3. M. Balduccini, M. Gelfond, M. Nogueira, and R. Watson. Planning with the USA-Advisor. In D. Kortenkamp, editor, *3rd NASA International workshop on Planning and Scheduling for Space*, Oct 2002.
4. C. Baral. *Knowledge Representation, reasoning and declarative problem solving with Answer Sets.* Cambridge University Press, Cambridge, 2003.

5. J. L. C. Carranza. *Fundamentos matemáticos de la semántica pstable en programación lógica*. PhD thesis, Benemérita Universidad Autónoma de Puebla, Nov 2008.
6. Coala. http://www.cs.uni-potsdam.de/ tgrote/coala/.
7. Y. Dimopoulos, B. Nebel, and J. Koehler. Encoding Planning Problems in Non-Monotonic Logic Programs. In *Proceedings of the Fourth European Conference on Planning*, pages 169–181. Springer-Verlag, 1997.
8. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.
9. M. Gelfond and V. Lifschitz. Action languages. *Electron. Trans. Artif. Intell.*, 2:193–210, 1998.
10. J. W. Lloyd. *Foundations of Logic Programming*. Springer, Berlin, second edition, 1987.
11. A. Marin. Computing the pstable semantics. https://sites.google.com/site/computingpstablesemantic.
12. M. Osorio, J. Arrazola, and J. L. Carballido. Logical weak completions of paraconsistent logics. *Journal of Logic and Computation, doi: 10.1093/logcom/exn015*, 2008.
13. J. Rintanen. *Introduction of Automated Planning*. Albert-Ludwings-Universitat Freiburg, 2006.
14. T. C. Son and E. Pontelli. Planning with preferences using logic programming. *Theory and Practice of Logic Programming (TPLP)*, 6:559–607, 2006.

N'_5 as an extension of G'_3

Mauricio Osorio[1] and José Luis Carballido[2]

[1] Universidad de las Américas - Puebla
CENTIA, Sta. Catarina Mártir, Cholula, Puebla, 72820 México
osoriomauri@gmail.com
[2] Benemérita Universidad Autóma de Puebla,
Facultad de Ciencias de la Computación, Puebla, México
jlcarballido7@gmail.com
(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. We present some results related to the substitution theorem and the standard form of formulas in the 5-valued paraconsistent logic N'_5 introduced in [1]. This logic has two negation connectives and extends G'_3, a paraconsistent logic recently introduced [15]. We also show that N'_5 is not a maximal paraconsistent logic.

Keywords: *paraconsistent, strong negation, substitution theorem, N'_5 logic.*

1 Introduction

The present work can be placed in the context of paraconsistent logics. Briefly speaking, following Béziau [4], a logic is paraconsistent if it has a negation $\neg$, which is paraconsistent in the sense that $a, \neg a \nvdash b$, and at the same time has enough strong properties to be called a negation. Nevertheless, there is no paraconsistent logic that is unanimously recognised as a good paraconsistent logic [4].

In spite of this lack of agreement, paraconsistent logics have important applications, specifically [7] mention three applications in different fields: Mathematics, Artificial Intelligence and Philosophy. In relation to the second one, the authors mention that in certain domains, such as the construction of expert systems, the presence of inconsistencies is almost unavoidable (see for example [8]). An application that has not been fully recognized is the use of paraconsistent logics in non-monotonic reasoning. In this sense [14,16,17,18] illustrate such novel applications. Thus, the research on paraconsistent logics is far from being over.

A paraconsistent logic of particular interest to us is G'_3, which has been studied in [15,16,18]. In this paper we study a logic closely related to G'_3. G'_3 is a three-valued paraconsistent logic that can express both, the Lukasiewicz L_3 and the Gödel G_3 logics. In particular it is worth mentioning that it can also express classical logic. G'_3 can also be expressed in terms of the Lukasiewicz L_3 logic [15]. G'_3 can be defined in terms of an axiomatic system, in fact in [15] the authors present an axiomatization of G'_3 together with a completeness and robustness theorem: the tautologies of the multivaled version of G'_3 are the

Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 199-210

same as the theorems in the proof system version of it. The family of axioms presented in [15] to define G'_3 include all the axioms of C_ω, thus G'_3 can be considered an extension of C_ω. An important property of G'_3 and shared by the logic Z [4], is that it satisfies the substitution theorem. G'_3 also satisfies the deduction theorem and the De Morgans laws. Another important feature about G'_3 is the fact that the formula $(a \wedge \neg a) \rightarrow b$ is not a theorem, that is, G'_3 is a paraconsistent logic. This property allows G'_3, as well as some other paraconsistent logics $(C_\omega, Pac, P-FOUR)$, to be the formalism to define the p-stable semantics, a semantics adequate to represent knowledge [18].

The p-stable semantics has been introduced recently as a tool to represent knowledge and has found applications in areas such as argumentation theory [10], updates [19] and preferences [20]. An implementation of the p-stable semantics using open sources is described in [23]. For more information about applications of the p-stable semantics see [21,22].

We study some properties of a five-valued paraconsistent logic strongly related to G'_3, we call it N'_5. N'_5 is a logic with strong negation that has the property of being a conservative extension of G'_3, i.e. a formula is a tautology in G'_3 if and only if, it is a tautology in N'_5. N'_5 satisfies the substitution theorem and besides, with N'_5 one can easily express the 5-valued Nelson's logic $N5$ [11].

N'_5 has one more connective not present in G'_3, its strong negation, which makes the logic richer from the theoretical point of view and also gives to it more possibilities of applications in the area of knowledge representation. Specifically, we have empirical evidence that it may be possible to extend the p-stable semantics to a more expressive one by using N'_5 in the same way G'_3 is used to formalize the p-stable semantics. At this point it is worth to mention the fact that the strong negation connective can be used in knowledge representation to express the notion of "usually", so that semantics defined in terms of a logic having a strong negation are more suitable in certain applications that those based on logics without it. For more information about strong negation in this context see[13,24].

The structure of our paper is as follows. In section 2 we present some of the related work that has been done. Section 3 describes the general background needed for reading the paper, including the definition of C_ω logic, a paraconsistent logic whose axioms are also axioms of G'_3. In the same section we present the original definition of the three-valued logic G'_3. In Section 4 we present N'_5 logic, a five-valued logic with two negations, and show that each of these negations can not be expressed by a formula in terms of the other four connectives in the logic. In the same section we present Theorem 6, one of our main results, which establishes that our logic N'_5 is a conservative extension of G'_3. We also present a substitution theorem for N'_5 and introduce the concept of standard form. On Section 5 we present our conclusions and we address future work.

2 Related Work

Let us recall that there are many paraconsistent logics, however most of them are defined in terms of axiomatic systems. The family of paraconsistent logics

$C_n, 0 < n < \omega$ were introduced in [6], and have been very influential. Later they were generalized to stronger versions: $C_n^+, 0 < n < \omega$ [7]. All these logics are not many-valued, and are not maximal in the sense that they can be extended to other paraconsistent logics [5]. We are more interested in multivalued logics with a paraconsistent negation, a strong negation and a substitution theorem valid for a strong biconditional. Among some well studied paraconsistent logics we can mention Pac [2], however it does not have a strong negation and does not satisfy the standard substitution theorem, as the formula $\neg(a \rightarrow b) \leftrightarrow (a \wedge \neg b)$ shows: it is a tautology, but the formula $\neg\neg(a \rightarrow b) \leftrightarrow \neg(a \wedge \neg b)$ is not a tautology according to an interpretation that assigns values 1 and 0 to the atoms a and b respectively. $J3$ [9] is a paraconsistent logic that possesses a strong negation and extends Pac, but since it is a conservative extension of Pac it does not satisfy the substitution theorem either. In [5] the authors introduce new logical systems, $LFI1$ and $LFI2$ to handle inconsistent data. $LFI1$ is inter-definable with $J3$ and does not satisfy the standard substitution theorem [5]. We do not know whether any of these two logics satisfies a version of the substitution theorem for a strong biconditional. Both, $LFI1$ and $LFI2$, are maximal systems with a strong negation and are defined in terms of axioms.

3 Background

We assume that the reader has some familiarity with basic logic such as chapter one in [12].

We first introduce the syntax of logic formulas considered in this paper. Then we present a few basic definitions about how logics can be built to interpret the meaning of such formulas in order to, finally, give a brief introduction to several of the logics that are relevant for the results of our later sections.

3.1 Syntax of formulas

We consider a formal (propositional) language built from: an enumerable set $\mathcal{L}$ of elements called *atoms* (denoted a, b, c, ...); the binary connectives $\wedge$ (*conjunction*), $\vee$ (*disjunction*) and $\rightarrow$ (*implication*); and the unary connective $\neg$ (*negation*). Formulas (denoted α, β, γ, ...) are constructed as usual by combining these basic connectives together with the help of parentheses. We also use $\alpha \leftrightarrow \beta$ to abbreviate $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$ and $\alpha \leftarrow \beta$ to abbreviate $\beta \rightarrow \alpha$. It is useful to agree on some conventions to avoid the use of so many parenthesis in writing formulas. This will make the reading of complicated expressions easier. First, we may omit the outer pair of parenthesis of a formula. Second, the connectives are ordered as follows: $\neg, \wedge, \vee, \rightarrow$, and $\leftrightarrow$, and parentheses are eliminated according to the rule that, first, $\neg$ applies to the smallest formula following it, then $\wedge$ is to connect the smallest formulas surrounding it, and so on.

3.2 Logic systems

We consider a *logic* simply as a set of formulas that, satisfies the following two properties: (i) is closed under modus ponens (i.e. if α and $\alpha \rightarrow \beta$ are in the logic,

then so is β) and (ii) is closed under substitution (i.e. if a formula α is in the logic, then any other formula obtained by replacing all occurrences of an atom b in α with another formula β is still in the logic). The elements of a logic are called *theorems* and the notation $\vdash_X \alpha$ is used to state that the formula α is a theorem of X (i.e. $\alpha \in X$). We say that a logic X is *weaker than or equal to* a logic Y if $X \subseteq Y$, similarly we say that X is *stronger than or equal to* Y if $Y \subseteq X$.

Hilbert style proof systems There are many different approaches that have been used to specify the meaning of logic formulas, in other words, to define *logics*. In Hilbert style proof systems, also known as axiomatic systems, a logic is specified by giving a set of axioms (which is usually assumed to be closed under substitution). This set of axioms specifies, so to speak, the 'kernel' of the logic. The actual logic is obtained when this 'kernel' is closed with respect to the inference rule of modus ponens. In [15] the authors present an axiomatization of G'_3 , that includes all of the axioms of C_ω [6], (and in particular all of the axioms of positive logic). A slight variant of that axiomatization consists of all of the axioms of C_ω plus the following axioms:

E1 $(\neg\alpha \to \neg\beta) \leftrightarrow (\neg\neg\beta \to \neg\neg\alpha)$
E2 $\neg\neg(\alpha \to \beta) \leftrightarrow ((\alpha \to \beta) \wedge (\neg\neg\alpha \to \neg\neg\beta))$
E3 $\neg\neg(\alpha \wedge \beta) \leftrightarrow (\neg\neg\alpha \wedge \neg\neg\beta)$
E4 $(\beta \wedge \neg\beta) \to (- - \alpha \to \alpha)$
E5 $\neg\neg(\alpha \vee \beta) \leftrightarrow (\neg\neg\alpha \vee \neg\neg\beta)$

We observe that classical logic is obtained by adding to the set any of the formulas, $\alpha \to \neg\neg\alpha, \alpha \to (\neg\alpha \to \beta), (\neg\beta \to \neg\alpha) \to (\alpha \to \beta)$.

Multivalued logics An alternative way to define the semantics for a logic is by the use of truth values and interpretations. Multivalued logics generalize the idea of using truth tables that are used to determine the validity of formulas in classical logic. The core of a multivalued logic is its *domain* of values $\mathcal{D}$, where some of such values are special and identified as *designated*. Logic connectives (e.g. $\wedge$, $\vee$, $\to$, $\neg$) are then introduced as operators over $\mathcal{D}$ according to the particular definition of the logic.

An *interpretation* is a function $I\colon \mathcal{L} \to \mathcal{D}$ that maps atoms to elements in the domain. The application of I is then extended to arbitrary formulas by mapping first the atoms to values in $\mathcal{D}$, and then evaluating the resulting expression in terms of the connectives of the logic (which are defined over $\mathcal{D}$). A formula is said to be a *tautology* if, for every possible interpretation, the formula evaluates to a designated value. The most simple example of a multivalued logic is classical logic where: $\mathcal{D} = \{0, 1\}$, 1 is the unique designated value, and connectives are defined through the usual basic truth tables. If X is any logic, we write $\models_X \alpha$ to denote that α is a tautology in the logic X. We say that α is a logical consequence of a set of formulas $\Gamma = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ (denoted by $\Gamma \models_X \alpha$) if $\bigwedge \Gamma \to \alpha$ is a tautology, where $\bigwedge \Gamma$ stands for $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$.

Note that in a multivalued logic, so that it can truly be a *logic*, the implication connective has to satisfy the following property: for every value $x \in \mathcal{D}$, if there

is a designated value $y \in \mathcal{D}$ such that $y \to x$ is designated, then x must also be a designated value. This restriction enforces the validity of modus ponens in the logic. The inference rule of substitution holds without further conditions because of the functional nature of interpretations and how they are evaluated.

In this paper we consider the *standard* substitution, here represented with the usual notation: $\varphi[\alpha/p]$ will denote the formula that results from substituting the formula α in place of the atom p, wherever it occurs in φ. Recall the recursive definition: if φ is atomic, then $\varphi[\alpha/p]$ is α when φ equals p, and φ otherwise. Inductively, if φ is a formula $\varphi_1 \# \varphi_2$, for any binary connective $\#$. Then $\varphi[\alpha/p]$ will be $\varphi_1[\alpha/p] \# \varphi_2[\alpha/p]$. Finally, if φ is a formula of the form $\neg\varphi_1$, then $\varphi[\alpha/p]$ is $\neg\varphi_1[\alpha/p]$.

3.3 The multivalued logic G'_3

As previously noted G'_3 can also be presented as a multivalued logic. Such presentation is given in [18]. In this form it is defined through a 3-valued logic with truth values in the domain $\mathcal{D} = \{0, 1, 2\}$ where 2 is the designated value. The evaluation functions of the logic connectives are then defined as follows: $x \wedge y = \min(x, y)$; $x \vee y = \max(x, y)$; and the $\neg$ and $\to$ connectives are defined according to the truth tables given in Table 1. We write $\models \alpha$ to denote that the formula α is a tautology, namely that α evaluates to 2 (the designated value) for every valuation. We say that α is a logical consequence of a set of formulas $\Gamma = \{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ (denoted by $\Gamma \models \alpha$) if $\bigwedge \Gamma \to \alpha$ is a tautology, where $\bigwedge \Gamma$ stands for $\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_n$.

x	$\neg x$
0	2
1	2
2	0

$\to$	0	1	2
0	2	2	2
1	0	2	2
2	0	1	2

Table 1. Truth tables of connectives in G'_3.

The next couple of results are facts we already know about the logic G'_3

Theorem 1. *[15] For every formula α, α is a tautology in G'_3 iff α is a theorem in G'_3.*

Theorem 2 (Substitution theorem for G'_3-logic). *[15] Let α, β and ψ be G'_3-formulas and let p be an atom. If $\alpha \leftrightarrow \beta$ is a tautology in G'_3 then $\psi[\alpha/p] \leftrightarrow \psi[\beta/p]$ is a tautology in G'_3.*

Corollary 1. *[15]Let α, β and ψ be G'_3-formulas and let p be an atom. If $\alpha \leftrightarrow \beta$ is a theorem in G'_3 then $\psi[\alpha/p] \leftrightarrow \psi[\beta/p]$ is a theorem in G'_3.*

Next, we present a new result, it gives an extension of G'_3, however the resulting logic does not satisfy the substitution theorem.

Theorem 3. *The G'_3 logic is not maximal, there exists at least one paraconsistent logic that contains properly all of the tautologies of G'_3.*

Proof. Let CG'_3 be the logic that results from G'_3 when we allow the values 1 and 2 to be designated, then it is clear that any formula that is a tautology in G'_3 is also a tautology in CG'_3. On the other hand the formula $((a \to b) \to a) \to a$ which is not a tautology in G'_3 as shown by a valuation that assigns the values 1 and 0 to a and b respectively, becomes a tautology in CG'_3 as it is easy to check.

To see that CG'_3 is paraconsistent, we note that an interpretation that assigns the values 1 and 0 to the atoms a and b respectively shows that the formula $(a \wedge \neg a) \to b$ is not a tautology. □

Let us observe that the substitution theorem that holds in G'_3 is not valid in the new logic CG'_3. The formula $[((a \to b) \to a) \to a] \leftrightarrow [(a \vee \neg a)]$ is a tautology in CG'_3, but the formula $\neg[((a \to b) \to a) \to a] \leftrightarrow \neg[(a \vee \neg a)]$ is not.

4 The multi-valued logic N'_5

We present N'_5, a 5-valued logic. We will use the set of values $\{-2, -1, 0, 1, 2\}$. Valid formulas evaluate to 2, the chosen designated value. The connectives $\wedge$ and $\vee$ correspond to the *min* and *max* functions in the usual way. For the other connectives, the associated truth tables are as follows:

$\to$	-2	-1	0	1	2
-2	2	2	2	2	2
-1	2	2	2	2	2
0	2	2	2	2	2
1	-1	-1	0	2	2
2	-2	-1	0	1	2

$\neg$	
-2	2
-1	2
0	2
1	2
2	-2

$\sim$	
-2	2
-1	1
0	0
1	-1
2	-2

$\leftrightarrow$	-2	-1	0	1	2
-2	2	2	2	-1	-2
-1	2	2	2	-1	-1
0	2	2	2	0	0
1	-1	-1	0	2	1
2	-2	-1	0	1	2

Table 2. Truth tables of connectives in N'_5.

We have defined 5 logical connectives, namely $N_c := \{\to, \wedge, \vee, \neg, \sim\}$. Formulas in this logic will also be referred to as N-formulas, they are built from this set of connectives. As usual, if α always evaluates to the designated value, then it is called a tautology. For example the formula $(\alpha \wedge \sim \alpha) \to \beta$ is a tautology for any formulas α and β. The formula $\sim \alpha \to \neg\alpha$ is also a tautology, that is why we will call the connective $\sim$ strong negation. On the other hand, the formula $(\alpha \wedge \neg\alpha) \to \beta$ is not a tautology, a fact easy to verify.

Let us note that N'_5 logic is in some way similar to N_5 logic (see [11] for more information on N_5). The only difference is that in N_5, $\neg 1 = -1$, but in N'_5, $\neg 1 = 2$. Moreover, with N'_5 logic we can express N_5 logic.

Remark 1. N'_5 logic can express N_5 logic.

For there are at least two ways of expressing the N_5 formula $\neg\alpha$ in terms of the connectives of N_5'. These are given by the expressions $\alpha \rightarrow \sim\alpha$ and $\alpha \rightarrow (\neg\alpha \wedge \neg\neg\alpha)$. In particular $\neg\alpha \wedge \neg\neg\alpha$ expresses the N_5 constant $\bot$.

It is important to note that N_5' is in fact a paraconsistent extension of Nelson $N5$ logic, it is the slight difference in the definition of the connective $\neg$, which makes the difference between the two logics. In fact the next result holds.

Theorem 4. *N_5' is a conservative extension of N_5. A N_5-formula A is a tautology in N_5' if and only if it is a tautology in N_5.*

We will use the abbreviation $\alpha \leftrightarrow \beta := (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$.

The next proposition observes the fact that the connective $\neg$ can not be expressed in terms of the other 4 connectives:

Proposition 1. *There is no formula $\beta(a)$ in N_5' containing only the atom a and connectives in $\{\sim, \wedge, \vee, \rightarrow\}$, such that $\models \neg a \leftrightarrow \beta(a)$.*

Proof. First we notice that the connective $\neg$ gives different signs to the truth values 1 and 2. We show that any formula built with the other 4 connectives gives the same sign to the truth values 1 and 2. We apply induction on the number of connectives in the formula $\beta(a)$. Let v be a valuation such that $v(a) = 1$, then $v(\sim a) = -1$, $v(a \wedge a) = 1$, $v(a \vee a) = 1$ and $v(a \rightarrow a) = 2$. If v is a valuation such that $v(a) = 2$, then $v(\sim a) = -2$, $v(a \wedge a) = 2$, $v(a \vee a) = 2$ and $v(a \rightarrow a) = 2$.

Let ϕ, η be two N-formulas of the atom a. Let $\phi(i)$, $\eta(i)$ the truth value of the formulas for a given valuation v for which $v(a) = i$. Let us assume that $\phi(1)$ and $\phi(2)$ are both positive or negative, and also $\eta(1)$ and $\eta(2)$ are both positive or negative. Then according to the truth tables of the connectives $\rightarrow, \wedge, \vee, \sim$, we have:

$\phi(i) \rightarrow \eta(i)$ is positive for both $i \in \{1, 2\}$, or is negative for both $i \in \{1, 2\}$

$\phi(i) \vee \eta(i)$ is positive for both $i \in \{1, 2\}$, or is negative for both $i \in \{1, 2\}$

$\phi(i) \wedge \eta(i)$ is positive for both $i \in \{1, 2\}$, or is negative for both $i \in \{1, 2\}$

$\sim\phi(1), \sim\phi(2)$ are both positive or both negative.

We see from this and the definition of the connective $\leftrightarrow$, that the formula $\neg a \leftrightarrow \beta(a)$ does not evaluate to 2 for every interpretation. □

A similar result holds for the connective $\sim$.

Proposition 2. *There is no formula $\beta(a)$ in N_5' containing only the atom a and connectives in $\{\neg, \wedge, \vee, \rightarrow\}$, such that $\models \sim a \leftrightarrow \beta(a)$.*

Proof. Notice that the connective $\sim$ assigns the value 1 to the truth value -1. We show that any given formula of one single atom a built only in terms of the other 4 connectives assigns always one of the values $-1, -2, 2$ to the value -1.

We apply induction on the number of connectives in the formula $\beta(a)$. Let v a valuation such that $v(a) = -1$, then $v(\neg a) = 2$, $v(a \wedge a) = -1$, $v(a \vee a) = -1$ and $v(a \rightarrow a) = 2$.

Assume the result valid for any formula with less than n connectives. Let ϕ, η two formulas satisfying the induction hypothesis. Then $\phi(-1), \eta(-1) \in$

$\{2, -2, -1\}$. It follows from the tables for the connectives of N'_5 that $\neg\phi(-1) \in \{2, -2\}$, and $(\phi \wedge \eta)(-1), (\phi \vee \eta)(-1), (\phi \rightarrow \eta)(-1) \in \{2, -2, -1\}$.

Thus, according to the truth table for $\leftrightarrow$, the formula $\sim a \leftrightarrow \beta(a)$ does not evaluate to 2 for every interpretation. □

Remark 2. Observe the following formulas in N'_5:

$$
\begin{aligned}
&1.- \models\sim (\alpha \rightarrow \beta) \leftrightarrow \alpha\wedge \sim \beta.\\
&2.- \models\sim (\alpha \wedge \beta)\leftrightarrow \sim \alpha\vee \sim \beta.\\
&3.- \models\sim (\alpha \vee \beta)\leftrightarrow \sim \alpha\wedge \sim \beta.\\
&4.- \models\sim\sim \alpha \leftrightarrow \alpha.\\
&5.- \models\sim \neg\alpha \leftrightarrow \neg\neg\alpha.\\
&6.- \models\sim \alpha \rightarrow \neg\alpha.
\end{aligned}
$$

What is important about this remark, is that these formulas have the same structure as those theorems of Nelson introduced previously in [11] for the construction of extensions, with strong negation, of intuitionistic logic; In fact, formulas $1 - 6$ show the classical-like behavior of strong negation, in particular, formula 6 honors the adjective *strong*.

Next we present a couple of theorems relative to our logic N'_5. The proofs can be found in [1]

Theorem 5. *Given α and β two formulas, then:*

1. *If α and $\alpha \rightarrow \beta$ are tautologies, then β is also a tautology.*

2. *If α is a tautology, then $\neg\neg\alpha$ is also a tautology.*

A very important result is that N'_5 logic is a conservative extension of G'_3 logic, as the following theorem shows.

Theorem 6. *For every G'_3-formula α, α is a tautology in N'_5 iff α is a tautology in G'_3.*

This result together with theorem 4 shows that N'_5 extends two different logics N_5 and G'_3.

4.1 Substitution

A particular feature of our N'_5 logic is that the symbol $\leftrightarrow$ does not define a *congruential relation* on formulas, note that it can be the case that $\alpha \leftrightarrow \beta$ is a tautology, but $\sim\alpha \leftrightarrow \sim\beta$ is not. A particular example is the following: Take α_1 to be $\sim (a \rightarrow b)$ and α_2 to be $a\wedge \sim b$. Clearly $\alpha_1 \leftrightarrow \alpha_2$ is a tautology, but $\sim \alpha_1\leftrightarrow \sim \alpha_2$ is not (take $I(a) = I(b) = 1$). This property also holds in N_5.

Thus, when we refer to *equivalence* of formulas, we will have to be more precise and make some particular considerations. The term *weak equivalence* will mean that $\alpha \leftrightarrow \beta$ is a tautology. There is a stronger notion of equivalence of N'_5-formulas, which we will call N'_5*-equivalence*, and it holds when both $\alpha \leftrightarrow \beta$ and $\sim\alpha \leftrightarrow \sim\beta$ are tautologies. For this purpose, we define a new connective $\Leftrightarrow$. We write $\alpha \Leftrightarrow \beta$ to denote the formula: $(\alpha \leftrightarrow \beta) \wedge (\sim\alpha \leftrightarrow \sim\beta)$. The reader can easily verify that $\alpha \leftrightarrow \beta$ is a tautology iff for every valuation v, $v(\alpha) > 0$ implies

$v(\alpha) = v(\beta)$ and by symmetry, $v(\beta) > 0$ implies $v(\alpha) = v(\beta)$, while $\alpha \Leftrightarrow \beta$ is a tautology iff for every valuation v, $v(\alpha) = v(\beta)$. This can be seen in the following truth tables:

$\leftrightarrow$	-2	-1	0	1	2
-2	2	2	2	-1	-2
-1	2	2	2	-1	-1
0	2	2	2	0	0
1	-1	-1	0	2	1
2	-2	-1	0	1	2

$\Leftrightarrow$	-2	-1	0	1	2
-2	2	1	0	-1	-2
-1	1	2	0	-1	-1
0	0	0	2	0	0
1	-1	-1	0	2	1
2	-2	-1	0	1	2

Table 3. Truth tables for the biconditionals.

The next two theorems are proved in [1].

Theorem 7 (Basic Substitution theorem). *Let α, β and ψ be N_5'-formulas and let p be an atom. If $\alpha \Leftrightarrow \beta$ is a tautology then $\psi[\alpha/p] \Leftrightarrow \psi[\beta/p]$ is a tautology.*

To be able to apply standard substitution we require N_5'-equivalence of formulas to hold. However, in certain cases this condition may be too strong. We are only interested in the particular cases where weak equivalence of formulas suffices for substituting. The first such a case is when substitution is not done inside the scope of a $\sim$ symbol.

Theorem 8. *Let α, β and ψ be N_5'-formulas and let p be an atom such that p does not occur in ψ within the scope of a $\sim$ symbol. If $\alpha \leftrightarrow \beta$ is a tautology then $\psi[\alpha/p] \leftrightarrow \psi[\beta/p]$ is a tautology.*

4.2 Standard form

We present the notion of a standard form of a formula.

Definition 1. *We define the function $S : N_5' - formulas \rightarrow N_5' - formulas$ as follows: If a is an atom and α, β are N_5'-formulas, then*

$$
\begin{array}{ll@{\qquad}ll}
S(a) & = a, & S(\alpha \wedge \beta) & = S(\alpha) \wedge S(\beta), \\
S(\neg a) & = \neg a, & S(\alpha \vee \beta) & = S(\alpha) \vee S(\beta), \\
S(\sim a) & = \sim a, & S(\sim (\alpha \rightarrow \beta)) & = S(\alpha) \wedge S(\sim \beta), \\
S(\sim \neg\alpha) & = \neg\neg S(\alpha), & S(\sim (\alpha \wedge \beta)) & = S(\sim \alpha) \vee S(\sim \beta), \\
S(\neg\alpha) & = \neg S(\alpha), & S(\sim (\alpha \vee \beta)) & = S(\sim \alpha) \wedge S(\sim \beta), \\
S(\alpha \rightarrow \beta) & = S(\alpha) \rightarrow S(\beta). & S(\sim\sim \alpha) & = S(\alpha).
\end{array}
$$

Definition 2 (Standard Form). *An N_5'-formula φ is said to be in* standard form *if $S(\varphi) = \varphi$*

Intuitively a formula is in standard form if it has all occurrences of the $\sim$ connective just in front of an atom. Let us observe also that we did not define $S(\sim \neg\alpha)$ as $S(\alpha)$, we want the formula $\alpha \leftrightarrow S(\alpha)$ to be a tautology for any formula α and the formula $\sim \neg a \leftrightarrow a$ is not a tautology for an atom a.

Example 1. Take the formula $\varphi :=\sim (a \rightarrow \neg b) \wedge \sim c$. Then its standard form is $S(\varphi) := a \wedge b \wedge \sim c$.

The result we present in relation to standard forms affirms that $S(\phi)$ is a tautology if and only if ϕ is a tautology. In order to prove this, we point out the next properties about tautologies that are consequences of the definition of the $\leftrightarrow$ connective: If $A \leftrightarrow B$ and $B \leftrightarrow C$ are tautologies, then $A \leftrightarrow C$ is a tautology, if $A \leftrightarrow B$ and $C \leftrightarrow D$ are tautologies, then $A \wedge C \leftrightarrow B \wedge D$ and $A \vee C \leftrightarrow B \vee D$ are tautologies.

Theorem 9. *For any N'_5-formula ψ, $\psi \leftrightarrow S(\psi)$ is a tautology in N'_5.*

Proof. By structural induction:

Base case: One can easily check that the proposition is true for any of the twelve formulas that define the standard form if a, α, β are atoms.

Case 1) $\psi = \phi \wedge \eta$.

Let us assume that $\phi \leftrightarrow S(\phi)$ and $\eta \leftrightarrow S(\eta)$ are tautologies. Applying Theorem 8 to the formula $p \wedge \eta$ and the first tautology above we obtain that $\phi \wedge \eta \leftrightarrow S(\phi) \wedge \eta$ is a tautology. Now we apply the same Theorem to the formula $S(\phi) \wedge p$ and the second tautology and obtain that $S(\phi) \wedge \eta \leftrightarrow s(\phi) \wedge S(\eta)$ is a tautology. From this we conclude that $\phi \wedge \eta \leftrightarrow S(\phi) \wedge S(\eta)$ is a tautology.

The cases $\psi = \phi \vee \eta$ and $\psi = \phi \rightarrow \eta$ are done the same way by just replacing the corresponding connective.

Case 2) $\psi = \neg\phi$.

We apply Theorem 8 to the formula $\neg p$ and the tautology $\phi \leftrightarrow S(\phi)$ according to the inductive hypothesis to obtain the tautology $\neg\phi \leftrightarrow \neg S(\phi)$. This is equivalent to $\neg\phi \leftrightarrow S(\neg\phi)$.

Case 3) $\psi = \sim\neg\phi$.

By hypothesis $\phi \leftrightarrow S(\phi)$ is a tautology. Since $\sim\neg\phi \leftrightarrow \neg\neg\phi$ is a tautology and $S(\sim\neg\phi) = S(\neg\neg\phi)$, it is enough to prove that $\neg\neg\phi \leftrightarrow S(\neg\neg\phi)$ is a tautology. But $S(\neg\neg\phi) = \neg\neg S(\phi)$, and the result follows by induction hypothesis.

Case 4) $\psi = \sim(\phi \wedge \eta)$.

To prove that $\psi \leftrightarrow S(\psi)$ is a tautology is equivalent to prove, according to remark 2, that $\sim\phi \vee \sim\eta \leftrightarrow S(\sim\phi) \vee S(\sim\eta)$ is a tautology. But by induction hypothesis $\sim\phi \leftrightarrow S(\sim\phi)$ and $\sim\eta \leftrightarrow S(\sim\eta)$ are tautologies, from which the result follows.

The cases $\psi = \sim(\phi \vee \eta)$ and $\psi = \sim(\phi \rightarrow \eta)$ are done the same way by just replacing the corresponding connective.

Case 5) $\psi = \sim\sim\phi$.

By hypothesis $\phi \leftrightarrow S(\phi)$ is a tautology. Since $S(\sim\sim\phi) = \sim\sim S(\phi)$, this case reduces to prove that $\sim\sim\phi \leftrightarrow \sim\sim S(\phi)$ is a tautology. The result follows from the fact that $\alpha \leftrightarrow \sim\sim\alpha$ is a tautology for any formula α. □

As an immediate consequence we have the next important result that has been already presented in [1]:

Corollary 2. *For any N'_5-formula φ, φ is a tautology in N'_5 iff $S(\varphi)$ is a tautology in N'_5.* □

4.3 N'_5 is not a maximal paraconsistent logic

Before ending this section, we present one more result, similar to theorem 3, about the fact that N'_5 can be extended.

Theorem 10. *The N'_5 logic is not maximal, there exists at least one paraconsistent logic that contains properly all of the tautologies of N'_5*

Proof. Let CN'_5 be the logic that results from N'_5 when we allow the values 1 and 2 to be designated, then it is clear that any formula that is a tautology in N'_5 is also a tautology in CN'_5. On the other hand the formula $((a \rightarrow b) \rightarrow a) \rightarrow a$ which is not a tautology in N'_5 as shown by a valuation that assigns the values 1 and 0 to a and b respectively, becomes a tautology in CN'_5 as it is easy to check.

To see that CN'_5 is paraconsistent, we note that an interpretation that assigns the values 1 and 0 to the atoms a and b respectively shows that the formula $(a \wedge \neg a) \rightarrow b$ is not a tautology. □

As in the case of G'_3 and CG'_3 logics, the substitution theorem that holds in N'_5 is not valid in the new logic CN'_5. As the reader can easily check, the formula $[((a \rightarrow b) \rightarrow a) \rightarrow a] \leftrightarrow [(a \vee \neg a)]$ is a tautology in CN'_5, but the formula $\neg[((a \rightarrow b) \rightarrow a) \rightarrow a] \leftrightarrow \neg[(a \vee \neg a)]$ is not.

5 Conclusions and Future Work

We introduced a 5-valued logic called N'_5. This logic is a conservative extension of the 3-valued logic G'_3, which accepts an axiomatization. Our logic N'_5 possesses two negations, one of them, $(\sim)$, is a strong negation that makes the logic more expressive when representing knowledge. Results presented in this paper include a substitution theorem for N'_5, the preservation of tautologies by the standard form in N'_5 and the fact that the N'_5 logic can express N_5 logic, a logic that is suitable to express ASP. Finally, as mentioned in the introduction, we are interested in exploring possible ways of extending the p-stable semantics to a more expressive semantics by means of the use of a logic with strong negation, N'_5 seems to be a suitable candidate for the formalization of such a semantics.

6 Funding

This work was supported by the Consejo Nacional de Ciencia y Tecnología [CB-2008-01 No.101581].

References

1. J. Arrazola and M. Osorio and E. Ariza. The N'5 logic. Sixt Latin Americn Workshop On New Mrthods of reasoning, 25-35, 2010.
2. A. Avron. *Natural 3-valued logics- characterization and proof theory.* The Journal of Symbolic Logic, 56(1): 276-294, 1991.

3. A. Avron. *5-valued Non-deterministic Semantics for the Basic Paraconsistent Logic mCi.* School of Computer Science, Tel Aviv University 2008.
4. J. Y. Béziau. The Paraconsistent Logic Z. A possible solution to Jaskowski's problem. In Logic and logical philosophy, 15: 99–111, 2006.
5. W. A. Carnielli, J. Marcos and R. de Amo. Formal Inconsistency and evolutionary databases. Logic and Logical Philosophy, Vol 8, 2000, pages 115–152.
6. N. da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15(4):497–510, 1974.
7. N. da Costa, J. I. Béziau and O. A. Bueno. Aspects of Paraconsistent Logic. Bull: of the IGPL, 3(4):597–614, 1995.
8. N. da Costa, V. S. Subrahmanian. Paraconsistent logics as a formalism for reasoning about inconsistent knowledge bases. *Artificial Intelligence in Medicine*, 1: 167–174, 1989.
9. I. L. D'Ottaviano and N. da Costa. Sur un problme de Jaskowsky. *Compted Rendus de l'Academie de Sciences de Paris* Sr.A-B 270: 1349-1353, 1970
10. P. M.Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. In *Artificial Intelligence*, 77(2): 321–358, 1995.
11. M. Kracht On extensions of intermediate logics by strong negation evaluation. *The Journal of Philosophical Logic*, 27(1):49–73, 1998.
12. E.Mendelson. *Introduction of Mathematical Logic.* Wadsworth, Belmont, CA, third edition, 1987.
13. M.Ortiz and M. Osorio. Strong Negation and Equivalence in the Safe Belief Semantics. In *Journal of Logic and Computation* , 499 - 515, April, 2007.
14. M. Osorio. GluckG logic and its applications to non-monotonic reasoning. *LANMR*,286, 2007.
15. M. Osorio and J. L. Carballido. Brief study of G'$_3$ logic. *Journal of Applied Non-Classical Logic*, 18(4):79–103, 2008.
16. M. Osorio, J.Arrazola, and J. L. Carballido. Logical weak completions of paraconsistent logics. *Journal of Logic and Computation*, 18(6):913–940, 2008.
17. M. Osorio, J. A. Navarro, J. Arrazola, and V. Borja. Ground nonmonotonic modal logic S5: New results. *Journal of Logic and Computation*, 15(5):787–813, 2005.
18. M. Osorio, J. A. Navarro, J. Arrazola, and V. Borja. Logics with common weak completions. *Journal of Logic and Computation*, 16(6):867–890, 2006.
19. M. Osorio and C. Zepeda. Update sequences based on minimal generalized pstable models. In *MICAI*, pages 283–293, 2007.
20. M. Osorio and C. Zepeda. Pstable theories and preferences. In *Electronic Proceedings of the 18th International Conference on Electronics, Communications, and Computers (CONIELECOMP 2008)*, March, 2008.
21. M. Osorio, C. Zepeda, and H. Castillo. A formal design model for mechatronic systems. In *Proceedings of 19th International Conference on Electrical, Communications, and Computers 2009, Cholula, Puebla, Mexico*, pages 125–129, 2009.
22. M. Osorio, C. Zepeda, J. C. Nieves, and J. L. Carballido. G'3-stable semantics and inconsistency. *Special Issue of Computacin y Sistemas on Innovative Applications of AI, Revista Iberoamericana de Computación. ISSN: 1405-5546*, 13(1):75–86, Ed. IPN. Centro de Investigación en Computación, 2009.
23. S. Pascucci and A. Lopez. Implementing p-stable with simplification capabilities. *Revista Iberoamericana de Inteligencia Artificial*, 13(41), Spain, 2008.
24. D. Pearce. From here to there: Stable Negation in Logic Programming. *What is the negation?*, Kluver academic publishers, 161-181, 1998.

Extended Ordered Disjunction programs to model Preferences

Mauricio Osorio[1], Claudia Zepeda[2], and José Luis Carballido[2]

[1] Universidad de las Américas - Puebla
CENTIA, Sta. Catarina Mártir, Cholula, Puebla, 72820 México
osoriomauri@googlemail.com
[2] Benemérita Universidad Atónoma de Puebla
Facultad de Ciencias de la Computación,
Puebla, Puebla, México
{czepedac,jlcarballido7}@gmail.com

(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract In [3] Brewka proposed the connective of ordered disjunction to express default knowledge with knowledge about preferences. Later, the authors of [10] define the logic programs with extended ordered disjunction that extends ordered disjunction programs to a wider class of logic programs. Here, we propose to specify a preference ordering among the answer sets of a program with respect to an ordered list of atoms using a particular kind of logic programs with extended ordered disjunction.
Keywords: Answer set programming, preferences, Extended Ordered Disjunction.

1 Introduction

Preferences are useful when we need to find feasible solutions that most satisfy a set of additional requirements of a given problem. In [3] is defined the connective $\times$, called *ordered disjunction*, to express default knowledge with knowledge about preferences in a simple way. While the disjunctive clause $a \vee b$ is satisfied equally by either a or b, to satisfy the ordered disjunctive clause $a \times b$, a will be preferred to b, i.e., a model containing a will have a better *satisfaction degree* than a model that contains b but does not contain a. For example, the natural language statement *"I prefer travel by bus to walk"* can be expressed as $travelBus \times walk$ and a model containing $travelBus$ will be preferred to a model that contains $walk$. Later, the authors of [10] define the *logic programs with extended ordered disjunction* (**ELPOD**) that extends ordered disjunction programs to a wider class of logic programs. There have been other extensions for the operator $\times$ [2,4]. However, while the extension introduced in [10] is in the context of Answer Sets, the extension introduced in [2] for the operator $\times$ is in a different context. Moreover, the extension defined in [4] is in the context of Answer Sets and the authors present interesting examples, however they propose a different methodology to define it w.r.t. the methodology proposed in [10].

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 211-220

It is worth mentioning that if we use a ordered disjunction rule, as is defined in [3], to *specify a preference ordering among the answer sets of a program with respect to an ordered set of atoms* then, the preferred answer set does not corresponds to the answer set that we would expect. For instance, if $P_1 = \{a \leftarrow,\ \ b \leftarrow \neg c,\ \ c \leftarrow \neg b,\ \ d \leftarrow \neg a,\ \ f \leftarrow c, \neg a,\ \ e \leftarrow b, \neg a\}$, then the answer sets of this program are $\{a, b\}$ and $\{a, c\}$. Now, if we want to specify a preference ordering among the answer sets of the program with respect to the ordered set of atoms, such as $[f, c]$ then, we could consider to add to the program P the standard ordered disjunction rule $\{f \times c\}$ that stands for "if f is possible then f otherwise c". Thinking in a preference sense, the intuition indicates that the inclusion-preferred answer sets of $P \cup \{f \times c\}$ should be $\{a, c\}$. However, according to [3], we obtain two inclusion-preferred answer sets $\{a, b, f\}$ and $\{a, c, f\}$.

In this paper we propose to specify a preference ordering among the answer sets of a program with respect to an ordered list of atoms using a particular set of ELPOD's. We propose to use double default negation in each atom of the ordered rule that represents the mentioned list of atoms. Finally, we show how to compute the preferred answer sets for ELPOD's using `psmodels`.

The structure of our paper is as follows. In Section 2 describes the general background needed for reading the paper, including the definition of Answer Sets and ELPOD's. In Section 3, we present how to use double default negation to to obtain the preferred answer sets of a logic program. In Section 4 we propose how to compute the preferred answer sets. On Section 5 we present our conclusions.

2 Background

In this section we summarize some basic concepts and definitions used to understand this paper.

2.1 Logic programs and semantics

We want to stress the fact that in our approach, *a program is interpreted as a propositional theory* and that the only negation used is *default negation*. For readers not familiar with this approach, we recommend [11] for further reading. Hence, we will restrict our discussion to propositional programs.

We shall use *the language of propositional logic* in the usual way, using: propositional symbols: $p, q, \ldots$; propositional connectives: $\wedge, \vee, \rightarrow, \bot$; and auxiliary symbols: $(,)$. A *well formed propositional formula* is inductively defined as usual in logic. An *atom* is a propositional symbol. A *signature* $\mathcal{L}$ is a finite set of atoms. The *signature of a program* P, denoted as $\mathcal{L}_P$, is the set of atoms that occur in P. A *literal* is either an atom p (a positive literal) or the negation of an atom $\neg p$ (a negative literal). A *negated literal* is the negation sign $\neg$ followed by any literal, i.e. $\neg p$ or $\neg\neg p$. We remark that the only negation used in this work is *default negation* and it is represented by the symbol $\neg$. It is worth mentioning that we always can handle the other negation called *classical* or even *strong* negation,

denoted by $-$, by transforming [6] the atoms with classical negation as follows: each atoms with classical negation, $-a$, that occurs in a formula is replaced by a new atom, a', and the rule $\neg(a \wedge a')$ is added. In particular, $f \rightarrow \bot$ is called *constraint* and it is also denoted as $\leftarrow f$. A *regular theory* or *logic program* is just a finite set of clauses, it can be called just *theory* or *program* where no ambiguity arises.

2.2 Answer sets

In some definitions we use Heyting's *intuitionistic* logic [12], which will be denoted by the subscript I. For a given set of atoms M and a program P, we will write $P \vdash_{\mathrm{I}} M$ to abbreviate $P \vdash_{\mathrm{I}} a$ for all $a \in M$. For a given set of atoms M and a program P, we will write $P \Vdash_{\mathrm{I}} M$ to denote that: $P \vdash_{\mathrm{I}} M$; and P is consistent w.r.t. logic I.

Now we define answer sets (or stable models) of logic programs. The stable model semantics was first defined in terms of the so called *Gelfond-Lifschitz reduction* [5] and it is usually studied in the context of syntax dependent transformations on programs. We follow an alternative approach started by Pearce [11] and also studied by Osorio et.al. [8]. This approach characterizes the answer sets of a propositional theory in terms of intuitionistic logic and it is presented in the following theorem. The notation is based on [8].

Theorem 1. *Let P be any theory and M a set of atoms. M is an answer set for P iff $P \cup \neg(\mathcal{L}_P \setminus M) \cup \neg\neg M \Vdash_{\mathrm{I}} M$.*

For instance, if we consider the program $P_1 = \{\ p \leftarrow \neg q, \quad q \leftarrow\ \}$, then we can verify that $M = \{q\}$ is an answer set of P_1 and that $P_1 \cup \{\neg p\} \cup \{\neg\neg q\} \Vdash_{\mathrm{I}} \{q\}$.

2.3 ELPOD's

In [3] the head of ordered disjunction rules is defined in terms of ground literals. In [10] the head and the body of extended ordered disjunction rules are defined in terms of well formed propositional formulas. The authors of [10] consider that a broader syntax for rules could give some benefits. For example, the use of nested expressions could simplify the task of writing logic programs and improve their readability since, it could allows us to write more concise rules and in a more natural way. Other examples are presented in [7,9].

In the following example of an ELPOD, we can see that the broader syntax results more natural, direct and intuitive: I prefer *travel by airplane* to either *travel by bus* other *travel by train*, but between *travel by bus* and *travel by train* I don't have any particular preference. Then, using an ELPOD we could just write $travelByAirplan \times (travelByBus \vee travelByTrain)$.

Definition 1. *A* logic programs with extended ordered disjunction (ELPOD) *is either a well formed propositional formula, or a formula of the form:*

$$f_1 \times \ldots \times f_n \leftarrow g \tag{1}$$

where $f_1, \ldots, f_n, g$ are well formed propositional formulas. An extended ordered disjunction program *is a finite set of extended ordered disjunction rules.*

The formulas $f_1 \ldots f_n$ are usually called the choices of a rule and their intuitive reading is as follows: if the body is true and f_1 is possible, then f_1; if f_1 is not possible, then f_2; ...; if none of $f_1, \ldots, f_{n-1}$ is possible then f_n.

The particular case where all f_i are literals and g is a conjunction of literals corresponds to the original ordered disjunction programs as presented by Brewka in [3], and as we indicated before we call them *standard ordered disjunction programs*. We recall that Brewka's ordered disjunction programs use the strong negation connective. Here we will consider only one type of negation (default negation) but this does not affect the results given in [3]. If additionally $n = 0$ the rule is a constraint, i.e, $\bot \leftarrow g$. If $n = 1$ it is an extended rule and if $g = \top$ the rule is a fact and can be written as $f_1 \times \ldots \times f_n$.

Example 1. A person should travel from his/her home to school in winter. This person prefers to travel by bus and to drink tea inside the bus rather than travel by bicycle. Additionally, he/she prefers to travel by bicycle rather than walk. This person also should consider that part of the path from his/her home to school can become blocked by snow in winter. Then, we can model this situation considering the following ELPOD P:
$winter.$
$(travelBus \wedge drinkTea) \times travelBicycle \times walk \leftarrow winter, \neg pathBlocked.$

Now, we present the semantics of ELPOD's. Most of the definitions presented here are taken from [3]. The relevant difference is the satisfaction degree. The reader may see that the satisfaction degree as defined here is just a straightforward generalization of Brewka's definition, according to our notation and Theorem 1. Hence, standard ordered programs are special cases of ELPOD's, thus all results hold for this restricted class as well.

Definition 2. *Let $r := f_1 \times \ldots \times f_n \leftarrow g$ be an extended ordered disjuntion rule. For $1 \leq k \leq n$ the* k-th option of r *is defined as follows:*

$$r^k := f_k \leftarrow g, \neg f_1, \ldots \neg f_{k-1}$$

Definition 3. *Let P be an ELPOD. P' is a* split program of P *if it is obtained by replacing each rule $r := f_1 \times \ldots \times f_n \leftarrow g$ in P by one of its options $r^1, \ldots, r^k$. M is an answer set of P iff it is an answer set*[1] *of a split program P' of P. Let M be an answer set of P and let $r := f_1 \times \ldots \times f_n \leftarrow g$ be a rule of P. We define the* satisfaction degree of r with respect to M*, denoted by $deg_M(r)$, as follows:*

- *if $M \cup \neg(\mathcal{L}_P \setminus M) \not\vdash_I g$, then $deg_M(r) = 1$.*
- *if $M \cup \neg(\mathcal{L}_P \setminus M) \vdash_I g$ then $deg_M(r) = min_{1 \leq i \leq n} \{i \mid M \cup \neg(\mathcal{L}_P \setminus M) \vdash_I f_i\}$.*

[1] Note that since we are not considering strong negation, there is no possibility of having inconsistent answer sets.

Now we present the relationship between the answer sets of an ELPOD and the satisfaction degree.

Theorem 2. *[1] Let P be an ELPOD. If M is an answer set of P then M satisfies all the rules in P to some degree.*

Definition 4. *[3] Let P be an ELPOD and M a set of literals. We define:* $S^i_M(P) = \{r \in P \mid deg_M(r) = i\}$.

Example 2. Let us consider the ELPOD P from Example 1. The split programs and answer sets of P are:

P' :
$winter.$
$(travelBus \wedge drinkTea) \leftarrow winter, \neg pathBlocked.$

P'' :
$winter.$
$travelBicycle \leftarrow$
$winter, \neg pathBlocked, \neg(travelBus \wedge drinkTea).$

P''' :
$winter.$
$walk \leftarrow winter, \neg pathBlocked,$
$\neg(travelBus \wedge drinkTea), \neg travelBicycle.$

$M_1 = \{winter, (travelBus \wedge drinkTea)\}$, $M_2 = \{winter, travelBicycle\}$ and $M_3 = \{winter, walk\}$ are answer sets of P. If r_1 and r_2 denote the first and second rule of program P respectively, then we have the following satisfaction degrees of each rule: $deg_{M_1}(r_1) = 1$, $deg_{M_1}(r_2) = 1$, $deg_{M_2}(r_1) = 1$, $deg_{M_2}(r_2) = 2$, $deg_{M_3}(r_1) = 1$, $deg_{M_3}(r_2) = 3$. Finally we have that, $S^1_{M_1}(P) = \{r_1, r_2\}$, $S^2_{M_1}(P) = \{\}$, $S^3_{M_1}(P) = \{\}$, $S^1_{M_2}(P) = \{r_1\}$, $S^2_{M_2}(P) = \{r_2\}$, $S^3_{M_2}(P) = \{\}$, $S^1_{M_3}(P) = \{r_1\}$, $S^2_{M_3}(P) = \{\}$, $S^3_{M_3}(P) = \{r_2\}$

Now introduce two different types of preference relations among the answer sets of an ELPOD: inclusion based preference and cardinality based preference.

Definition 5. *[3] Let M and N be answer sets of an ELPOD P. We say that M is* inclusion preferred *to N, denoted as $M >_i N$, iff there is an i such that $S^i_N(P) \subset S^i_M(P)$ and for all $j < i$, $S^j_M(P) = S^j_N(P)$. We say that M is* cardinality preferred *to N, denoted as $M >_c N$, iff there is an i such that $\left|S^i_M(P)\right| > \left|S^i_N(P)\right|$ and for all $j < i$, $\left|S^j_M(P)\right| = \left|S^j_N(P)\right|$.*

Example 3. If we consider the ELPOD P from Example 1 we can verify that $M_1 >_i M_2$ and $M_2 >_c M_3$.

Definition 6. *[3] Let M be an answer set of an ELPOD P. M is an* inclusion preferred answer set *of P if there is no answer set M' of P, $M \neq M'$, such that $M' >_i M$. M is a* cardinality preferred answer set *of P if there is no M' answer set of P, $M \neq M'$, such that $M' >_c M$.*

Example 4. If we consider the ELPOD P from Example 1, we can verify that M_1 is the inclusion preferred answer set of P and also the cardinality preferred answer set of P.

3 Expressing preferences using double negation

In order to specify a preference ordering among the answer sets of a program with respect to an ordered set of atoms, in this section we propose to use a particular set of ELPOD's. Specifically, we propose to add to the original program an extended ordered rule such that this rule is defined using the ordered set of atoms and each atom has *double default negation.*

Formally, as we defined in Background Section, an atom with double negation corresponds to a *negated negative literal* where the only negation used is *default negation.* Let us consider $\neg\neg a$ where a is an atom. Since $\neg\neg a$ is equivalent to the restriction $\leftarrow \neg a$, the intuition behind $\neg\neg a$ is to indicate that it is desirable that a holds in the model of a program.

Hence, if we consider again the previous program P_1 of the Introduction section and we want to to specify a preference ordering among the answer sets of P with respect to the ordered set of atoms $[f, c]$ we have to do the following: First we define the extended ordered disjunction rule using the ordered set of atoms $[f, c]$ such that each atom has *double default negation*, i.e, we define $\{\neg\neg f \times \neg\neg c\}$. Then, we add this extended ordered disjunction rule to the original program P_1, i.e., we define the following extended ordered disjunction program: $P_1 \cup \{\neg\neg f \times \neg\neg c\}$. It is possible to verify that we obtain the desired inclusion-preferred answer set $\{a, c\}$ from this ELPOD.

We explained that the intuition behind an extended ordered rule using negated negative literals is to indicate that we want to specify a preference ordering among the answer sets of a program with respect to an ordered set of atoms. However, in case that the answer sets of the program do not contain any of the atoms in the given ordered set of atoms then the extended ordered rule must allow to obtain all the answer sets of the program. In order to obtain all the answer sets of the program we propose to add a positive literal to the end of the extended ordered rule that we defined as we explained above. This positive literal must be an atom that does not occur in the original program, such as the atom all_pref. For instance, let us consider again the program P_1 that has the answer sets $\{a, b\}$ and $\{a, c\}$. If we want to specify a preference ordering among the answer sets of this program with respect to the ordered set of atoms $C = [f, e]$ then we define the extended ordered rule as we explained above but we add the atom all_pref at the end of the rule, i.e., $\neg\neg f \times \neg\neg e \times all\_pref$. The new extended ordered rule indicates that we prefer the answer sets where f holds to the answer sets where e holds and if there is no answer sets of the program where f or e holds then all answer sets are preferable. It is worth to mentioning that in case that there is no answer sets of the program where f or e holds then all the answer sets will contain the atom all_pref. Hence, we have the following ELPOD, $P_2 = P_1 \cup \{\neg\neg f \times \neg\neg e \times all\_pref\}$. As we expected, we

obtain the inclusion-preferred answer sets: $\{a, c, all\_pref\}$ and $\{a, b, all\_pref\}$ since there is no answer sets of the program containing f or e.

The following short examples also show the role of negated negative literals in an extended ordered program.

Example 5. Let us consider the ordered program $a \times b$ as defined in [3]. We can verify that its inclusion preferred answer set is $\{a\}$, since ordered disjunction corresponds to a disjunction where an ordering is defined. However, the ELPOD $\neg\neg a \times \neg\neg b$ has no answer sets since the extended ordered rule only indicates that we prefer the answer sets containing a to the answer sets containing b but there is no answer sets.

Example 6. Let us consider the ordered program $\{b \leftarrow \neg a,\ \ a \times b\}$. We can verify that the inclusion preferred answer set of it is $\{a\}$ since ordered disjunction corresponds to a disjunction where an ordering is defined. However, the ELPOD $\{b \leftarrow \neg a,\ \ \neg\neg a \times \neg\neg b\}$ only has $\{b\}$ as its inclusion preferred answer set, since the extended ordered rule only indicates that we prefer the answer sets containing a to the answer sets containing b and program $b \leftarrow \neg a$ has only the answer set $\{b\}$.

Now, we formalize our previous discussion about the specification of an ordering among the answer sets of a normal program with respect to an ordered set of atoms using an extended ordered program.

We start introducing a definition and a proposition that allows us to define the most preferred answer set with respect to a program and an ordered set of atoms.

Definition 7. *Let P be a normal program and let M and N be two answer sets of P. Let $C = [c_1, c_2, \ldots, c_n]$ be an ordered set of atoms. The answer set M* is preferred to the answer set N with respect to C *(denoted as $M <_C N$) if*

1. *there exists $i = min(1 \leq k \leq n)$ such that $M \cup \neg(\mathcal{L}_P \setminus M) \vdash_{\mathrm{I}} c_i$ and $N \cup \neg(\mathcal{L}_P \setminus N) \nvdash_{\mathrm{I}} c_i$, and*
2. *for all $j < i$, $M \cup \neg(\mathcal{L}_P \setminus M) \vdash_{\mathrm{I}} c_j$ and $N \cup \neg(\mathcal{L}_P \setminus N) \vdash_{\mathrm{I}} c_j$ or $M \cup \neg(\mathcal{L}_P \setminus M) \nvdash_{\mathrm{I}} c_j$ and $N \cup \neg(\mathcal{L}_P \setminus N) \nvdash_{\mathrm{I}} c_j$.*

Proposition 1. *Let C be an ordered set of atoms; then $<_C$ is a partial order.*

Given an ordered list of atoms C, an answer set M of a normal program P *is most preferred with respect to C* if there is no other answer set N of P that is preferred to M with respect to C.

Example 7. Let $P = \{a \leftarrow,\ \ c \leftarrow \neg b,\ \ b \leftarrow \neg c\}$ and $C = [b, c]$ be an ordered list of atoms. We can verify that P has two answer sets, $\{a, b\}$ and $\{a, c\}$. We also can verify that the answer set $\{a, b\}$ *is preferred to the answer set* $\{a, c\}$ *with respect to C*, i.e., $\{a, b\} <_C \{a, c\}$ and that $\{a, b\}$ is also *the most preferred answer set.*

Now, we define the extended ordered rule that we join to the original normal program in order to obtain a particular ELPOD. From this last ELPOD we obtain the most preferred answer sets of the normal program with respect to the given ordered set of atoms C.

Definition 8. *Let P be a normal program and $C = [c_1, c_2, \ldots, c_n]$ be an ordered set of atoms such that $C \subseteq \mathcal{L}_P$. We define an* extended ordered rule defined from C, *denoted as r_C, as follows: $r_C := \neg\neg c_1 \times \neg\neg c_2 \times \ldots \times \neg\neg c_n \times all\_pref$ such that $all\_pref \notin \mathcal{L}_P$.*

Example 8. Let us consider again program P from Example 7, and also the ordered set of atoms $C = [b, c]$. Then, the *extended ordered rule defined from* C, denoted as r_C is the following: $r_C := \neg\neg b \times \neg\neg c \times all\_pref$.

Here, we present Lemma 1 that allows us to obtain the most preferred answer set of a normal program with respect to an ordered set of atoms based on a particular extended ordered program.

Lemma 1. *Let P be a normal program and let $C = [c_1, c_2, \ldots, c_n]$ be an ordered list of atoms such that $C \subseteq \mathcal{L}_P$. Let r_C be the extended ordered rule defined from C. Then M is an inclusion preferred answer set of $P \cup r_C$ iff $(M \cap \mathcal{L}_P)$ is the most preferred answer set with respect to $C \cup P$.*

Example 9. Let us consider again program P from Example 7 and also the ordered list of atoms $C = [b, c]$. Then, $P \cup r_C = \{a \leftarrow, \quad c \leftarrow \neg b, \quad b \leftarrow \neg c, \quad \neg\neg b \times \neg\neg c \times all\_pref\}$. We can verify that $\{a, b\}$ is *the most preferred answer set of P with respect to C* and it is also an inclusion preferred answer set of $P \cup r_C$.

In the following section, we show how to compute the preferred answer sets for ELPOD's using `psmodels` [3].

4 Computing preferred answer sets for extended ordered programs

`Psmodels`[2] is a software implementation useful to obtain the inclusion preferred answer sets for the ordered disjuntion programs as defined in [3]. Hence, it is not possible to obtain the inclusion preferred answer sets for extended ordered programs using `Psmodels`. The reason is that the definition given by Brewka for ordered disjunction has syntactical restrictions. However, in particular when this program has extended ordered rules using negated negative literals, we can translate easily this program to a standard ordered disjunction program (as defined by Brewka in [3]) and in this way to use `Psmodels` to obtain the preferred answer sets.

[2] http://www.tcs.hut.fi/Software/smodels/priority/

Lemma 2 allows us to translate an extended ordered program that results from joining a normal program with an extended ordered disjunction rule with negated negative literals to a standard ordered program.

In the following definition and lemma the atoms $a^\bullet$, a°, are atoms that do not occur in the original program P.

Definition 9. *Let $\neg\neg a$ be a negated negative literal. We define the associated set of $\neg\neg a$ as follows:*
$R(\neg\neg a) := \{ \leftarrow \neg a, a^\bullet , \quad a^\bullet \leftarrow \neg a^\circ ,$
$a^\circ \leftarrow \neg a , \quad \leftarrow a, a^\circ \}.$

Lemma 2. *Let P be a normal program and let $C = [c_1, c_2, \ldots, c_n]$ be an ordered set of atoms such that $C \subseteq \mathcal{L}_P$. Let $C^\bullet = \{c_1^\bullet, c_2^\bullet, \ldots, c_n^\bullet\}$ be a set of atoms such that $C^\bullet \cap \mathcal{L}_P = \emptyset$. Let r_C be the extended ordered rule defined from C. Let $r_C^\bullet$ be the following ordered rule $c_1^\bullet \times c_2^\bullet \times \ldots \times c_n^\bullet \times all\_pref$. Let $A = \bigcup_{c_i \in C \text{ and } 1 \leq i \leq n} R(\neg\neg c_i)$. Then M is an inclusion preferred answer set of $P \cup \{r_C^\bullet\} \cup A$ iff $M \cap \mathcal{L}_P$ is an inclusion preferred answer set of $P \cup \{r_C\}$.*

Example 10. Let us consider again the program P_1 at the beginning of this Section 3, and the set of atoms $C = [f, c]$ then $r_C = \neg\neg f \times \neg\neg c \times all\_pref$,
$A = \{\leftarrow \neg f, f^\bullet, \quad f^\bullet \leftarrow \neg f^\circ, \quad f^\circ \leftarrow \neg f, \quad \leftarrow f, f^\circ,$
$\leftarrow \neg c, c^\bullet, \quad c^\bullet \leftarrow \neg c^\circ, \quad c^\circ \leftarrow \neg c, \quad \leftarrow c, c^\circ\}$ and $r_C^\bullet = \{f^\bullet \times c^\bullet \times all\_pref\}$.

Then, by running `psmodels` we obtain the following inclusion preferred answer set of the standard ordered program $P \cup r_C^\bullet \cup A$: $\{a, c, c^\bullet, f^\circ\}$. Finally, we can see that the intersection of the answer set with $\mathcal{L}_P$ corresponds to the inclusion preferred answer sets of the original extended ordered program $P \cup r_C$ as we described before: $\{a, c\}$.

5 Conclusions

We review the syntax and semantics of ELPOD. We explain how to specify and compute the preferred answer sets of a logic program with respect to a preference order. This order is expressed as an extended ordered disjunction rule using double default negation. Finally, we show how to compute the preferred answer sets for ELPOD's using `psmodels`.

References

1. G. Brewka. Logic Programming with Ordered Disjunction. In *Proceedings of the 18th National Conference on Artificial Intelligence, AAAI-2002.* Morgan Kaufmann, 2002.
2. G. Brewka, S. Benferhat, and D. L. Berre. Qualitative choice logic. *Artif. Intell.*, 157(1-2):203–237, 2004.
3. G. Brewka, I. Niemelä, and T. Syrjänen. Implementing Ordered Disjunction Using Answer Set Solvers for Normal Programs. In *Proceedings of the 8th European Workshop Logic in Artificial Inteligence JELIA 2002.* Springer, 2002.

4. R. Confalonieri and J. C. Nieves. Nested logic programs with ordered disjunction. In *Latin-American Workshop on Non-Monotonic Reasoning 2010 (LANMR10)*, volume 677, pages 55–66, 2010.
5. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.
6. M. Gelfond and V. Lifschitz. Logic Programs with Classical Negation. In D. Warren and P. Szeredi, editors, *Proceedings of the 7th Int. Conf. on Logic Programming*, pages 579–597, Jerusalem, Israel, June 1990. MIT.
7. N. Leone and S. Perri. Parametric Connectives in Disjunctive Logic Programming. In *ASP03 Answer Set Programming: Advances in Theory and Implementation*, Messina, Sicily, Sept. 2003.
8. M. Osorio, J. A. Navarro, and J. Arrazola. Applications of Intuitionistic Logic in Answer Set Programming. *Theory and Practice of Logic Programming (TPLP)*, 4:325–354, May 2004.
9. M. Osorio and M. Ortiz. Embedded Implications and Minimality in ASP. In M. H. U. G. Dietmar Seipel and O. Bartenstein, editors, *15th International Conference on Applications of Declarative Programming and Knowledge Management. INAP 2004*, Postdam, Germany, Mar. 2004.
10. M. Osorio, M. Ortiz, and C. Zepeda. Using CR-rules for evacuation planning. In G. D. I. Luna, O. F. Chaves, and M. O. Galindo, editors, *IX Ibero-american Workshops on Artificial Inteligence*, pages 56–63, 1994.
11. D. Pearce. Stable Inference as Intuitionistic Validity. *Logic Programming*, 38:79–91, 1999.
12. D. van Dalen. *Logic and Structure*. Springer, Berlin, second edition, 1980.

Another implementation of the p-stable semantics, a parallel aproach

David López[1], Gabriel López[1], Mauricio Osorio[2], and Claudia Zepeda[2]

[1] Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
[2] Universidad de las Américas - Puebla,
CENTIA

(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. In this paper we review some theoretical results about the p-stable semantics, and based on that, we design some algorithms that search for the p-stable models of a normal program. An important point is that we propose algorithms that can also be used to compute p-stable models in a parallel approach. **Keywords** non-monotonic reasoning, p-stable, parallel.

1 Introduction

Currently, is a promising approach to model features of commonsense reasoning. In order to formalize NMR the research community has applied monotonic logics. In [3], Gelfond and Lifschitz defined the stable model semantics by means of an easy transformation. The stable semantics has been successfully used in the modeling of non-monotonic reasoning (NMR).

Additionally, Pearce presented a characterization of the stable model semantics in terms of a collection of logics in [12]. He proved that a formula is "entailed by a disjunctive program in the stable model semantics if and only if it belongs to every intuitionistically complete and consistent extension of the program formed by adding only negated atoms". He also showed that in place of intuitionistic logic, any proper intermediate logic can be used. The construction used by Pearce is called a weak completion.

In [9], a new semantics for normal programs based on weak completions is defined with a three valued logic called G'_3 logic. The authors call it the P-stable semantics. In [7], the authors define the *p-stable semantics* for disjunctive programs by means of a transformation similar to the one used by Gelfond and Lifschitz in their definition of the stable semantics. The authors also prove that the p-stable semantics for disjunctive programs can be characterized by means of a concept called weak completions and the G'_3 logic, with the same two conditions used by Pearce to characterize the stable semantics of disjunctive programs, that is to say, for normal programs it coincides with the semantics defined in [9].

In fact, a family of paraconsistent logics studied in [7] can be used in this characterization of the p-stable semantics.

In [8], the authors offer an axiomatization of the G'_3 logic along with a soundness and completeness theorem, i.e., every theorem is a tautology and vice-versa.

Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 221-228

We also remark that the authors of [7] present some results that give conditions under which the concepts of stable and p-stable models agree. They present a translation of a disjunctive program D into a normal program N, such that the p-stable model semantics of N corresponds to the stable semantics of D when restricted to the common language of the theories. Besides, they show that if the size of the program D is n then the size of the program N is bounded by An^2 for a constant A. The relevance of this last result is that it shows that the p-stable model semantics for normal programs is powerful enough to express any problem that can be expressed with the stable model semantics for disjunctive programs.

It is important to mention that the p-stable semantics, which can be defined in terms of paraconsistent logics, shares several properties with the stable semantics, but is closer to classical logic. For example, the following program $P = \{a \leftarrow \neg b, a \leftarrow b, b \leftarrow a\}$ does not have stable models. However, the set $\{a, b\}$ could be considered the intended model for P in classical logic. In fact, it is the only p-stable model of P.

In [11], a schema for the implementation of the p-stable semantic using two well known open source tools: Lparse and Minisat is described. In [11], a prototype[3] written in Java of a tool based on that schema is also presented. In [6] the author presents an improved implementation [4] using optimized code and algorithms, resulting in an error-free tool, getting at least the same afficienty that [11]. In [5] there is another implementation where the author manage some work through the use of a high performance and efficiency software Suite call Potassco[5], coded in java without a final version.

Currently one processor computers had reached enormous speeds and they have pushed hardware to its physical limits. This trend has gradually been displaced given the physical limits in design delimiting the computational power we can get with a single processor. Multiple architectures are known that use parallel through the interaction of multiple processing units.

One of these takes place in the form of various processors working together internally which was an initial progress. With advances in computer networks, new protocols and speeds reached by them, it has been created new methods for developing parallelism using small autonomous networks functioning as parallel systems. To achieve the greatest benefits of parallelism, both designers and programmers must understand the virtues and disadvantages associated with the development on multicore systems.

Despite the large advances observed in sequential Answer Set Solving technology as seen in the beginning, only a few implementations have been developed with a parallel approach[13][1][2]. Its alarming, seeing how easily we can have access to devices such as clusters, multiprocessor and/or multicore computers. It is worth mentioning that the development of these parallel systems has been made following the standards of stable semantics, which we know its extent.

[3] http://cxjepa.googlepages.com/home
[4] http://sites.google.com/site/computingpstablesemantics/downloads
[5] http://potassco.sourceforge.net/

This is the reason for us to implement a version that can parallelize the case of p-stable.

Our paper is structured as follows. In section 2, we summarize some definitions, logics and semantics necessary to In section 3, we show how to find the the p-stable models of a program P, in section 4 we desribe the current implementation workflow, in section 5 we introduce the parallel algorithms proposed. Finally, in section 6, we present some conclusions and further work.

2 Background

In this section we summarize some basic concepts and definitions necessary to understand this paper.

2.1 Syntax

A signature $\mathcal{L}$ is a finite set of elements that we call atoms. A *literal* is either an atom a, called *positive literal*, or the negation of an atom $\neg a$, called *negative literal*. Given a set of atoms $\{a_1, ..., a_n\}$, we write $\neg\{a_1, ..., a_n\}$ to denote the set of atoms $\{\neg a_1, ..., \neg a_n\}$. A *normal* clause or *normal* rule, r, is a clause of the form

$$a \leftarrow b_1, \ldots, b_n, \neg b_{n+1}, \ldots, \neg b_{n+m}.$$

where a and each of the b_i are atoms for $1 \leq i \leq n+m$, and the commas mean logical conjunction. In a slight abuse of notation we will denote such a clause by the formula $a \leftarrow B^+(r) \cup \neg B^-(r)$ where the set $\{b_1, \ldots, b_n\}$ will be denoted by $B^+(r)$, the set $\{b_{n+1}, \ldots, b_{n+m}\}$ will be denoted by $B^-(r)$, and $B^+(r) \cup B^-(r)$ denoted by $B(r)$. We use $H(r)$ to denote a, called the head of r. We define a *normal program* P, as a finite set of normal clauses. If for a normal clause r, $B(r) = \emptyset$, $H(r)$ is known as a *fact* . We write $\mathcal{L}_P$, to denote the set of atoms that appear in the clauses of P.

2.2 Semantics

From now on, we assume that the reader is familiar with the single notion of *model*[4]. In order to illustrate this basic notion, let P be the normal program $\{a \leftarrow \neg b.,\ \ b \leftarrow \neg a.,\ \ a \leftarrow \neg c.,\ \ c \leftarrow \neg a.\}$. As we can see, P has five models: $\{a\}$, $\{b, c\}$, $\{a, c\}$, $\{a, b\}$, $\{a, b, c\}$.

Now we give the definition of p-stable model semantics for normal programs.

Definition 1. *[10] Let P be a normal program and M be a set of atoms. We define the reduction of P with respect to M as $\boldsymbol{RED}(P, M) = \{a \leftarrow B^+ \cup \neg(B^- \cap M) | a \leftarrow B^+ \cup \neg B^- \in P\}$.*

Definition 2. *[10] A set of atoms M is a p-stable model of a normal program P iff $\boldsymbol{RED}(P, M) \models M$, where the symbol $\models$ means logical consequence under classical logic semantics. The set of p-stable models of P is denoted by $PS(P)$.*

We say that two normal programs P and P' are equivalent if and only if they have the same set of p-stable models, this relation is denoted by $P \equiv P'$.

The following definition states the definition of stratification and module of a program P.

Definition 3. *Let P be a normal program which can be partitioned into the disjoint sets of rules $\{P_1, ..., P_n\}$. Let $P_i, P_j \in \{P_1, ..., P_n\}, P_i \neq P_j$, we say that $P_i < P_j$ if $\exists r \in P_j : \exists r' \in P_i : H(r') \in B(r)$, if from this condition we do not conclude that $P_i < P_j$ or $P_j < P_i$ then we can choose to hold whether $P_i < P_j$ or $P_j < P_i$ as long as the following properties hold. For every $X, Y, Z \in \{P_1, ..., P_n\}$, the strict partial order relation properties and the totality property hold:*

1. *$X < X$ is false (this property holds trivially).*
2. *If $X < Y$ then ($Y < X$ is false).*
3. *If ($X < Y$ and $Y < Z$) then $X < Z$.*
4. *$P_1 < ... < P_n$*

then we refer to this partition as the stratification of P, sometimes we will write it as $P = P_0 \cup ... \cup P_n$. And we will refer to $P_i, 1 \leq i \leq n$ as a module of P.

2.3 Parallel Computing

Two types of information flow through a processor: instructions and data. The instruction stream is defined as a sequence of instructions by the processing unit. The data flow is defined by the exchange of data between memory and processing unit, according to Flynn's taxonomy any of the two streams can be individual or multiple, given these configurations Flynn proposed the following categories to identify a system:

- A block of instructions and a data stream (SISD)
- A block of instructions and multiple data streams (SIMD)
- Multiple blocks of instructions and a data stream (MISD)
- Multiple blocks of instructions and multiple data streams (MIMD)

Parallel processing can occur in SIMD or MIMD architectures.

Models of parallel algorithms There are different paradigms of parallel programming, below we name yhe ones with relevance to this article.

Work Pool Model

The *work pool* or the *task pool* model is characterized by a dynamic mapping of data or tasks among processors, emulating a list of available tasks where virtually each processor can work with any source of the list.

Master-Slave Model

Also known as the *boss-worker* model, it is said that one or more teachers distribute the flows between the different slaves, on a random or a-priori way.

3 Computing the p-stable models

Now we present the implementation of a p-stable model solver.

To find the p-stable models of a program P we can first apply the transformations to P, however the application of the transformations is not absolutely necessary nor sufficient to find the p-stable models of P. In this section we start presenting the application of the transformations, and then we give two approaches to find the p-stable models of P, both following the theorem 1 which is also presented in this section, each one with a different model of parallel algorithms.

In most cases the application of the transformations is not enough to find a p-stable model of a normal program, and other techniques are required. One of those techniques is to partition the program into sets of rules called modules. Those modules are created based on its graph of dependencies [5][6].

Theorem 1. *[10] Let P be a normal logic program, and M a model of P with stratification $P = P_1 \cup P_2$, then $\boldsymbol{RED}(P, M) \models M$ iff $\boldsymbol{RED}(P_1, M_1) \models M_1$ and $\boldsymbol{RED}(P_2', M_2) \models M_2$ with P_2', M_1, and M_2 defined as follows: $M = M_1 \cup M_2$, $M_1 = h(P_1, P) \cap M$, $M_2 = h(P_2, P) \cap M$, and P_2' is obtained by transforming P_2 as follows:*

1. *Removing from P_2 the rules r' such that $B^-(r') \cap M_1 \neq \emptyset$ or $B^+(r') \cap (h(P_1, P) \setminus M_1) \neq \emptyset$, obtaining a new program P_2''.*
2. *For every $r \in P_2''$, removing from $B(r)$ the occurrences of the atoms in $h(P_1, P)$, obtaining P_2'.*

In other words M is a p-stable model of P iff M_1 is a p-stable model of P_1 and M_2 is a p-stable model of P_2', where P_2' is obtained by removing from P_2 the occurrences of the atoms in $h(P_1, P)$ according to the theorem 1. If P can be stratified as $P = P_1 \cup ... \cup P_n$, $n > 2$, then $P = P_1 \cup Q$ with $Q = P_2 \cup ... \cup P_n$ is also an stratification of P that has only two modules, and then we can apply the theorem 1.

4 Workflow of implementation

Now that we have described the basics for obtaining p-stable models of a program p, it is necessary to explain the workflow followed by our implementation and the part which can be parallelized.

This time we use an ordered list of processes to show the workflow, a brief description of each one may get you to understand the funtion of the implementation.

1. *Reading of logic program p.* We use the lparse format due to its compatibility with a large amount of systems.
2. *Grounding.* Necessary to make the work easier, in our application we use GrinGo from Potassco suite. This gets us a program with no variables just facts.

3. *Transformations.* As deffined before, those are made to reduce the program lenght.
4. *Stratification if possible.* Our parralel approach only gets necessary if the program can be stratified.
5. *Model generation.* Using ClaspD from the Potassco suite, we generate candidates for the program.
6. *Model verification.* This part needs a longer explanation given its importance for parallel application.
7. *Showing results.*

4.1 Model verification

After performing the analysis to the implementation we decided to perform the parallelization of the part that makes the verification of the models, since in this part of the code shows that there is a sequential process where there are no strict dependencies when performing checks. Once again we show the process of model verification through an ordered list.

1. *Receiving stratifications and proposed models of program P* We use an stack were the stratifications are stored.
2. *Test the feasibility of each stratification and its model assigned* This step repeats until each stratification and its model have been analised or when one isnt satisfactible. Here we can easily see a natural parallelizable code.
3. *Analysis of results*

5 Parallel algorithms

From theorem 1,and the analisys of the model verification code two different approaches to compute the p-stable models of a program P in parallel are proposed. The first one uses an hybrid work-pool schema as we can see below *Algorithm 1.* The second one uses another hybrid using as a base the master-slave paradigm *Algorithm 2.*

6 Conclusions and further Work

The development of hybrid algorithms based on the paradigms of the workpool and master-slave brings great benefits when parallelized, since both approaches have a natural way to be implemented the first allows each node look for work in a list, which allows a more dynamic data manipulation; the second approach is classic, but on examination it was found that the master should have a well constructed load balancer to avoid becoming a bottleneck when allocating tasks, since in many cases segments of P are very small and may cause lost of time during communications. Currently the basic application its made in Java, and its being migrated to C++ in order to use MPI [6] to run the application in parallel plataforms such a cluster and a multiprocessor server getting faster services natural to C applications.

[6] http://www.open-mpi.org/

Algorithm 1 Work_pool_check

Require: Pi and Mi {Pi are the stratiffied elements of P and Mi their proposed models}
Ensure: Shows if Mi are valid {Otherwise it finish early}
  while Board.hasElements= $TRUE$ and Board.status=NoError **do**
    setPtemporal(null);
    setMtemporal(null);
    findPi() and findMi() {on board}
    setPtemporal(Pi);
    setMtemporal(Mi);
    deleteFromBoard(Pi, Mi);
    Pi.setStatus(pendant);
    if testFactibility(Pi)= $TRUE$ **then**
      Pi.setStatus(factible);
    else
      Board.status=Error;
    end if
  end while

Algorithm 2 Master_Slave_check

Require: Pi and Mi {Pi are the stratiffied elements of P and Mi their proposed models}
Ensure: Shows if Mi are valid {Otherwise it finish early}
  if amIMaster= $TRUE$ **then**
    while PiList.hasElements= $TRUE$ and PiList.status=NoError **do**
      checkPiList();
      giveWorkToNodes(); {free nodes receive Pi and Mi sets}
      waitResponseFromNodes();
      updatePiList();
    end while
    giveEndNoticeToNodes();
  else
    while moreWorkToCome= $TRUE$ **do**
      waitWorkFromMaster();
      verifyPiMiFactibility();
      sendResultsToMaster();
    end while
  end if

References

1. M. Balduccini and E. Pontelli. Issues in parallel execution of nonmonotonic reasoning systems. In *Parallel Computing*, number 31 in 6, pages 608–647, 2005.
2. E. Ellguth and M. Gebser. A simple distributed conflict-driven answer set solver. In *Logic Programming and Nonmonotonic Reasoning*, volume 5753 of *Lecture Notes in Computer Science*, pages 490–495. Springer Berlin / Heidelberg, 2009.
3. M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.
4. J. W. Lloyd. *Foundations of Logic Programming.* Springer, Berlin, second edition, 1987.
5. G. López. Solver para p-stable. To be publish at BUAP.
6. A. Marín. Algoritmos para semánticas de programación lógica. Master's thesis, BUAP, 2011.
7. M. Osorio, J. Arrazola, and J. L. Carballido. Logical weak completions of paraconsistent logics. *Journal of Logic and Computation, doi: 10.1093/logcom/exn015*, 2008.
8. M. Osorio and J. L. Carballido. Brief study of G'$_3$ logic. *Journal of Applied Non-Classical Logic*, 18(4):79–103, 2008.
9. M. Osorio, J. A. Navarro, J. Arrazola, and V. Borja. Logics with common weak completions. *Journal of Logic and Computation*, 16(6):867–890, 2006.
10. S. Pascucci. Syntactic properties of normal logic program under pstable semantics: theory and implementation. Master's thesis, March 2009.
11. S. Pascucci and A. Lopez. Implementing p-stable with simplification capabilities. *Submmited to Inteligencia Artificial, Revista Iberoamericana de I.A.*, Spain, 2008.
12. D. Pearce. Stable Inference as Intuitionistic Validity. *Logic Programming*, 38:79–91, 1999.
13. E. Pontelli and M. Balduccini. Non-monotonic reasoning on beowulf platforms. In Springer, editor, *PADL '03 Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages*, pages 37–57, 2003.

Armin: Automatic Trance Music Composition Using Answer Sets Programming

Flavio Omar Everardo Pérez
Universidad de las Américas Puebla
Cholula, Puebla
flavio.everardopz@udlap.mx

(Paper received on November 28, 2010, accepted on January 28, 2011)

Abstract. The Artificial Intelligence (AI) has taken a leading role in many activities which used to make "by hand", one of them is the musical composition. Such task now has another alternative implementation, through the support of software that can compose different musical genres to the point where a computer can compose pieces with some autonomy. This paper proposes Armin, a system dedicated to the electronic trance music composition. Armin's aim is to provide a trance music composition to serve as a template or a base audio file, ready to add more instruments in a mastering (remastering) production ---Additionally, it seeks to enable greater collaboration between a machine and a human, both seen as composers.

Keywords: Armin, Automatic Music Composition, Algorithmic Compositon, Answer Sets Programming, Trance Music, Artificial Intelligence.

1 Introduction

The Computer science and mathematics provide a large repertoire of algorithms that can be used to generate and process musical material [6] we mean, it is possible to automate certain musical genres based on knowledge obtained through the application rules of its musical components. In this paper we show that it is possible to use answer sets solvers for the composition of short pieces in the trance genre. Armin is named in honor to the great sympathy felt by the author with Armin van Buuren, who stands out as Disk Jockey (DJ), producer inside the trance genre, founder and owner of Armada Music record label. Armin is a system based on Anton [2] with some knowledge of the rules according to trance music, which is able to make a plan and its execution of a etude using Answer Sets Programming (ASP) [10] and at the same time assist the user in the musical production. In addition we show the models inside the trance music and the ASP application in a genre which is characterized by percussive rhythms to set the pace over time. Also we are interest to answer in the future questions like: How much music can be automated? Is there a limit in the music automation? How do you measure the music automation? Is the music automation determined by a number of instruments or if is related to the duration of the piece? During this paper we will talk more deeply about these questions exploring the author's and literature opinions.

2 Background inside the Computer-aided Composition

The Computer-aided composition (CAC) [1] has taken a very important role in composition and musical production, from music pieces which have been created from synthetic instruments (Sound Synthesis) which give lifelike sounds and a wide variety of sound effects to the automatic scores composition. All this avoids the need to go to a studio and record each instrument like before.

(C) C. Zepeda, R. Marcial, A. Sánchez
J. L. Zechinelli and M. Osorio (Eds)
Advances in Computer Science and Applications
Research in Computing Science 53, 2011, pp. 229-237

The CAC in the same way is known as algorithmic composition. This provides the power to map an idea to software with the options to create and edit the music contents before rendering the final song.
Musicians and scientists have been studied to understand exactly how humans compose music [12] and that have raised questions as “where is the next note coming from?” [3], but the process of selecting the next note is difficult to explain and maybe has some personal taste. Similarly the questions arise once again. If we have already chosen a note, how long must it last? Not only are these questions which we must find the answer, also there are certain considerations to observe in detail because they may pose a problem in the CAC.

2.1 Considerations of Computer-Aided Composition

Music is a finite art where eventually all possible combinations of time and sounds will be explored. A software tool shares the same property, is finite due to the knowledge that counts inside. Although it is possible to get a lot results without repeating any, but we need to think on flexibility and scalability of the system.

One of the difficulties in the CAC is when the "power" is given completely to the tool to produce a complete piece, two cases may occur as mentioned in [12]. Create or imitate. Imitation is given to perceive a result (in this case musical) similar to any other known to us whom we can identify. What would be worth asking is: What rules allow such behavior? Similarly goes the question how much is piece similar to another? Or how many and which notes should vary to conclude that there is a known piece by us? One solution is to find a way get deterministic results by certain rules to provide additional knowledge and maybe some rules that crosses the music boundaries.
With this proposal it is possible to explore a very large amount of results and some fragments that may lead to a "never heard before" by humans or ever even imagined by us or by the pioneers of trance music.

3 Inside the Trance Music

Trance is a genre of the electronic music with a percussion frequency between 130 and 140 Beats per Minute (BPM). The melodies are long and they change or evolve over time. One of the main features of trance is that its time signature is four quarters (4/4) which specifies 4 pulses (beats) per measure. Each beat is identified by a kick (Bass Drum or BD), possibly accompanied at times 2 and 4 with snare sounds (SD) or claps.

Another feature lies in the change of pace that is usually done every two, four or eight bars. However, it is possible that rather given a change of pace, just add or discard drum sounds or other minor instrument. This is what is called progression. Throughout the song we get to hear that the rhythm changes are often strategically placed in certain places and not so random (stochastic) and it is said that the rates are increasing in intensity compared to the previous (also progression).

There are moments in songs where there are no beats, in other words the percussions stops to give importance to synthetic sounds, long chord and a

possibly slower paces, this part of the music is known in trance as breakdowns or breaks. These breaks are made to give a break to the beat loops, or to change in an attractive way the rhythm of the song. The beats will later returns to resume the song's tone. A trance song structure may differ from the length of the same, but the essence remains between different versions of the song. Currently the major rankings of electronic music included in its top trance DJs as Armin van Buuren, DJ Tiësto, Markus Schulz, Andy Moor and Paul Van Dyk, to say some. Today the trance sounds has evolved and have achieved a high degree of sound perception and polyphony.

4 Anton

Anton[1] [2] is a melodic, harmonic and rhythmic composition system which uses ASP to generate short pieces. Anton exists in versions 1.0.0, 1.5.0 and 2.0.0 and their main tasks are:

- Compose
- Diagnosing errors
- Completing a piece

4.1 Composition

Anton offers both melodic and harmonic and also rhythmic composition and is available for the implementation up to four voices (solo, duet, trio and quartet). Anton works in the style of "Rules Palestrina" from Renaissance music in the modes: major, minor, Dorian, Phrygian, Lydian and Mixolydian.

4.2 Diagnosing errors

This section allows detecting errors within a piece. This part also appears in the composition section for generating pieces musically valid or correct.

4.3 Completing a piece

Is possible insert a piece fragment in a logic programming format and select the complete piece option in Anton. If the part is complete Anton returns one solution or model. This model consists of a set of steps number given by the letter T which refers to the time at which the note N should be played in a part P.

The complete system consist of three major phases; building the program, running the solver and interpreting the results. Finally there is an option for playing the audio file generated by Csound at the end of the execution.

5 Answer Sets Programming

ASP is a declarative programming paradigm where a software program is used to describe the requirements which must be completed successfully by solving a specific problem [2], this paradigm is based on stable models (answer set). ASP

[1] Official Website: http://www.cs.bath.ac.uk/~mjb/anton/

is a strategy that has knowledge based and constraints in order to prevent undesired executions [10].

Each instrument both melodic and percussion consist in a set of rules which allow us to model some part of a musical piece as shown in Figure 1 and thereby determine their behavior over time. The code inside the Figure 1 is an ASP code and the main feature in here is that plays the mandatory hits at the times 0, 4, 8, and 12 like a common or standard trance rate (counting from 0 to 15). In addition there are 12 possible hits in a sixteenths measure so the other 12 hits should be decided if the hit is made or not, if the hit is done, take amplitude from 4 to 6 and play it. If the stroke is not carried out, it is equivalent to mention that the amplitude of the hit is zero.

```
%%default bassDrum Amplitude = 8
bassDrumAmplitude(8).

%%mandatory BD hits for trance music
mandatoryHit(0;4;8;12).

%% if is mandatory hit... get the amplitude and play it
playDrum(X,A) :- bassDrumAmplitude(A), mandatoryHit(X).

%% pick an amplitude and a non mandatory hit...
1{ playDrum(Y,X) : amplitude(X) }1 :- bassDrumHits(Y).
```

Figure 1 Fragment of Armin Bass Drum Generator

The solver and grounder used for this system is Clasp [7] and Gringo [8] respectively, taken from the Potassco Project Site[2]. Both implementations are currently leading the ASP area.

6 Armin – System Description

Armin[3] is a trance music composition algorithm which takes the composition rules represented declaratively using ASP. These rules have the property to model a desired result within the genre and to explore future extended compositions. Using in part the operation of the Anton for melodic composition, Armin provides the feature to expand into the main percussion and musical sections chaining, like introduction to verse, verse to chorus, chorus to breakdown... in order assemble these sections with some dependency to generate the next state (Markov Chains). One feature that is sought is the ability to perform a "valid" musical composition inside the trance genre with its dependencies of the internal parts of a traditional song and percussion and its respective progression between the different sections. It should be noted that the objective of Armin in its first version is not fully compose a piece of the trance music, but to serve as a "template" both melodic and percussive and other sounds and lately make some post production to this source. Additionally

[2] Potassco official Website: http://potassco.sourceforge.net/

[3] Armin official Website: Flavio Everardo: http://flavioeverardo.com in the Automatic Trance Music Composition section.

it seeks to explore different outcomes (each answer a set corresponds to a valid piece of music) that have openness within the music style and looking forward to propose variations in its composition completely valid. To generate a new piece of music just ask for a new model or answer set.

6.1 Armin – Architecture and Components

The Armin architecture is divided into two big sections of ASP as shown in Figure 2 which are the Score Generator and the Sounds Generator. It is important to add that there is a strong dependency between these two parties, because the Score's product works as the input of the Sound Generator.

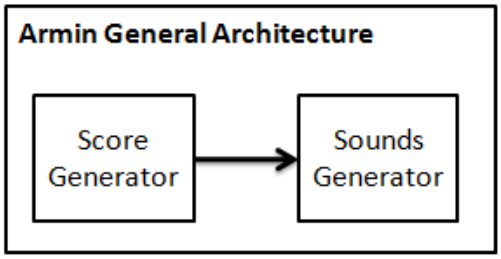


Figure 2 Armin Architecture

The Score Generator consists on a driver and the assembly file as shown in Figure 3. The Armin driver is in charge to manage the tasks during the score execution and is the main file which receives some input parameters, which are:

- BPM – valid values given inside the interval of 130 to 140 including those.
- Fundamental (for melodic instruments) - any of the 12 notes that conforms the chromatic scale.
- Parts / Fragments - this is the number of parts or sections that will be played in the piece. This value is variable depending on the length of the piece that the user wants to generate. The fragments correspond to the parts of a trance song (intro, verses, choruses, breakdowns ...).

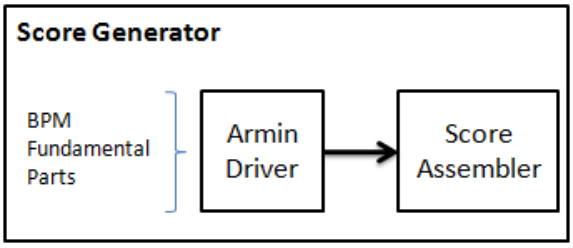


Figure 3 Score Generator

The Score Assembler is an ASP file that generates the order and the appearance frequency of the parts during the resulting song. These parts as mentioned before correspond to the internal structure of a piece. This generation is made by certain dependencies between parts of the piece and its possible behavior over time. Each answer set corresponds to one configuration of a song (this doesn't mean that every time we get the same answer, we will get the same musical result). At this point, we know what that a fragment is going to be played and also we know when is going to happen, but it has not generated any audio until now. The section in charge of this task is Sounds Generator.

Once we have got the score model of the song, this information works as input in the Sounds Generator section which consists of five components as shown in Figure 4, the components are: results interpretation, instruments generator (ASP), Csound parser, render, and finally the final product, an audio file.

The Sounds Generator starts interpreting the obtained results by the model (answer set) in the Results Interpretation section. This file besides understanding the behavior of the song is responsible for sending a call to each instrument that is going to be part of the current structure by calling the Instruments Generator section. This file is responsible for generating the model of each instrument depending on the part or section in the score. Once generated the model of each instrument at a time, is executed the Csound Parser which is responsible for mapping the results to a file in the Csound format. This process takes place until it has reached the total time of the piece or until there are no fragments remaining to be parsed. Once the file is done, is time to run the Csound's render and then Csound provides an audio file in WAV format.

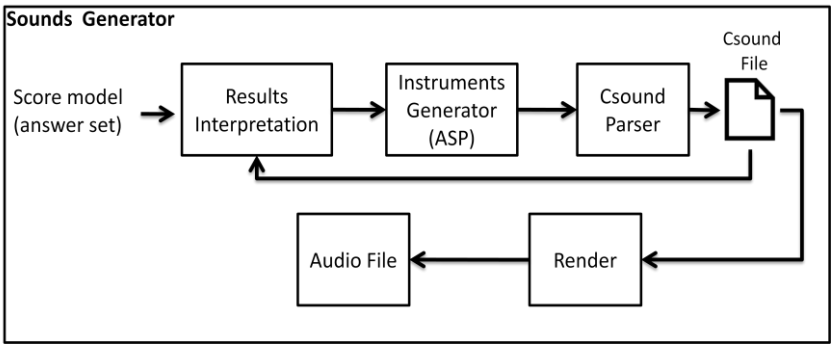


Figure 4 Sounds Generator

The way as an automatic music composition tool creates a musical piece can vary from the goal or essence of the application and in some ways what is evaluated is not the methodology but the final product (song, album, performance ...). This is always true when the objective of the application is public and not a personal domain [11]. If there is a personal reason or purpose behind the application, we can say that there is no exact methodology neither restrictions that limits the production and musical performance.

There are different approaches to the composition of a musical piece which depend on a parameter that is set as a starting or reference point for the production of the piece, such as time or duration, the number of elements within the structure of the song or even the instrument(s) chosen. Armin presents a sequential composition structure taking the particularity that the time in Csound is cumulative, and it takes by default an entry point called introduction (intro). All pieces start off with their respective intro (this doesn't mean that all the pieces will have the same behavior) at time zero and the following states are calculated using the current state. The last fragment is dedicated to the end of the song (outro) which also has its own dependency on the previous state.

For Armin architecture it is used the Potassco software (Clasp and Gringo) as the declarative language, the Perl programming language is used for the results interpretation and parsing and Csound for the audio synthesis.

7 The Automation

We believe that the music can be automated in several ways, which depends on the purpose of the software or the developers. The important question here to determine how much can music be automated is: what are you going to let the user do after delivering a musical piece? In other words what would be the composer's contributions?

The automation cannot be measure in an easy way. Many people will be only cared for the musical results and answering themselves if the system is doing what it is supposed to do. Perhaps in the future many algorithms composition systems will have to adopt a base or standard to be evaluated among them.

8 Results

Armin doesn't provide a finish work; it provides us a base (template) song, ready for adding more instruments in the way the users wants. Armin delivers a musical piece as a starting point to compose trance music. The features inside a base song are the implementation of the bass drum, hi hats, snare drum and the dependency of this instruments among the time for short pieces (30 – 60 seconds). The melodic part is currently in developing for trance rhythms. Building these short pieces it doesn't mean a problem for the system and we are looking to create a more complete template song. In addition Armin provides openness in the drums generators, respecting the principles of the trance music, but adding some new beats features to give us different kind of songs among executions. At this point because of the length of the song it is not easy to identify the elements of a trance song by hearing, this is because the beats nuances are not completely natural. What we mentioned about the Markov Chains it is currently at a conceptual level; however is in current developing by now. We have beats fragments already in audio format which you can hear and download in Armin's site[4].

9 Future work

9.1 Musical work

Inside the musical work there are a lot of things to do starting with the addition of more instruments suitable for trance music. Also, like the Computer Systems for Expressive Music Performance (CSEMP) [9] Armin can be extended to a more real and not machine performance. This can be reached by working with the nuances inside the notes and beats.

An interesting point inside the trance music would be to model different versions of a generated piece like extended ones, which have the particularity of lasting more than six or seven minutes. Besides creating the variations of a given piece would be an interesting challenge for the taste of every user.

Another feature is the rules testing. Study the ways in which a rule is acting over the others or if it's worth to change the rule or model the rule in a different way to get another and a better meaning. In addition the possibility to get more in touch with professional people inside the trance area will be very grateful. Finally add more rhythms to play with and fills on every two or four measure,

[4] Short pieces and examples can be found in http://flavioeverardo.com in the Automatic Trance Music Composition section.

this will also provide the opportunity to make more "unique" pieces among executions.

9.2 Computational work

Among the computational work we can find in a first order, the use of data mining to find more patterns inside the style and adding a plus inside the "reality" of the musical production. The second and also important will be the user's feedback in a machine learning purpose. Having the chance to "vote" or "grade" the songs quality, compositional structure, notes sequence… can be a useful knowledge for future performances.

Another computational work is the chance that the user can modify and edit every step during the execution in a graphical user interface to make more suitable the human interaction.

10 Conclusions

Music is much to be a personal taste area; this is because we perceive music in several ways. Besides, judging is a subjective process [4] and we all do not like the same music. Armin has the flexibility of creating several solutions with the same score results and of course, it can provide more results if the score model change among executions. This provides the users the capability of discover many unconsidered options by them. Adding more rules and instruments can supply us even more unexpected results and more creative pieces. Besides this allows the composers to create the moldings for his own creations [3] and styles.

Acknowledgement

We would like to thank Jorge Nava who stands out as DJ and music producer, for his involvement within this paper and for their contributions and supervision within the definition of the trance music in section 3.

References

[1] Anders, T. Composing Music by Composing Rules: Design and Usage of a Generic Music Constraint System. Ph.D thesis, Queen's University, Belfast, Department of Music (2007)

[2] Boenn, G., Brain, M., Vos, M., and Ffitch, J. 2009. ANTON: Composing Logic and Logic Composing. In *Proceedings of the 10th international Conference on Logic Programming and Nonmonotonic Reasoning* (Potsdam, Germany, September 14 - 18, 2009). E. Erdem, F. Lin, and T. Schaub, Eds. Lecture Notes In Artificial Intelligence, vol. 5753. Springer-Verlag, Berlin, Heidelberg, 542-547. DOI= http://dx.doi.org/10.1007/978-3-642-04238-6_55

[3] Boenn, G., Brain, M., Vos, M., and Ffitch, J. 2008. Anton: Answer Set Programming in the Service of Music. In *Proceedings of the Twelfth International Workshop on Non-Monotonic Reasoning* (Sydney, Australia, September 13 - 15, 2008). 85-93 http://www.cse.unsw.edu.au/~kr2008/NMR2008/nmr08.pdf

[4] Boenn, G., Brain, M., Vos, M., and Ffitch, J. 2008. Automatic Composition of Melodic and Harmonic Music by Answer Set Programming. In *Lecture Notes in Computer Science*, 2008, Volume 5366/2008, 160-174, DOI: 10.1007/978-3-540-89982-2_21 http://www.springerlink.com/content/rj422j5831w0278m/

[5] Boulanger, R. (ed.): The Csound Book: Perspectives in Software Synthesis, Sound Design, Signal Processing and Programming. MIT Press, Cambridge (2000)

[6] Dahlstedt, P. 2007. Autonomous evolution of complete piano pieces and performances. In Proceedings of the ECAL Workshop on Music and Artificial Life (MusicAL , Lisbon, Portugal, Sep.), 10.

[7] Gebser, M., Kaufmann, B., Neumann, A., and Schaub, T. 2007. Conict-Driven Answer Set Solving. In Proceedings of the Twentieth International Joint Conference on Arti_cial Intelligence (IJCAI'07), M. Veloso, Ed. AAAI Press/The MIT Press, 386{392. Available at http://www.ijcai.org/papers07/contents.php.

[8] Gebser, M., Schaub, T., and Thiele, S. 2007. GrinGo: A New Grounder for Answer Set Programming. In proceedings of the Ninth International Conference on Logic Program-ming and Nonmonotonic Reasoning (LPNMR'07), C. Baral, G. Brewka, and J. Schlipf, Eds. Lecture Notes in Arti_cial Intelligence, vol. 4483. Springer-Verlag, 266{271.

[9] Kirke, A. and Miranda, E. R. 2009. A survey of computer systems for expressive music performance. *ACM Comput. Surv.*42, 1 (Dec. 2009), 1-41. DOI= http://doi.acm.org/10.1145/1592451.1592454

[10] Lifschitz, V. 2008. What is answer set programming?. In *Proceedings of the 23rd National Conference on Artificial intelligence - Volume 3*(Chicago, Illinois, July 13 - 17, 2008). A. Cohn, Ed. Aaai Conference On Artificial Intelligence. AAAI Press, 1594-1597.

[11] Pearce, M. T., Meredith, D. and Wiggins, G. A. (2002). Motivations and methodologies for automation of the compositional process. Musicae Scientiae, 6(2), pp. 119-147.

[12] Senaratna N. I. 2006. Automatic Music Composition with ACMTIES. University of Colombo School if Computing.