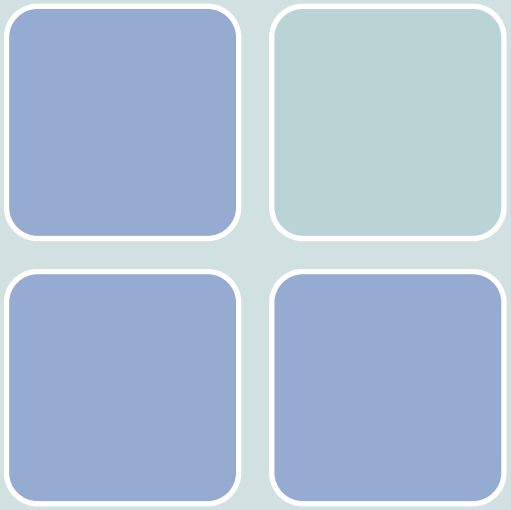


# csaw '04

## Computer Science Annual Workshop

23-24 September 2004

Editors:
Gordon J. Pace
Joseph Cordina

Department of Computer Science and AI University of Malta

# Preface

Last year, the Department of Computer Science and AI or the University of Malta, organised the first departmental research workshop — the University of Malta, Computer Science Annual Workshop 2003 (CSAW '03). We called this first workshop an *annual* one in the hope that it would induce us to organise it again in 2004. These proceedings are evidence that we have managed to live up to the workshop's name.

It is interesting to compare research presented in last year's workshop and that presented in this volume. Not only does one find development of the work being carried out by the members of staff and graduate students, but also evidence of areas of research evolving through the interaction between members of staff working in different fields. We hope that CSAW'03 has contributed to encouraging such interaction.

This year, we have extended the scope of the workshop further. We have invited finishing undergraduates with an exceptional final project to present their work. We have also invited a number of local IT companies to participate in the the workshop.

We would like to thank the CS&AI staff and research students who cooperated, making it possible to organise the workshop. Thanks also are due to Go Mobile for sponsoring the workshop, Rupert Cefai for designing the proceedings cover, and the Malta Council for Science and Technology for, once again, providing us with a venue for CSAW.

Wishing you interesting and fruitful research discussions in CSAW'04, do not forget to mark CSAW'05 in your diaries!

September 2004

Gordon Pace & Joseph Cordina
CSAW '04 Organisers

# Table of Contents

# Towards Effectively Appraising Online Stores

Ernest Cachia and Mark Micallef

Department of Computer Science and AI,
University of Malta

**Abstract.** This paper introduces research being carried out into the measurement of the quality of e-commerce systems. Considerable work has been done on software metrics in the last few decades but e-commerce specific metrics seem only applicable to already deployed systems. It is proposed that a set of metrics is needed, which can be applied from the earlier stages of E-Commerce system development to improve risk management. This paper attempts to appraise e-commerce systems by proposing a set of essential attributes for an e-commerce site to succeed. This paper also serves as groundwork for future e-commerce metrication work based on these same attributes.

## 1 Language

Electronic Commerce (e-commerce) is most often referred to as the buying and selling of products and services using the Internet. The British government broadly and completely defines e-commerce as "the exchange of information across electronic networks, at any stage in the supply chain, whether within an organisation, between businesses, between businesses and consumers, or between the public and private sectors, whether paid or unpaid"[1]. Throughout this paper, references to e-commerce systems should be taken to imply a Business-to-Consumer (B2C) type model.

Whatever definition one gives it, E-Commerce is fast becoming a popular means of purchasing almost anything you need from books to diamonds. So much so that it has become an almost discounted fact for a modern-day business to provide its goods/services online. Research reveals that in 2003 online retail sales in the US again jumped by 29.1% from $13.8 billion in 2002 to $17.8 billion in 2003. To put things in perspective, in 1999 (only) $5.4 billion were spent online in the US[2]. Similar trends have been observed in Europe.

With this in mind, focus is naturally drawn to the quality of IT systems used to facilitate commercial transactions, and more specifically the quality of e-commerce systems. The issue of classifying e-commerce systems as being of definable quality can be approached from two aspects: the technical and the business aspects. The technical aspect deals with how such systems actually work. The business aspect is more related to products/service handling. Bearing in mind that an e-commerce initiative (venture) is made up of a sound business model and technical innovation, both technical and business aspects of e-commerce must go hand-in-hand in order for an e-commerce venture to succeed. Although both the technical and business aspects are important for the success of an e-commerce venture, this paper will focus on technical issues.

In order to substantiate this argument, results of various research exercises have been utilised, including a survey which was carried out by the authors of this paper amongst 350 regular e-commerce users.

## 2 How is E-Commerce System Measurement Different?

Apart from the six generic software quality attributes as set out by ISO-9126[6], E-Commerce systems contain certain attributes which would seem to feature more strongly in them than in more generic software systems[10].

Since the first software metrics appeared in the 1970s, new ones have been developed as new technologies surfaced (e.g. Chidamber and Kemerers metrics suite for object oriented design[9]). It is the opinion of this paper's authors, that e-commerce systems have sufficient unique characteristics to merit their own e-commerce metrics and measurements suite. This is not to say that conventional attributes need not be present in e-commerce systems. On the contrary, they maintain their fundamental importance.

Distinguishing factors of e-commerce systems are highlighted in bold text throughout the remainder of this section.

First of all, e-commerce systems are largely content-driven. Customers log on looking for information about products/services, be it a simple price and description, a very detailed technical specification, or the terms of purchase. Enabling customers to effectively browse through a myriad of products and providing them with exactly all the information they need can be a challenging task, especially when considering that different customers may have different needs. Issues arise as to how to organize content in a system, how to prioritize it, how to allow users to navigate through content and so on. Clearly, navigability would be an important attribute to consider when appraising an e-commerce site.

More than any other type of software applications, e-commerce systems are exposed to the world. Given the open nature and main intent of the Internet many aspects of the Internet can work against the security interests of an e-commerce website[10]. This highlights two other important attributes: security and privacy. In most cases, a customer will trust an online vendor with personal information with the proviso that his/her personal data is protected and will not be misused in anyway. A betrayal of this trust, whether intentional or not, could have serious negative repercussions on the vendor.

Another distinguishable feature of e-commerce systems is that they are mostly browser-based. The HTTP protocol is not so user-friendly when compared to the event-driven programming that most of us are used to when developing user-interfaces. Functionality that we have grown accustomed to in other paradigms present a much tougher challenge. Scripting languages such as JavaScript™ and more recently the emergence of the Microsoft .Net™ framework attempt to tackle this problem but then again, support for these technologies is not the same on all browsers[11]. This presents problems relating to usability and portability of web systems.

A site becoming popular will generally translate to an increase in profits to a vendor but there is another side to having an enormous user base. It should be ensured, that the site performs as well with a million hits a day as it would with 1000 hits a day. As Deters[12] puts it, having fast and dependable access to the most relevant information available is of the utmost importance in a competitive information-oriented society. Therefore, performance and scalability become key issues. The research presented later in this paper indicates that only 18% of users are likely to remain unconditionally loyal to an e-commerce site after its performance degrades due to increased popularity. Another problem associated with having a large number of hits is the problem of portability. The higher the number of hits experienced by an e-commerce site, the higher the chances are that the given site is being accessed from different devices, operating systems, and browsers. This can cause problems especially if a site is using technologies that are not universally implemented or uniformly rendered in different environments. Also, having a large customer-base poses a problem with defining a mechanism for customer feedback.

Lastly, e-commerce systems are likely to change quite often. Whether it be changing site content, site aesthetics or even site functionality. Just like in any software system, changes to a site will introduce additional risks of failure thus affecting its reliability. Clearly, a change in site functionality carries more risk than a change in content. However, even a simple change in website content can bring with it layout problems (text too long, image of an incorrect size or missing, etc.) potentially causing a deterrent to new customers. Developers should make sure that a site ages well, indeed matures, as changes are implemented. This is reflected in the generic software attribute of maintainability.

## 3 Survey Design

A survey can be a powerful tool to figure out what your market needs and how you can market to them[18]. The main raison d'tre for online stores is to be sold so-to-speak to everyday online shoppers. Therefore it was deemed imperative at this stage to elicit and highlight online shopper opinion.

On the basis of the e-commerce quality attributes identified above, a survey was designed to help obtain a user perspective on the issues involved in e-commerce systems appraisal. The survey was divided into two sections. The first section focused on collecting information about the participants that would later help filter results and identify sub-trends according to certain criteria (e.g. age, education level, etc). The second section was designed to tap into the participants view on the quality of e-commerce systems.

Based on the discussion in section 2 of this paper, the following attributes were felt to be relevant regarding e-commerce systems:

- Security and Privacy
- Portability
- Performance and Scalability
- Navigability, Usability and Aesthetic Features
- Multi-lingual Features
- Low-Bandwidth version of sites
- Reliability

The questionnaire was therefore designed to elicit information relevant to the above attributes. It should be noted, that due to paper length requirements, it was decided not to include the explanation and justification regarding the structure and choice of questions in the survey. However the actual questionnaire together with supporting explanatory documentation can be accessed at http://www.cs.um.edu.mt/∼mmica/survey

## 4 Results

This section will discuss results from the survey and propose a set of attributes which should be deemed essential in an e-commerce system in order for it to be considered a quality system.

Please note, that some figures collectively amount to more than 100% because users where offered the possibility to make multiple choices.

### 4.1 General Observations

One of the first steps taken when analyzing the data was to analyze the user sample. Just over 95% of participants claim to use Internet Explorer™ whilst 7% use Netscape™. Other browsers compare poorly. Also, the Microsoft Windows™OS family seems to be the most popular amongst our sample with over 98% of users using Windows™. Linux/Unix come in second with 8.92%. With regards to device usage, the desktop PC claims the top spot with 92.9% of users using desktop PCs for e-commerce transactions. 24% use laptops whilst mobiles and PDAs trail with 5.23% and 1.85% respectively. These figures compare well with other usage surveys that have been carried out[13][14].

Regarding demographics, 94% of participants were under 50 years old and the most popular items bought online are books with 80% of users having bought books online. Consequently, other popular purchases ranked as follows: Software (32%), hardware (29.6%) and music (29.3%). 71.9% of users claim they would be negatively affected had e-commerce not been available to them.

Disturbingly, 77% of users claim to have abandoned transactions mid-way through. The top reasons for this were stated as:

- User decided (s)he did not want the product (43%)
- Website Error (36%)
- Purchasing Process too long (35%)
- Site too slow (33%)
- Delivery, Payment, or pricing problems (14%)
- Browser Compatibility Problems (4%)

### 4.2 Security and Privacy Issues

Security turned out to be the attribute that was perceived by most participants to be of great importance. 35% of respondents claimed that if they were to choose a reason not to use e-commerce, it would be for fear of compromised security. It is interesting to note that another 30% would not choose e-commerce because they prefer to physically touch goods before buying them. This might be interpreted as users not trusting online vendors outright when it comes to delivering good quality products. Also, when asked how sure they would have to be of a sites capability to offer them security and privacy before they purchased from it, 43.5% of the users said they would have to be as certain as they possibly can be (water-tight privacy policy, secure connection, etc.). A further 42% said they would also buy if they had minor doubts (such as there being a risk of the site giving their e-mail to third parties). Security was placed first when participants were asked to rank quality attributes in order of importance. It obtained an average score of 6.235 (out of a maximum of 7). Surprisingly, 33 participants 13.87% claimed that security was not an essential attribute they would look for in an online store. However, on closer examination, these participants might have been inconsistent because when looking at their answers in isolation, they still placed security as the most important attribute with a score of 6.182. It can be safely concluded that security is an indispensable attribute in e-commerce systems.

### 4.3 Portability

Portability in e-commerce systems refers to the extent to which a system is accessible from different operating systems, browsers and devices without loss in functionality. The case for portability is not a strong one if one relies on the results of this survey. Consider the following results:

1. Participants ranked portability as the 5th most important attribute (out of 7)
2. 98% of participants use Windows™-based OSs
3. Almost 93% of participants use Desktop PCs
4. Over 95% of participants use Internet Explorer™
5. Less than 4% of users who abandoned transactions midway through did so because of compatibility problems. Less than half of these users where using Internet Explorer™

The results seem to suggest that if an e-commerce system where to be tailored for Windows™-based PCs or laptops using Internet Explorer™, any portability problems with other systems would cause minimal negative repercussions in the vendors business. Nevertheless, one should always remain vigilant with regards to portability issues. Firstly, when asked whether or not they would be willing to install another browser if an e-commerce site was not compatible with the one they were currently using, 88.65% of users said they would not. Therefore one must keep a close eye on the market share commanded by browsers and operating systems and invest in the portability of e-commerce sites as necessary. Another concern is the much talked about rise of mobile commerce (m-commerce). Even though early high hopes for M-Commerce failed to materialise in the first years of this century[15], falling costs and faster mobile networks have raised hopes on the feasibility of M-Commerce[16].

Portability is being recommended as a necessary attribute by the authors of this paper although in the current environment, compatibility with dominant technologies would seem to ensure a far greater reach and influence of online shoppers.

,dustin

### 4.4 Performance and Scalability

Speed is important to users. Over 33% of users who abandoned transactions mid-way through did so because the site was too slow. Also, when asked how they would react if their favorite e-commerce site experienced performance degradation due to popularity, only 18.4% of users claimed they would remain loyal. However, 57% claimed they would try to use the site at off-peak times in order get better performance. It is also worth noting, that 22% of participants consider the speed of a site the most important first impression factor. This would mean that when they first visit a site, the primary deciding factor on whether they decide to stay or not is performance. This is backed up by the popular 7-second rule, which states that a web page should take no more than 7 seconds to download and display to the site visitor on the slowest probable connection[10]. Participants rated performance as the 4th most important attribute in an e-commerce system with an average score of 4.128 (out of a possible 7). The authors are therefore recommending performance as a required attribute in all e-commerce systems.

### 4.5 Navigability, Usability and Aesthetic Features

Of the 77% of users who have abandoned transactions mid-way through, 35.6% did so because the process was seen as being too long. Also, these same users where more likely to look for an alternate site after abandoning a transaction rather than try again or contact the vendor by other means. This suggests that poor usability will have a tremendous impact on the success of an e-commerce site.

72% of users know beforehand what they are looking for when visiting an e-commerce site. This indicates that good search and navigation features are important in e-commerce systems. 30% of participants also chose good navigation as the primary first impression factor.

With regards to aesthetics, only 6.7% of users rate aesthetics as the primary first impression factor.

Navigability was ranked as the third (out of seven) most important attribute in e-commerce systems with an average score of 5.492 (out of a possible 7).

Usability with an emphasis on navigability would therefore be recommended as an essential attribute of e-commerce systems.

### 4.6 Multilingual Features

The importance of multilingual features depends very much on the context in which an online store is operating. For example, in some regions of northern Italy, both Italian and German are spoken to varying degrees. Therefore, an online store servicing such regions would do well to provide both Italian and German versions of its site. Respondents ranked multilinguality as the sixth (out of seven) most important attribute with an average score of 2.043 (out of a maximum of 7). Also, almost 51% of participants claim that an e-commerce system could still be classified as a quality e-commerce system if it did not have multilingual features. The reason for this might be that one tends to automatically use e-commerce sites in one's own language. Therefore users might not really be aware of the multilinguality problem.

Providing multilingual versions of an e-commerce site does not simply involve translating content into different languages. Besides there being a vast research area into the technical issues involved in offering such a service, there are also accompanying business and legal structures that would also need to be set up. For example, what happens if Japanese speaker places an order and needs customer support? The vendor would also need to have multilingual customer relationship management (CRM) processes in place. A recent study claimed that 46% of companies interviewed turn away international orders because they do not have the processes in place to handle them[19]. Implementing such processes would clearly require a certain amount of resources which most businesses, especially those in their early stages might not be able to afford. Therefore it might make more sense for businesses that are not large multi-national corporations to concentrate on markets where only one language is spoken and expand multilingual features as business grows.

Although a desirable attribute, multilinguality is not being recommended as an essential attribute by the authors of this paper.

### 4.7 Low-Bandwidth Version of Site

Some e-commerce sites have a text-only or low-bandwidth version available for users with slow connections. When asked about the importance of this feature, participants in the survey ranked it as being the least important attribute of all (average score of 1.587). Also, 52.5% of participants deem the feature to be unnecessary in an e-commerce site. Considering the increased Internet bandwidth available to users and the increasing popularity of broadband communications[17] as well as the giant strides in technology as a whole, the authors are not recommending low-bandwidth versions as an essential attribute of an e-commerce system.

### 4.8 Reliability

Reliability is considered to be an extremely important attribute in e-commerce systems because of the following indicators:

1. 36.6% of users have abandoned transactions midway through due to website errors
2. Reliability was ranked as the second most important attribute by participants with an average score of 5.55 out of a possible 7.
3. Almost 37% of users consider a sites reputation as the primary first impression factor when they first visit it and a site with frequent errors and crashes is unlikely to gain a good reputation.

Considering the above results from the survey, the authors recommend reliability as an essential attribute of e-commerce systems.

## 5 Conclusions and Future Work

Based on results of the survey and other research cited in this paper, a number of quality attributes are being recommended as being essential to e-commerce systems. That is to say, that no e-commerce system can be reasonably expected to succeed if it does not exhibit a considerable degree of each recommended attribute. The attributes are also being given an importance ranking as follows (most important first):

1. Security
2. Reliability
3. Navigability
4. Performance
5. Portability

Further work needs to be done before online store quality can be effectively measured. One still needs to define the minimum level of each characteristic that an e-commerce system needs to exhibit. In order to do that, each attribute should be measurable in some way. Therefore, the next step in the ongoing research will be to define a metrication and measurement framework for these attributes. When that is achieved, some form of progressive benchmarking system could be defined whereby e-commerce systems can be classified as being of a given quality depending on the level of each attribute exhibited by the system.

## References

1. Prime Minister's Strategy, "E-Commerce@its.best.uk", `www.number-10.gov.uk/su/ecomm/ec\_body.pdf`, 1999
2. Emarketer.com "Online Retail Update: Latest Q4 Quote", `www.emarketer.com/news/article.php?1002631`, Emarketer.com, 2004
3. Crosby P.B., "Quality is Free: The Art of Making Quality Certain", McGraw-Hill, 1979
4. Chaffey D., "E-Business and E-Commerce Management" Financial Times/Prentice Hall, 2002
5. Lee J., Podlaseck M., "Using a Starfield Visualization for Analyzing Product Performance of Online Stores", Proceedings of the 2nd ACM conference on Electronic commerce, 2000
6. ISO/IEC 9126:2001, "Software Engineering Product Quality", International Standards Organization for Standardization, 2001
7. Kafura D. "A Survey of Software Metrics", Proceedings of the ACM annual general conference on the range of computing, 1985
8. McCabe T.H., "A Complexity Measure", IEEE Transactions on Software Engineering, 1976
9. Chidamber S.R., Kemerer C.F., "Towards a Metrics Suite for Object Oriented Design", ACM Conference proceedings on Object-oriented programming systems, languages, and applications, 1991
10. Dustin E. et al, "Quality Web Systems", Addison Wesley, 2001

11. Chandra K., Chandra S.S. "A Comparison VBScript, JavaScript and JScript", Journal of Computing in Small Colleges (2003)
12. Deters R., (2001) "Scalability and Information Agents", ACM SIGAPP Applied Computing Review
13. www.w3schools.com, "January 2004 browser usage statistics", http://www.w3schools.com/browsers/browsers_statistics.asp, 2004
14. www.arkom.co.uk - "Web Browser Usage Survey 2003", http://www.arkom.co.uk/news-article.asp?NewsID=42, 2003
15. Mahoney M, "Whatever happened to mobile commerce?", E-CommerceTimes.com, http://www.ecommercetimes.com/perl/story/15042.html, 2001
16. Halper M, "Back From the Dead", Time Europe Magazine Vol. 163 No.7, 2004
17. Gill L, "Study: Broadband Adoption on the Rise", E-CommerceTimes.com, http://www.ecommercetimes.com/perl/story/18355.html, 2002
18. TWM, "How to Design a Survey", www.thewritemarket.com/marketing/survey-design.htm, 2003
19. European Business Management School, "SME Management of Multilingual Web sites", http://www.global-presence.org.uk

# Simulo Tempestas:
# A Simulation Tool for Hydrological Modelling

Andrew Calleja[1]

Department of Computer Science and AI,
University of Malta

**Abstract.** Man's existence has clearly interfered with nature. Ever since man appeared on the face of our planet, the landscape was adapted to make it a more habitable environment. Although early humans transformed land to accommodate their dwellings and livestock, land was changed only to a certain extent. However, modern man has altered his surroundings beyond recognition through the building of road networks, sprawling cities, industrial zones, so on and so forth. This project studies the natural flow of water over the terrain through the use of a simulation tool, with particular focus on the Maltese Islands. Such a tool would help engineers plan better the assignment of construction zones as it allows proper analysis of the effects of land development on the flow of storm water before making any commitment that would change the landscape irreversibly. Different weather scenarios, based either on past statistics or completely fictitious ones, could be fed to the tool and its effects studied in the quest of finding the best solutions to avoid man-made catastrophes.

## 1 Introduction

Water is a major constituent of all living matter and is the key element to life. The hydrological cycle which entails, more than anything else, the precipitation of water on the ground is thus a part of a very important cycle, as it enables this precious resource to reach every corner of the earth. At times, however, this cycle causes extreme conditions like drought and floods. *Simulo Tempestas*, derived from the Latin meaning *'simulate storms'*, is a simulation tool that performs hydrological modelling on any given area of land to help in the prediction and prevention of flooding. While this can be applied to any area, the main focus shall be on Maltese terrain.

*Hydrology*, is defined by The Concise Oxford Dictionary, as *'the science of the properties of the earth's water especially of its movement in relation to the land.'*

The simulator models the interaction of water with a terrain in a three dimensional virtual world to help identify possible solutions to a flooding problem. The rainy weather plays the important role of 'watering' our world and our hydrological model must be able to mimic nature in dealing with the water once it touches the ground.

Almost every year the rainy season starts with a violent storm that causes damage to our small but precious islands. The development of this project has been motivated by the serious implications of events that often have great economic repercussions and at times even loss of life. These events are brought about by two major factors that can be attributed to man's interference with nature. The first factor is the climatic change brought about by what is known as the Global Warming

[1] This work was presented in June 2004 to the Board of Studies of Information Technology at the University of Malta as a Final Year Project for the B.Sc.(Hons.)I.T. degree programme, supervised by Dr. John Abela.

phenomenon which the entire world is experiencing. Although there is strong political and scientific debate whether or not this phenomenon is the result of atmospheric pollution, there *are* changes in the climate. Extreme weather conditions like flash flooding and drought are clearly becoming more common.

Then there is the impact of buildings on the flow of storm water. On the local scene the construction boom over the last three decades has been eating away a disproportionate amount of land while vast areas occupied by post-war buildings have been left untouched. This puts pressure on the little land left available to cater for the increasing population and the exigencies of industry, causing damage to nature's own way of dealing with run-off water. The construction of whole towns and villages snap-on in the middle of major water courses is a huge mistake of the not so distant past which could have been easily avoided had common sense prevailed and the right tools been available to plan better.

### 1.1 General Objectives

The objective of Simulo Tempestas is to help hydrologists, engineers and town-planners in their daily work. In order to prevent catastrophes, engineers can assess the consequences that hypothetical land transformations may have on the hydrological balance. This is a very practical methodology as it allows experimentation without putting at risk the lives of people and the destruction of our land. The simulation operates in a world characterised by two entities:

- **Terrain** - the surface on which the simulation tool will model the flow of water.
- **Climate** - the entity responsible for pummeling the terrain with water in the form of precipitation.

Having defined our virtual world the objectives of Simulo Tempestas can be better understood. The following are the basic aims of the system:

1. to allow the creation of a customisable world including both the terrain and the climate
2. to simulate storms over primarily Maltese terrain, but flexible enough for other custom terrains
3. to render the simulation graphically in three dimensions
4. to present statistics and results in such a format that it is of great use to hydrologists and engineers

To achieve these objectives the project development is subdivided into two main parts:

1. World-editor tools
2. Simulation tool

**World-editor tools** The idea of having world-editor tools is to create a customised simulation project for the simulation tool. Over all, the primary aims to be achieved by the world-editor tools are:

- to let a user create/modify a terrain,
- to allow the user assign different parameters to the ground like absorptivity and roughness parameters,
- to give the user the ability to divide the terrain into named regions for better identification,
- to be as customisable as possible in the creation of a climate,
- to be able to save and load project files,
- to allow the exchange of data between different projects,
- to have a mixture of 2D and 3D elements to give the user all the relevant feedback.

**Simulation Tool** The simulation tool is responsible for running the actual simulation. Its primary objectives are:

- to allow the user load project files from the world-editor tools,
- to allow the configuration of simulation run-time parameters,
- to be as explicit as possible by having a mixture of 2D and 3D elements,
- to produce results relevant to flood analysis,
- to allow the user to save the outcome of a simulation for future use.

## 2 Design

Simulation tools are quite useless if they are not versatile in their configuration. However, versatility might also mean the introduction of additional complexity to the system which results in an unnecessary cognitive overload on the user. It is for this reason that Simulo Tempestas has been subdivided into three different parts, each part designated as an application of its own. With three distinct applications, each of them having their own specific role in the system, the user will be able to grasp their use faster than having one big application carrying out a multitude of different functionalities. The following is how the system is divided:

1. Terrain Modifier: A tool that allows the user to modify the terrain to be able to test new situations; e.g building dams, creating deeper basins, creating new canals, etc
2. Simulation Configuration Tool: This tool allows the user to create the climatic environment and terrain settings to be used in the simulation.
3. Simulation Tool: The tool responsible for carrying out the simulation itself depending on the settings given to it using the previously mentioned tools.

During the research phase it was seen how the problem domain has been tackled by others during the past years. The different techniques related to hydrological modelling with their pros and cons were analysed. It is rather useless implementing something already done before, therefore important changes are made by using the best of the these different models to come up with a fresh solution to the problem.

### 2.1 Terrain

The terrain is a very crucial aspect in hydrological modelling as without it there is nothing to model. Hence, in a computer simulation, the terrain itself as well as the quality of the terrain data is quite important. Something of real importance is that the terrain model is represented in the most efficient way possible so that any operations can be performed quickly. This is a big prerequisite especially in real-time simulations. The size of the terrain and the underlying data structure with which it is represented directly affect the efficiency of the algorithms manipulating the terrain data.

**Digital Elevation Model** The system makes use of 50-metre Digital Elevation Model which is nothing more than a 2-dimensional grid for which elevation data exists at the intersections of the grid rows and columns. The spacing between the points on a DEM grid (i.e. the row and column, height and width respectively) directly affects the level of detail of a DEM. In the case

of a hydrological modelling tool, a refined and detailed DEM is required. This elevation data can be turned into what is called a *height-map.* The points on the DEM grid which have been taken at regular intervals, when stored in a 2D grid, are represented by the row/column cells. These individual pixels/cells represent elevation values on the map and are assigned a particular shade depending on the altitude. Usually shades of grey are used to produce a gray-scale image that shows the topographic relief.

**Ground Properties** There are many geological parameters attributed with the terrain that enhance the accuracy of hydrological models. However, from the research performed, three detrimental parameters have been identified as being crucial to our simulation and cannot be ignored. These three properties are:

1. Absorption
2. Ground Saturation
3. Terrain Roughness

Absorption and ground saturation properties are linked to *infiltration modelling* whilst the terrain roughness is linked to *flow modelling.* Infiltration modelling is important in establishing the amount of surface run-off generated during a storm. The more absorptive the ground is, the less run-off is generated. On the other hand, flow modelling is needed to establish the flow patterns and velocities over the ground over time. To be able to perform such modelling all the relevant data must be stored in an efficient and accessible manner.

### 2.2 Layering

The extra information associated with the terrain can be seen as a series of layers stacked on top of each other. These layers on are not necessarily dependent on each other, they just enhance the detail of our bottom layer - the terrain. This layering approach has several advantages from both the technical and usability point of view. The user is able to store many different layers on disk and load any one of them into the application at will. Different roughness layers, for example, might be used in conjunction with a particular climate layer. This allows better objective comparisons due to the ability of comparing like with like. The discrepancies that result from the simulation are known to be due to the change in roughness parameters. Once all the layers have been loaded and all the parameters set, everything is saved to a project file. These project files can then be fed into the simulation engine, through the simulation tool, and run. The following are the layers designed for Simulo Tempestas:

**Terrain Layer** This layer is the most important layer of all and is at the bottom of our layer structure. The terrain layer is represented, as has been explained earlier on, as a 2D grid of height values called a DEM. Having the terrain as a layer gives us that versatility that allows us to modify the terrain's geo-morphology independently from the rest of the ground properties.

**Absorption Index Layer** The absorption index is a value that represents the rate of absorptivity of the ground. The absorption rates of the ground can be set and adjusted depending on the surface geology of the terrain. Such information can be usually found in soil maps. This absorption rate depends on another factor also stored in this layer - the ground saturation. These two properties are tightly coupled together as they inevitably affect each other. The decrease in the absorption rate of the ground is due to a change in its saturation. The mapping between these two parameters can be either a system-generated linear decay relationship or else a custom drawn decay graph.

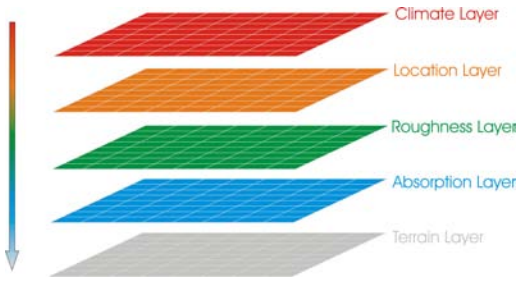



**Fig. 1.** *The layering in Simulo Tempestas*

**Roughness Index Layer** It was mentioned earlier on that our flow model will simulate the flow of water against time. A semi-empirical formula called Manning's Equation is adopted for such flow-rate prediction and is described later on in this synopsis. The so called *Manning constant,* is used in conjunction with other geo-morphological values, derived from the terrain data, to be able to calculate such water velocities. This *Manning constant* is what is being called in this system the *Roughness Index.* Allowing different areas on the ground to be assigned a roughness index enables better modelling of flow. To cite a small example, it can be said that a cemented surface allows better flow of water than a rocky and weedy surface. This index helps identify such features on the terrain.

**Location Layer** This layer serves the purpose of partitioning the terrain into indexed, logical partitions. Partitioning helps us understand and assimilate the results better since humans, by nature, give names to regions and places so that they are identifiable. The user might want to collect results from the tool using some particular scheme such as dividing the Maltese terrain by region e.g. North, South and Central or by locality e.g Attard, Birkirkara, Balzan.

**Climate Layer** The last and top most layer in our system, is as important as the rest. Ultimately, to be able to run a simulation something is needed to generate precipitation over the terrain. This layer consists of virtual clouds and wind which, as seen later on, are responsible for providing the data to our hydrological model.

## 3 Simulation Engine

Apart from performing the actual modelling, the simulation engine is also responsible for preparing and populating the necessary data structures that will be used during the simulation. These populated structures will be used throughout the entire simulation process many times over and have been meticulously designed and optimised to increase the efficiency of the simulation. Increasing the efficiency meant finding a balance between processor and memory usage.

The simulation process involves the triggering of appropriate events in an orderly fashion for a number of cycles. In one cycle of the simulation process, referred to as a *Tick*, the model is evolved by integrating the new input with the results of the previous cycle to produce another set of results. These in turn will be used again as input in the next cycle. Since the modelling is performed against time and our simulation engine works in a discrete manner, one cycle of events must represent some fixed amount of time. Having a cycle represent a small amount of time means that the simulation

will take longer to complete. It is however left up to the user to decide how much time each cycle represents as this depends on the exigencies. The following mentions very briefly the processes required to be abstracted from nature and most importantly the way it is done.

### 3.1 Flow modelling

Routing the flow from one pixel to the next is one of the most important models in a hydrological simulation. Many different techniques have been presented along the years to determine how and where the water flows once it is 'dropped' on the ground by the rain-falling process. The very first flow path algorithms were the *steepest descent* methods [3], more popularly known as the D8 (Deterministic 8-node) algorithms. This method is in use by many GIS applications but has a major disadvantage as it only assigns one flow direction to one of eight possible neighbours and thus fails to model divergent flow properly. Another set of deterministic methods, called Fractional 8 (F8) assign flow to multiple directions by weighting the flow according to the slopes of descent. Such multiple-flow direction algorithm give more realistic representation of flows[2].

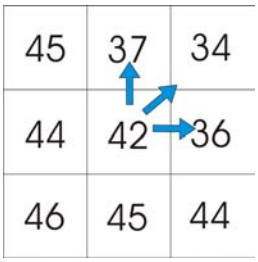



**Fig. 2.** *F8 multiple-flow method*

The method designed for Simulo Tempestas operates in a multiple-flow fashion and is integrated with a 'diffuse wave equation', called the Manning equation, to be able to calculate flow speeds depending on the surrounding lower-lying cells. If one or more neighbouring cells are lower in height, water will flow to each of them at a rate which is directly proportional to the gradient of descent - the steeper the descent the more the water will find it 'attractive' to use as an escape. The Manning equation is reproduced below:

$$V = \frac{U_m}{n} \,.\, R_h^{\frac{2}{3}} \,.\, S^{\frac{1}{2}} \tag{1}$$

$$F = V.CA \tag{2}$$

Due to the discrete nature of the DEM each of these cell-blocks, with a base area of 50 x 50 metres (i.e. 2500 $m^2$), behaves like a container. Therefore the volume introduced into the cell, ignoring for a moment the amount of water infiltrated into the ground, is contained within it and will contribute toward raising the water level of the entire cell.

Having the the **slope percentage** (pre-calculated), the **Manning constant** (provided), the **water level** in the cell (calculated at run-time) the flow **velocity** can be calculated using equation (1). The **volume** discharged can then be calculated by making an assumption about the geo-morphology

of the channels that the cell shares an equal boundary of 25m with its eight neighbours. Having the width of the stream and water level the volume of water discharged can be derived through equation (2).

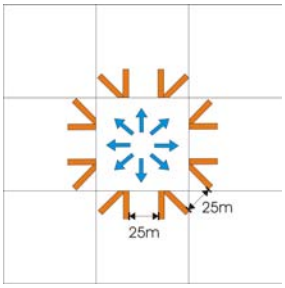



**Fig. 3.** *The channels assumed to exist between the cells*

The flow system developed can be considered to be particle-oriented system as each cell acts individually by pumping out the water to its neighbouring cells. This abstraction, termed as *FlowCell* in the context of Simulo Tempestas, is used to signify that to model water flow these pumps shall be pumping water into each other.

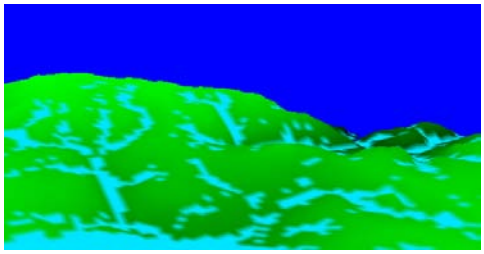


**Fig. 4.** *Screenshot - Run-off generation minutes after a storm begins*

**Flat lands and Catchment areas** All pixels on the terrain must be able to route flow downward if we are to produce a flood model. However, certain regions on the terrain like flat and pit areas cannot pass water as no steeper neighbours exist. Most hydrological modelling applications implement a pixel filling algorithm in the pre-processing stages to alter those cells on the ground that cannot pass water to other cells. The terrain is modified in these areas to force flow. This action seems to be to drastic so in our case, instead of making modifications to the terrain like the pixel filling algorithm, these regions are treated differently by creating to different concepts - *Flatlands* and *Catchments*.

**Flatlands** Flatlands are areas on the terrain composed of two or more neighbouring cells that have the same elevation value. In real-life, when water is poured onto a flat surface like a table, the water spreads all over the surface uniformly. The water keeps spreading, given that there is a sufficient

volume of water to do so, until it encounters a drop in elevation (like the edge of a table) after which it all drains out from the boundary leaving only a thin film due to water tension properties. This is how Simulo Tempestas models the flow over a flat terrain. Whenever water is introduced to a small part in a flat piece of land, the water spreads throughout uniformly contributing to a rise in the water level throughout the entire area. Then, at the boundary of the flat land two different scenarios are possible:

1. The flat land might be part of a pit and thus all boundary pixels have a higher lying neighbour (discussed under the *Catchments* right after),
2. or else at least one boundary cell has a lower lying neighbour.

Since these group of cells are now treated as one object there is no need to make any alterations to the individual cells. During the simulation, when modelling precipitation, the entire flat land is treated as one piece therefore any water introduced is contributing to a global rise and any water flowing out is contributing to a global lowering of the level.

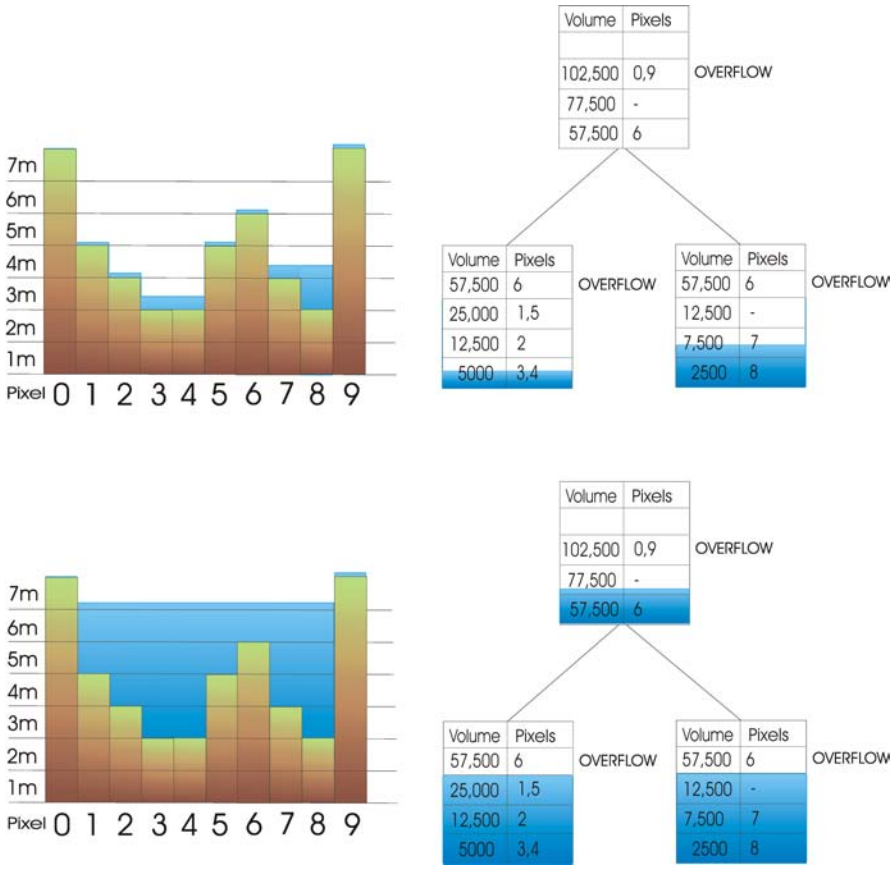
**Catchments** When surface run-off is generated and made to flow down slopes it might end up in some areas where there is no escape. Water flowing into parts on the terrain that are contained start to fill up and thus change completely the dynamics of the flow. Catchments, as defined in the context of Simulo Tempestas, are areas on the terrain that take water in to form 'pools'. This filling process happens because of the existence of pixels that are lower than the rest. The discretised terrain provides height values to the nearest metre, therefore when catchments start filling up, changes occur only at intervals of one metre. At these intervals, with incoming water flow increasing the height level, the surface area of the catchment grows bigger as it consumes new cells on the catchment brim. These cells are not regarded anymore as a separate pump with the previous slope gradients to lower pixels, but as cells that share the same water level with all the neighbouring cells that are part of this new catchment surface.

At one point, when the water level reaches the 'brim' of the catchment and makes contact with the rim of other neighbouring catchments, the water will start to overflow accordingly. Once these neighbouring catchments fill up to the brim too they merge into one bigger catchment forming a bigger surface area and once again the dynamics of the flow changes as they now share common boundaries. Mimicking such interconnectivity between catchments is possible through the use of a tree structure. These catchments are interconnected in such a way that help us keep track of the water in these pits. The purpose of such a tree is to store a hierarchy of these catchments as they fill up with water. The idea behind this approach came from a paper by Lee Vincent and Pierre Soille [1]. The authors of this paper discuss simulating a flooding process as if water is coming out of the ground, moving upward. They do this to delineate the watershed areas on the terrain without any pre-processing of the pits on the ground. To establish these catchment areas they process the pixels one altitude at a time starting from lowest going up to flooding the entire terrain.

### 3.2 Climate

Just as nature has its clouds and wind, our system too has its virtual clouds and wind. These clouds are stored in a climate layer in the layering system mentioned before. The precipitation process is responsible for feeding the hydrological model with rain. The rain data is fed into each individual cell on the ground by the simulation engine. Due to the nature of our DEM data i.e. the fact the cells have a large surface area of 50m by 50m, an assumption must be made that water is dropped

**Fig. 5.** *Simplified catchment tree at work*



as a thin sheet over the cell surface. Depending on the precipitation rate and the cloud coverage, the volume 'emitted' by the cloud can be easily calculated. Raining on the terrain is done in three phases;

1. Raining on catchment areas
2. Raining on flat lands
3. Raining on the individual flow cells

The reasons behind the introduction of these three particular objects into our system have already been discussed. The flat lands and catchments, both comprised of many flow cells, are now seen as a group of cells that have to be all treated in the same way as they share the same water surface. They have to be viewed as different-sized 'particles' and thus have to be rained upon differently. Since each ground cell is defined as a flow cell and acts in an autonomous fashion the climate must feed water to them individually.

## 4 Conclusions and Future Work

Simulo Tempestas allows an engineer to effectively plan the development of a region so as not to disrupt the natural course of storm water. Better still, it could actually be used to improve an area, by making changes that will allow it to cope with extraordinary heavy downpours. All the four project primary objectives that were set at the outset have been achieved since the end product:

- allows the creation of a customisable terrain complete with its climate
- it is fully capable of simulating storms
- it presents such storms and its effects graphically in three dimensions
- it captures and presents statistics and results in a useful format for hydrologists and engineers.

It gives the user the possibility to examine hypothetical weather scenarios and their impact over time. At the same time it is structured in such a way that future add-ons can be plugged in with very minor modifications. The simulator can be developed further into an early warning system for flood prone areas if it is integrated with live satellite DEMs and weather forecasting software. Such a package could be radio-linked to earth stations in vulnerable areas from where evacuation plans can be triggered to mobilise communities in danger and move them to safer zones.

**Acknowledgements**: First of all I would like to thank my Final Year Project supervisor Dr. John Abela for his time and dedication throughout the project's realisation. Thanks go out as well to hydrologist Dr. Godwin Debono, head of the Oil Exploration division at the Ministry for Rural Affairs and the Environment and Mr Saviour Porter, Head of Meteorological Office in Luqa. Finally, a big thanks to the Department of Computer Science and AI at the University of Malta for giving me the opportunity to present the project at the Computer Science Annual Research Workshop.

## References

[1] Lee Vincent and Pierre Soille. Watersheds in digital spaces: An efficient algorithm based on immersion simulations. IEEE PAMI, 1991, 13(6):583598, 1991.
[2] Quinn, P., K. Beven, P. Chevalier, and O. Planchon. The prediction of hillslope flow paths for distributed hydrological modelling using digital terrain models, Hydrological Processes, 5, 5979, 1991.
[3] O'Callaghan, J. F., and D. M. Mark. The extraction of drainage networks from digital elevation data, Computer Vision Graphics Image Process., 28, 323344, 1984.

# Forecasting Using Non-Linear Techniques In Time Series Analysis: An Overview Of Techniques and Main Issues

Michel Camilleri

Department of Computer Science and AI,
University of Malta

**Abstract.** The development of techniques in non linear time series analysis has emerged from its time series background and developed over the last few decades into a range of techniques which aim to fill a gap in the ability to model and forecast certain types of data sets such a chaotic determinate systems. These systems are found in many diverse areas of natural and human spheres. This study outlines the background within which these techniques developed, the fundamental elements on which they are based and details some of the predictive techniques. This study aims to provide some insight into their mechanisms and their potential.

## 1 Introduction: The Need for Forecasting

No one knows the future, but there is a lot of benefit to be derived from attempting to form a picture of what the future could be - even if it is imprecise. It is one of the task most decision makers are expected asked to do - from the hunter-farmers of prehistoric age to the stock analysts of today.

[1] Almost all managerial decisions are based on forecasts. Every decision becomes operational at some point in the future, so it should be based on forecasts of future conditions.

A significant amount of research has been carried out using time series techniques in such diverse fields as [13] modelling and predicting solar sun spot activity, [5] analysis of Ethernet traffic, Business cycles, [3] Intelligent analysis of clinical time series in Diabetes Mellitus. Non linear techniques are emerging as powerful tool in the hands of the analyst to leverage patterns of behaviour underlying hiding dynamical systems - which may elude analysis using other traditional techniques.

[1] A model is an external and explicit representation of a part of reality as it is seen by individuals who wish to use this model to understand, change, manage and control that part of reality. Analysts try to model the system or process which is being studied in a variety of ways.

Regression Is the study of relationships among variables, a principle purpose of which is to predict, estimate the value of one variable from known or assumed values of other variables related to it. Using one predictor it is called *simple linear regression*. Using two or more predictors, *multiple regression* analysis is employed

A Scatter diagram is a graphical representation of the pairs of data to gain an overall view of the problem — is there an apparent relationship? Direct, inverse? If the points lie within a band described by parallel lines we can say there is a *linear relationship* between the pair of $x$ and $y$ values. If the rate of change is generally not constant then the relationship is *curvilinear*.

Linear Models: One would determine if a linear relationship exists between $t$ and $y$ and then attempt to produce a linear equation stating $y$ as a function of $x$ in the form of $y = a + bx + e$

where $a$ is the intercept, $b$ is the slope and $e$ is the error term accounting for variables that affect $y$ but are not included as predictors, and/or otherwise unpredictable and uncontrollable factors.

Least squares method: To predict the mean $y$-value for a given $x$-value, we need a line which passes through the mean value of both $x$ and $y$ and which minimizes the sum of the distance between each of the points and the predictive line — the best fit to the sample data. The least squares method achieves this by calculating the minimum averaged squared deviations between sample $y$ points and the estimated line.

A procedure is used for finding values of $a$ and $b$ which reduces to the solution of simultaneous linear equations. Shortcut formula have been developed as an alternative.

Solution methods: Techniques of matrix algebra can be manually employed to solve simultaneous linear equations especially useful when there are more than two equations and two unknowns. Several computer packages using matrix algebra to solve simultaneous equations are widely available and can be utilised to relieve the computational problems - can solve both linear and polynomial equations

Variables of interest: To make predictions or estimates we must identify the effective predictors of the variable of interest which variables are important indicators? And can be measured at the least cost? Which carry only a little information? Which are redundant?

## 2 Time Series Analysis

[10] A time series is a set of numbers that measures the status of some activity over time. It is the historical record of some activity, with measurements taken at equally (not always) spaced intervals with a consistency in the activity and the method of measurement [1].

Time Series Analysis comprises methods for the quantitative characterization, the manipulation, and the understanding of time series data and the underlying systems.

Time Series Forecasting Approaches There are two basic approaches to forecasting time series: the self-projecting time series and the cause-and-effect approach. It is possible that both approaches still lead to the creation of accurate and useful forecasts. The first is often the more difficult to implement and validate than the second.

Self-projecting time series: This approach uses only the time series data of the activity to be forecast to generate forecasts. The basic idea behind self-projecting time series forecasting model is to find a mathematical formula that will approximately generate the historical patterns in a time series.

Linear correlations in time are the most common and best studied sources of predictability. Linear statistical methods assume a random nature of the data since the underlying auto-regressive and moving average models are stochastic. Auto-regressive models and moving average models are stochastic due to the assumed random nature of the data [10]. The randomness prevents one from making any precise predictions, even with an absolute knowledge of the present state, but the strong correlation assures one that the next observation will be given by approximately by a linear combination of the preceding observations, except for additive noise ie:

$$s_{n+1} = a_0 + \sum_{j=1}^{m} a_j s_{n-m+j}$$

Autoregressive models is one of a group of linear prediction formulae that attempt to predict an output of a system based on the previous outputs and inputs. Autoregressive processes as their name implies, regress on themselves. The current value of the series is a linear combination of the

$p$ most recent past values of itself plus an error term which incorporates everything new in the series at time $t$ that is not explained by the past values.

An autoregressive model AR is a model which depends only on the previous outputs of the system is called. A moving average model MA. Is a model which depends only on the inputs to the system. AN autoregressive-moving-average model ARMA is a model based on both inputs and output.

Deterministic dynamical systems: Pure dynamical systems are described either by discrete time maps where:

$$x_{n+1} = f(x_n) \text{ ($n$ is a discrete unit of time)}$$

Or by first order ordinary differential equations:

$$dx(t0)/dt = f(t, x(t)) \text{ ($t$ is a real value of time)}$$

Both are descriptions of the fact that all future states of such nonlinear deterministic dynamical systems are unambiguously determined by specifying its present state at some time $n$ (or $t$). Thus there also exists a deterministic forecasting function.

Nonlinear deterministic dynamical systems: Another type of correlation in time is present in non-linear deterministic dynamical systems. The underlying systems would have a deterministic nature but display irregularities. Chaos hence is an alternative source for unpredictability and irregularity in observed data, an alternative with respect to stochastic inputs.

These irregularities in the measurements arise from processes which co-exist with the system under study or are due to inaccuracies (noise) in the measurement process itself.

Any irregularities in the knowledge of the present state will evolve over time, in in the case of a chaotic system, will grow exponentially. However the uncertainty is amplified only at a finite rate and one can still hope to make reasonable short term forecasts.

Nonlinear prediction techniques exploit correlations over time visible only by using non-linear statistics.

## 3 Non-Linear Time Series Analysis

[11] The pioneering work by Packard et al. and the seminal paper by F. Takens stimulated, about 20 years ago, a new approach towards complex dynamical phenomena, nowadays called nonlinear time series analysis. Extensions towards nonlinear stochastic models and towards nonlinear statistical analysis are manifold and have been considered within various frameworks.

Stationarity: [10] A signal is called stationary if all transition probabilities from one state of the system to another are independent of time within the observation period. All relevant parameters have to be kept constant and that phenomena belonging to the dynamics are contained in the time series sufficiently frequently so that the probabilities or other rule scan be inferred properly.

Unpredictability: [10] of the future is the most striking feature of chaos despite a deterministic evolution. This unpredicatbiity is a consequence of the inherent instability of the solutions reflected by what is called sensitive dependence on initial conditions. The tiny deviations between the intial conditions of all the trajectories are blown up in a few time steps and every unobserved detail of the state at time zero is important for the precise path of the trajectory in state space.

This lead to two concepts (i) loss of information related to unpredictability; and (ii) nearby trajectories separate fast — exponentially fast over time.

If separation over time is very slow this is not in itself dramatic — typical of predominantly periodic systems. If this separation of trajectories over time is fast, exponentially fast — exponential divergence — we can speak of chaos. This increase of separation over time can be a characteristic of the underlying system.

Measurement technique and Sampling Rates: [10] A time series (as any other measurement) has to provide enough information to determine the quantity of interest unambiguously. Algorithms have some minimal requirements as to how long and how precise the time series must be and how frequently measurements have to be taken in order to obtain meaningful results. If meaningful results are not obtained then a smaller sub-phenomenon is to be studied. Sufficient sampling is required in general not just to particular methods of time series analysis

Example of sufficient sampling: Measurement of temperature changes on roof of house. How many single measurements are needed? Answer: it depends:

**For one day:** check one per hour for one week to get a fair average of the temperature profile
**For seasonal fluctuations:** fixed times, once a day for a year
**For year to year:** couple of measurements every year
**For a complete description of phenomenon:** once an hour, every day, for a couple of years

Modelling and Prediction depend on the measurement process, sampling technique and also the form of representation of the data.

## 4 Data Representation

One can create different perspectives of the raw data in order to be in a better position to examine and discover relationships over time. One may also apply techniques, such as — temporal abstraction, Phase space embedding, Poincare Map to render the data more suitable for prediction prior to prediction itself.

Scalar measurement/time chart: A dynamical system can be represented by a measurement which changes value over time e.g. in the shape a periodic function $\sin(t)$. The observations in an experiment are usually a time series - a scalar sequence of measurements over time. [10]

Describing dynamical systems in phase space: [10] A dynamical system can be described either by an m-dimensional map or by an explicit system of m first-order differential equations. Dynamical systems can be described by equations of motion which are defined in phase space.

It is important to establish a vector space (state space or phase space) for the system. Specifying a point in phase space specifies the state of the system and vice-versa. In a purely deterministic system once the present state is fixed, the states at all future points are determined as well.

By studying the dynamics of the corresponding phase space points one can thus study the dynamics of the system. There can be different choices for what phase space of a system can be.

Non linear prediction techniques, determination of Lyaponuv Exponents, noise filtering and other related applications use approximations of the dynamical equations to represent the underlying dynamical system. Such equations are defined in phase space so it is most natural to use a phase space description for these approximations too.

A trajectory in phase space of the dynamical system is a sequence of points $x_n$ or $x(t_0)$ solving the above equations. An interesting exercise is to plot trajectories which start with similar phase

states (at least the first three phase states) on top of each other and observe the variation as time progresses. In the case of non-linear prediction, one is interested in those trajectories which stay in a bounded area forever. Other trajectories may run away to infinity as time proceeds. One is also interested in the rate at which they separate away from each other as time progresses

When irregularity is present in the signal we can talk of chaos in a deterministic system.

### 4.1 Phase Space Reconstruction

One plots the phase state of the function at different time points. For a single coordinate pair, the value of $y$ will be the value of the function at time point n is plotted and the value of x will be the value of the function at the previous time point $n-1$ the $x$ coordinate. All phase states i.e. the coordinate pairs $s(n)$, $s(n-1)$ are evaluated for all $n$.

The resulting chart will be a plot of the state of change of the system at different time points ie. the phase state map. If stationarity is present in the process, such a map of phase states will demonstrate this aspect more evidently than a normal time series plot of $F(t)$ vs $t$.

Phase Space Reconstruction Is at the core of nonlinear time series analysis. In such reconstructed spaces, the deterministic features of systems can be identified and studied, such as empirical invariant measures, attractor dimensions, entropies, Lyapunov exponents, equations of motion, and short-term forecasts can be made.

### 4.2 Elements of Phase Space Embedding

[10] The time series is initially a sequence of scalar measurements of some quantity which depends on the current state of the system taken at multiples of a fixed sampling time:

$$S_n = s(x(ndt)) + N_n$$

Some measurement function $s$ and make observations only up to some random fluctuations $N_n$, the measurement noise. For a moment we will disregard the effect of noise.

A delay reconstruction in m dimensions is then formed by the vectors $s_n$, given as

$$S_n = (s_n - (m-1)v, s_n - (m-2)v, \ldots, s_{n-v}, s_n)$$

The lag or delay time is the time difference in number of samples $v$ (or in time units $vdt$) between adjacent components of the delay vectors.

A number of embedding theorems are concerned with the question under which circumstances and to what extent the geometrical object formed by the vectors $s_n$, is equivalent to the original trajectory $x_n$. Under quite general circumstances if the dimension m of the delay coordinate space is sufficiently large, the attractor formed by $s_n$ is equivalent to the attractor in the unknown space in which the original system is living.

The most important embedding parameter is the product ($m$ x $t$) — embedding dimensions $x$ delay time. It is the time span represented by the embedding vector. This can be derived after finding the two parameters separately.

Finding a good embedding in phase space: [10] In an ideal noise-free data there exists a dimension m such that the vectors $s_n$ are equivalent to phase space vectors Choose too large $m$ — will add redundancy and degrade performance of many algorithms — predictions, Lyaponuv Exponent. Choose too small an $m$ and the vector will not represent the system and prediction errors will

be high. One must carry out analysis increasing the values of $m$ until the typical behavour of deterministic data appears.

Finding value of smallest $m$ with no False Neighbours: Consider an attractor with an embedding dimension $m_0$ i.e. it represents the system. If one were to generate a number of predicted values for the attractor itself using the coordinates of the attractor one on each other - the predicted values should all be close neighbours of the actual attractor points. If one were to reduce the dimension of the attractor by one — it could be that the attractor no longer represents the system and the predicted values will not be close or neighbours of the actual attractor coordinates. This number of false neighbours are generated by this $(m_0 - 1)$ dimension attractor.

Turning this concept into an algorithm one would start with small values of $m$ and calculating the number of false neighbours generated. Gradually increase the number of dimensions by one till the number of false neighbours is zero. This means that this number of dimensions is the best embedding for the system - the one with the least dimensions required to correctly represent the system.

Finding a good estimate of the lag or time delay: A good estimate for the lag time is more difficult. Embeddings with different $r$ but same $m$ are the same mathematically and for noise-free data. However a good choice of $t$ facilitates analysis.

If $t$ is small compared to the internal time scales of the system, successive elements of the delay vector are strongly correlated. All vectors $s_n$ are then clustered around the diagonal in $Rm$ unless $m$ is very large. If $t$ is very large, successive elements are already almost independent and the points fill a large cloud in the $Rm$, where the deterministic structures are confine to the very small scales.

## 5 Examination of Data

Prior to applying any modelling or predictive techniques, the analyst is to examine the data using a variety of techniques amongst which is visual inspection of various plots which can be obtained prior to processing.

Visual examination is still a powerful tool which must not be dismissed. Human visual interpretation of data and results still lies at the top of the methods to understand systems.

A number of mathematical/computational techniques also exist which assist the analyst in obtaining a deeper understanding of the dynamics of the system being studied. Here are two of the principal ones.

Lyapunov exponent [10] can be a characteristic of the underlying system. It will indicate the change in distance of trajectories with initially neighbouring points. can be described by the properly averaged exponent of the increase of separation of trajectories over time. A number of these exponents can be calculated for various dimensions $m$.

Autocorrelation sum is a measure of how many neighbours each point has in phase space. One may superficially deduce that the more neighbours the better the predictability. [10] However autocorrelation of signals from deterministic chaotic systems decay exponentially with increasing lag. One has to be very cautious about interpreting these values.

## 6 A Simple Non-Linear Prediction Technique using Delay Vectors

[10] Here a simple prediction technique exploits the deterministic structures in the signal. The data is assumed to originate from measurement of an underlying dynamical system.

One needs to predict a future state at a desired time point $n+1$ using the last value at time point $n$. One scans the list of all states in phase space and tries to find the one closest (according to some norm) to the phase state value at time point $n$.

If a time point $n_0$ Is found where the phase state is similar to that at $n$ (this means $x_{n0}$ it is close to $x_n$). Then the continuity of the underlying dynamical system and the representation guarantees that $x_{n0+1}$ will also be close to $x_{n+1}$.

However usually it is not the actual states that are measured but one or more quantities which depend on these states. More often than not the measurement function is as unknown as the underlying equation of motion.

Thus we have scalar measurements $S_n = s(x_n)$, where $x_n$ is the actual state at time point $n$ and $n$ can take the values from 1 to $N$. One can then make use of a set of such measurements taken at a particular time point $n$ and $(m-1)$ time points prior to $n$ to represent the original sequence of phase states. This is called a delay reconstruction and the sequence of phase state values representing is called the delay vector:

$$S_n = (s_{n-(m-1)}v, s_{n-(m-2)}v, \ldots, s_n - v, s_n)$$

This introduces two adjustable parameters into the prediction method (which in principle is parameter free). The number of dimensions m and the time delay which is the time delay $v$ between successive measurements. For all measurements $s_1, \ldots, s_n$ available so far, one can construct the corresponding delay vectors $s_{(m-1)}v + 1, \ldots, s_n$.

In order to predict a future measurement $s_n + d_n$, one can find the embedding vector closest to $s_n$ and use $s_{no+dn}$ as a predictor. This scheme has been used in tests for determinism by Kennel and Isabelle (1992). The idea may go back to Pikovsky 1986. The method used by Sugihara and May (1990) is closely related.

This means that one scans all measurements and for each measurement the whole delay vector of $s_n$ is compared with the delay vectors of each measurement in the phase state data i.e. for points 1 to $n-1$. To compare delay vectors of two points one could check the distance between corresponding elements of the two delay vectors. If each of the distances between the pairs are all within a desired distance then the two delay vectors are close.

Finding just the closest delay vector close to the $s_n$ is not enough. Any measurement of a continuous quantity makes sense within a finite boundary or resolution. The size of this limit or resolution depends on the differences arising from the measurement equipment and the fact that the values are eventually discretised. This implies that it is probable there would be more than one delay vector which would come reasonably close the delay vector of $s_n$. Each of these have to be considered as valid predictors of the future point. They have to be considered as neighbours of $s_n$ Every neighbour of $s_n$ will be used to predict a future value. The average of these predicted future values will be the finally accepted prediction

Thus the neighbourhood distance eps is another is another important parameter which has an effect on the success of prediction. This determines how many neighbours are found, which in turn effects the average predicted value.

This method is very inefficient if more than a few predictions are needed, e.g. because prediction error needs to be determined accurately to find the optimal set of parameters of time delay and dimension. For each prediction all points in the reconstruction are considered. All points are tested and most are rejected because they are too far away.

## 7 Prediction Technique with Box Assisted Neighbour Search

[10] Here will describe a method that will reduce this problem significantly by proposing a modification to the process of the repeated search for neighbours.

Creation of Phase State Map Array : An imaginary map with two axes $y$ representing $s_n$ and $x$ representing $s_{n-1}$ is drawn and subdivided into a number of boxes of equal sizes forming a grid. Each box represents a defined area of the phase state and each box is delimited by a unique set of lower and upper limits for both $y$ and $x$ coordinates. The size of the box will be defined by these limits may be varied. Every measurement taken is converted to the phase state form. Each phase state pair is then placed in the box according to the coordinate values.

Eventually all points are mapped into this grid. This grid can be represented in a number of ways. One of which is a simple array representing a list of pointers to actual data points. With a simple algorithm it is possible to sequence the array to reflect the boxes in the grid and create another pointer list to point at the start pointer and end pointer of each box. Another alternative, possible faster to run, is using a tree structure.

Reduced computation in finding neighbours: According to the prediction technique proposed earlier one has to scan the whole set of points for neighbours of the selected point. Using this phase state map of the points one can home in to the box in the grid related to the phase state of the selected point and determine which of the points in that box are actually neighbours. This would involve only a small subset of the total data set and would result in a huge reduction in computation cost.

Searching in neighbouring boxes: Some points may be neighbours of a selected point but will fall in a neighbouring box because they fall just outside the limit values. So one has to include the neighbouring boxes (8 in all) in the search for neighbours. Although this adds to the computation cost, overall there is still a significant gain over the previous method.

Defining an appropriate box size and number of boxes: A further condition to this approach is that the number of boxes has to be defined. If the boxes are too sparsely populated that there is little gain. If there are too few boxes but densely populated then there is little gain too. One may attempt to define the box size after examining the distribution of the phase state map. Once the box size and number of boxes are defined one will find that a number of points may refer to boxes outside the boundaries of the whole grid. In this case one has to mirror the coordinates of the 'outlying' point back into the grid using a suitable mod function. This is why one still has to check for closeness of each point in the selected boxes because there may be points which actually belong to a box outlying the grid.

Creating a list of neighbours: Once the array or tree representing the data set in phase state form is created it is a relatively smaller effort to create a list of neighbours for a selected point. In the case of a delay vector — each member of that delay vector is compared to the corresponding members of the potential neighbour point before deciding if the point being compared is a neighbour to the selected point.

Prediction using the list of neighbours: Once the list of neighbours is found the average predicted value for the future point or points can be determined. 8 Measuring success of prediction

[10] An important part of the prediction process is the measurement of the difference between the predicted value and the actual value. One can separate the data set into two parts — the training set and a hidden set to be used to test the success or not of the predictive technique being used. Usually a technique is used to predict values for a number of time points in the future. So it is necessary to first compute the error for each prediction and then carry out some form of summation to get a single number.

[21] These are some methods to calculate the degree of success of prediction, each of which contributes in a different manner to the evaluation of the prediction: mean error, mean absolute error, sum of squared error, root mean square error, percentage error, mean percentage error (MPE), mean absolute percentage error (MAPE).

Measurement of prediction error is important to improve on the prediction process by helping to identify the sets of input parameters which gave the best prediction results.

## 8 Number of Variables Used in Analysis

[10] Univariate time series (measurement of one observable) has been considered so far. Phase space has been reconstructed using delay vectors based on combinations of these measurements. However it is possible to form an embedding using a set of different variables obtained at the same time. One could think of other ways of representing the underlying system.

Multichannel measurements is analysis using multiple sources of simultaneous data — several measurements of the same type — at the same time — but at different segments of the process — which may give a more complete representation of the underlying system

Equivalent variables at different positions — In other situations one may find it useful to measure similar physical quantities simultaneously at different positions in a spatially extended system.

Variables with different physical meanings - One may also want to measure several observables with different physical meanings simultaneously. E.g. a 9-dimensional space with three different observables each at three different times.

Distributed systems multivariate time series are very frequent in physiology, economics or climatology, but are generally too complicated for a systematic investigation of the interrelation between the observables. There are research problems which are still unsolved. Take the example of a bivariate time series. There is a problem in deciding which observable to use in order to define the delay embedding based on one coordinate. Which observable yields more additional information?

## 9 Methodology

Forecasting methods: A number of methods exist which help the analyst to develop a forecasting model with a degree of objectivity, in a way which is measurable and reproducible.

These methods usually involve:

- the preparation of the data for analysis — smoothing, identification of cyclic (seasonal) components, trends, outliers, noise reduction;
- the application of a selected technique or set of techniques;
- measurement of forecasting error;
- refinement of parameters of model;
- updating of time series data as the current data becomes out of date;
- repeated evaluation of accuracy of model and refinement of model.

Standards and Practices for forecasting: [17] A set of standards is proposed by Scott Armstrong. From a detailed study in the use of standards a model with 139 points falling under these 16 categories was formulated, covering formulating problems, obtaining information, implementing methods, evaluating methods and using forecasts.

## References


1. Hossein Arsham, *Time-Critical Decision Making for Economics and Finance*. Available from `http://home.ubalt.edu/ntsbarsh/stat-dta/Forecast.htm`.
2. R. Bellazzi, C. Larizza, P. Magni, S. Montani and G. De Nicolao, *Intelligent Analysis of Clinical Time Series by combining Structural Filtering and Temporal Abstractions*, Artificial Intelligence in Medicine '99. LNAI 1620, 1999.
3. R. Bellazzi, C. Larizza, P. Magni, S. Montani and M. Stefanelli, *Intelligent Analysis of Clinical Time Series: an Application in the Diabetes Mellitus Domain*, AI in Medicine, vol 20, no. 1, 2000.
4. Michael Breakspear, *Perception of Odors by a Nonlinear Model of the Olfactory Bulb*, International Journal of Neural Systems, Vol 11, No.2:101-124, 2001.
5. Kavitha Chandra, Chun You, Gbenga Olowoyeye and Charles Thompson, Non-Linear Time-Series Models of Ethernet Traffic, CACT, Tech. Rep., June 1998.
6. Ana Beatriz Galvao, *Univariate Non-Linear Time Series Models and the Business Cycle Stylised Facts*, 2000.
7. Ben Goertzel, *Delay Vectors as Perceptual Chunks: Understanding Nonlinear Time Series Analysis as Self Organising Cognition* Dynamical Psychology An International, Interdisciplinary Journal of Complex Mental Processes, 1999.
8. R. Hegger, H. Kantz, and T. Schreiber *Practical Implementation of nonlinear time series methods: The TISEAN package*, CHAOS 9, 413, Winter School on Nonlinear Time Series Analysis, 1998.
9. B.P.T. Hoekstra, C.G.H. Diks, M.A. Allessie and J. DeGoede, *Non Linear Time Series Analysis: Methods and Applications to Atrial Fibrillation, Analysis and Processing of Cardiac Electrograms* in Atrial Fibrillation, Annali dell'Istituto Superiore di Sanita', Roma, Ref Ann. 1st Super. Sanita' vol 37, no.3, 2001.
10. H. Kantz and T. Scheiber, *Nonlinear Time Series Analysis*, Cambridge University Press, 1997.
11. Holger Kantz, *Time Series Analysis* in Reconstructed State Spaces, Stochastics and Dynamics, Vol 1, No. 1, 2001.
12. Kulkarni D. R., A. S. Pandya, and J. C. Parikh, *Dynamic Predictions from Time Series Data — An Artificial Neural Network Approach*, International Journal of Modern Physics C, in press, 1997.
13. Kulkarni D. R., A. S. Pandya, and J. C. Parikh, *Modeling and Predicting Sunspot Activity — State Space Reconstruction + Artificial Neural Network Methods*, GeoPhys Lett. 25, 457, 1998
14. P. Magni, R. Bellazi, and F. Locatelli, *Using uncertainty management techniques in medical therapy planning: a decision theoretic approach.* In Applications of Uncertainty Formalisms, Springer-Verlag, Berlin, 1998.
15. Pakalapati Rajani, *Network Traffic Analysis: Identifying Invariant Features in Ethernet Traffic*, M.Sc. Thesis, Dept of Elec. Eng. Univ of Massachusetts Lowell, 2000.
16. C. P. Price, D. Prichard, E. A. Hogenson, *Do the Sunspot Numbers Form a "Chaotic" Set?* Available from `citeseer.ist.psu.edu/price92do.html`.
17. J. Scott Armstrong, *Standards and Practices for Forecasting taken from Principles of Forecasting: A handbook for Researchers and Practitioners,* Kluwer Academic Publishers, 2001.
18. Sello Stefano, *Time Series Forecasting: A Multivariate Stochastic Approach*, Topic Note Nr. NSG/260199, Los Alamos National Laboratories Preprint Archive, Physics/9901050, 1999.
19. Sello Stefano, *Time Series Forecasting: A nonlinear Dynamics Approach*, Los Alamos National Laboratories, preprint archive Physics, 9906035, or Thermo Fluid Dynamics Research Centre Enel Research USG/180699.
20. Andreas Seyfang, Silvia Miksch, Werner Horn, Michael S. Urschitz, Christian Popow, Christian F. *Poets Using Time-Oriented Data Abstraction Methods to Optimize Oxygen Supply for Neonates*, Lecture Notes in Computer Science, 2001.
21. Statsoft Inc, *Time Series Analysis — a review of techniques Electronic Textbook*, Statsoft Inc. Available from `http://www.statsoftinc.com/textbook/stathome.html`.
22. Statsoft Inc, *Statistica, Product Information*. Available from `http://www.statsoftinc.com/downloads`.
23. David M. Walker, Nicholas B. Tufillaro and Paul Gross, *Raidal-Basis Models for Feedback Systems with Fading Memory*, IEE Transactions on Circuits and Systems, Vol.48, No. 9, 2001.
24. Chun You and Kavitha Chandra, *Time Series Models for Internet Data Traffic*, Proc. 24th Conference on Local Computer Networks, IEEE Press, p164-171, 1999.
25. Xiru Zhang and Kurt Thearling, *Non-Linear Time-Series Prediction by Systematic Data Exploration on a Massively Parallel Computer*, Available from `http://www./thearling,com/Text/sfitr.html`, Santa Fe Technical Report 94-07-045.

# The Use of Model-Checking for the Verification of Concurrent Algorithms

Joseph Cordina

Department of Computer Science and AI,
University of Malta

**Abstract.** The design of concurrent algorithms tends to be a long and difficult process. Increasing the number of concurrent entities to realistic numbers makes manual verification of these algorithms almost impossible. Designers normally resort to running these algorithms exhaustively yet can never be guaranteed of their correctness. In this report, we propose the use of a model-checker (SMV) as a machine-automated tool for the verification of these algorithms. We present methods how this tool can be used to encode algorithms and allow properties to be guaranteed for uni-processor machines running a scheduler or SMP machines. We also present a language-generator allowing the designer to use a description language that is then automatically converted to the model-checker's native language. We show how this approach was successful in encoding a concurrent algorithm and is able to verify the desired properties.

## 1 Introduction

Designing any new algorithm tends to be a strenuous and tough mental exercise. This is especially more difficult when designing concurrent algorithms due to their complex interaction. Commonly any such algorithm is painstakingly designed and is then dry-run to test for errors. Such errors normally tested for are deadlocks, memory value inconsistencies, etc. Dry-runs for concurrent algorithms involve testing each line of code when run in parallel with any other line of code in its concurrent counterpart. This ensures that no race-conditions could arise. Understandably this is a very long and difficult task especially since the problem grows exponentially with the number of concurrent tasks present within the system. These algorithms are normally tested in their assembly format, ensuring the atomicity of each instruction but making it more difficult for the designer to reason with. Often algorithms more than few lines long and running on more than a handful of concurrent tasks quickly become impossible to verify manually and thus designers tend to exhaustively execute them on hardware. While testing on hardware is seen as sufficient by most system software designers, the lack of certainty of the absence of errors makes such algorithms unsuitable for critical software. While novel concurrent algorithms can enhance the potential of computer hardware and software, the lack of proper execution guarantees makes work on such algorithms futile.

In this paper we propose the use of model-checking to facilitate and provide guarantees for the construction of concurrent algorithms. Such concurrent algorithms are not limited to be used on single-processor systems (making use of an un-deterministic scheduler) but also on synchronous and symmetric multi-processor algorithms. We also envisage this research to encompass un-synchronous multi-processor systems. In addition, we propose a meta-language, allowing the designer of the algorithms to concentrate on the construction of the algorithms itself and also widen the scope of this research to users that are not familiar with model-checking system.

## 2 Model-Checking

The term *Model-Checking* is very adequately described by Clarke and Emerson as '... an automated technique that, given a finite-state model of a system and a logical property, systematically checks whether this property holds for that model' [2]. In fact most modern model-checking systems are able to build a finite-state model of the system given a description language. In addition, they are then able to verify certain properties of the model using the same or some other description language. A brute-force model-checker merely traverses the state table for the possibility of ending up in a state that contradicts the property in hand. Yet the number of states quickly become too large for any hardware to handle. Thus research in model checkers aims to provide means and ways to verify properties while keeping the number of states small.

Research of the application of model-checking systems for concurrent algorithms is common. One interesting example is the work by Bauer et al. [6] that makes use of PVS [9] for the verification of concurrent algorithms when applied to dynamic memory. There is also a large number of model-checkers each suited for a particular verification domain. Very popular systems include Lustre [10] and Esterel [1], synchronous declarative languages for programming reactive systems; SPIN [5], a tool best suited for analysing data communication protocols and many others.

One of the reasons behind our work is the application of such model-checkers for the verification of algorithms that were designed within the Department of Computer Science and A.I. namely the wait-free algorithms proposed by Debattista [4] and Vella [13]. We would like to extend this research to be applied for CSP communication algorithms proposed by Vella [14, 12] yet this would require a different model-check system than we are concentrating upon here. For these reasons we have made use of SMV [7], a tool for checking finite state systems against specifications in the temporal logic CTL. It makes use of the OBDD-based symbolic model checking algorithm to control the state-explosion problem [8]. The SMV language was found ideal to verify concurrent algorithms since its very similar to imperative programming languages. In addition, the use of CTL temporal logic lends itself naturally to the specification of properties of these algorithms. SMV also has freely-available visual tools for both Windows and Linux systems making is easy to use. After parsing a specification, SMV will verify certain properties on demand and is able to present a counter-example to negated properties.

In our implementation we assume that the user will approach the system with a concurrent algorithm that has been reduced to its assembly equivalent[1]. The algorithm can be made up of different parts that can be executed separately. We have built two specifications in SMV that are able to verify concurrent algorithms. One of these is able to verify concurrent algorithms that are executed on a single processor system whereby a scheduler schedules instances of the algorithm in an undeterministic format. The second system specifies a multi-processor system that executes instances of the algorithm on each processor. Note that running the same algorithm on these systems is not guaranteed to have the same behaviour. One can envisage a scenario whereby two instances of one algorithm are run, whereby the algorithm takes a shared resource and releases it. On a fair multi-processor system, deadlock will never happen while in a single processor system, the first instance might never be de-scheduled when it does not have control of this resource. In such a case the second instance will be starved of the resource completely. We envisage these two systems to be integrated together thus allowing schedulable tasks on multi-processor systems.

[1] Most concurrent algorithms are normally designed in their assembly format anyway, otherwise one can simple use the assembler output of the original code.

### 2.1 Single Processor System

In this system, we have defined an SMV specification, such that each instance of a concurrent algorithm is executed within a task. Multiple tasks are then scheduled on the same CPU using a scheduler. In SMV, we have created the definition of a *task* which is made up of a task control block containing the CPU registers, the program counter and other CPU flags. In addition, we specified the list of tasks as an array of control blocks. We specified an input to the system that specifies which task to start executing. Every time the input changes, the previous state is stored within the appropriate task control block and the next one is started. To emulate the execution of instructions, we have hard-coded the end result of each instruction within the algorithm. Thus basically the current instruction that is executed (specified by the value of the current instruction pointer) will affect the next state in terms of Program Counter, registers, flags and other external memory locations.

Using such a system, since the input can change un-deterministically, SMV will verify all combinations of execution of each instruction. We also assumed that no more than one instruction can be executed at any one time, thus allowing a deterministic value for shared variables in the next state. A user of the system can encode concurrent algorithms that contain different independent parts as separate tasks.

### 2.2 SMP Systems

Using SMV, we have specified a symmetric multi-processor system that is able to execute instructions concurrently on each CPU. Unlike the single-processor system, we have not included a scheduling entity but we assume that once an algorithm starts, it will continue executing to termination (if it terminates). In this system, we have implemented the multi-processor system by specifying a series of registers and flags for each CPU that are independent of each other and that can change concurrently. To simulate the execution of the algorithm, we have specified an input trigger for each CPU and for each independent code section of the algorithm. This trigger will start a specific algorithm on one of the CPUs being used in the verification[2]. Using this system, we ensure that the start of the algorithm on a particular CPU is un-deterministic with respect to other CPUs. Thus SMV will verify all permutations of execution of the algorithm. Likewise to the uni-processor system, we have hard-coded each instruction so that when a particular CPU's program counter reaches a certain value, the next state of the registers and memory locations is determined. In our implementation we have assumed that all processors are synchronised in their execution cycles. In other words, one instruction on one CPU will take the same amount of time as another instruction on another CPU and not more than one instruction can execute within one clock cycle. While this is not strictly true in SMP systems, we felt confident in applying this assumption since deadlocks and memory consistency problems only occur when algorithms share data and on SMP systems access to shared variables is always synchronised (normally through the use of a *lock* instruction). Creating a system that is extensible to non-symmetric multi-processor systems would require removing this assumption, something that would not require major changes in our implementation.

The SMP specification has presented certain difficulties in implementing shared variables since one needs to guarantee the value of a shared memory location in the next clock cycle. We have tackled this situation by checking which instruction each CPU is executing and verifying what these instructions are. If only one instruction is modifying a shared memory value, then the value is deterministic. Otherwise we specify an un-deterministic value (from a range) to the shared

[2] If the trigger is enabled while the CPU is executing some code, the trigger is ignored.

memory location, allowing SMV to verify the algorithm for all possible values. Note that concurrent algorithms should ensure that values are always deterministic, and SMV clearly shows where these restrictions have to be applied.

## 3 Pre-Processing

While SMV presents a language that is easily-understandable, it can quickly become a daunting task to re-write complex algorithms using this language. To aid in the process of verification, we have created a description language that would enable the user to specify a concurrent algorithm which is then used to create an SMV specification.

Implementing several common concurrent algorithms in SMV, we have pinpointed several elements that such a description should contain. Apart from the algorithm itself, written in assembly format, the user should be able to provide general assumptions for the model in addition to properties that need to be proven. These assumptions and properties should be stated in temporal logic, a simple yet powerful construct. Additionally, we have provided the facility to create certain logical assertions depending on the algorithm's line number, thus enabling logical properties to be based on these assertions. We have made use of FLEX [11] to generate a parser and scanner for the algorithm description that is then able to generate a specification in SMV. Figure 1 shows the data flow in the process of verification. We envisage to create a parser for the SMV counter example output, allowing visual representation of the execution trace leading to the occurrence of the false assertion of a particular property.

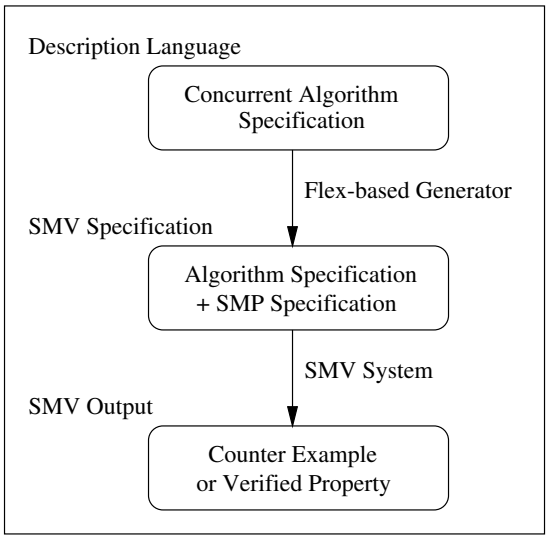



**Fig. 1.** Data Flow in the Verification Process

### 3.1 Results

What follows is a sample description file showing all characteristics making up an algorithm description together with all properties to be proven. This example is targeted to be used for SMP

systems. In this description we have specified a trivial yet valid concurrent algorithm, namely the *basic spin-lock algorithm with cache-invalidation* [3].

```
CPU 2

REGISTERS
carry:boolean;

MEMORY
lock:boolean;

STATEVARS
lockacquired:boolean;

LABELS
LABEL1 1
LABEL2 4
LABEL3 5

CODE
btsl lock
jne LABEL2
jmp LABEL1
nop

ASSERTIONS
4:lockacquired:=1;

TRIGGER
getlock:boolean;

CODE
btrl lock
nop

ASSERTIONS
2:lockacquired:=0;

TRIGGER
releaselock:boolean;

START
lock:=0;
lockacquired:=0;

PERCPUASSUMPTIONS
norelease: G(releaselock->lockacquired)

EXCLUSIVEINSTRUCTIONS
LABEL1
LABEL3
```

```
PROVE
mutex: G ~(&lockacquired[x]);
```

In the above description one can see how internal registers are specified (*carry*), external memory locations that might be shared (*lock*) and also the two code sections marking the acquire and the release stage of the shared lock. In addition labels used within the code are specified using a global line numbering starting from line number 1. One can also create state variables (*lockacquired*) stating those properties one wants to assert or check upon. For each code segment, one also needs to specify the variable that is used to trigger each particular code segment (*getlock* and *releaselock*). These variables are used as the input to the SMV module, thus enabling a non-deterministic execution start of the algorithm. In addition one can specify assertions for each code segment making use of the state variables specified previously. These assertions are executed upon reaching the given line number.

After specifying the algorithm itself, one can specify any initial values to variables using the `START` tag. There are also two forms of assumptions one can provide to the model. There are assumptions that are specific to each CPU, in other words, they are assumptions that are specific to the code itself (in our case we specify that the code corresponding to releasing the lock is only executed if one owns the lock[3]). In addition, one can specify interactions between different CPUs (using the `EXCLUSIVEINSTRUCTIONS` tag). Since the only interaction between the different CPUs on standard SMP systems is specifying those instructions that cannot access memory concurrently, we have opted to allow the user to specify labels indicating instruction position within the code. Consequently, SMV will ensure that instructions specified here will not be allowed to executed concurrently, but will introduce a one time step delay when this happens. Finally one can specify the properties that needs to be proven. We have introduced a special syntax allowing proofs to vary over particular CPUs or all of them. In our example we state that *lockacquired[x]* is expanded to the AND of all *lockacquired[x]* where $x$ varies over all CPU indices.

In our example it turns out that *mutex* can be verified in SMV using 14 state variables with 4 combinatorial variables for 2 CPUs. This rises to 21 state variables with 6 combinatorial variables for 3 CPUs. While the number of state variables rapidly increases with the number of CPUs making verification difficult for a large number of CPUs, at least we do have some guarantee properties to work with for a small number of CPUs. To have complete confidence that the verification applies to $n$ CPUs one can then apply some other technique such as proof by induction to arrive to a satisfactory verification.

While the example given is quite trivial, much more complex algorithms can be verified using our tool. All that is required to verify wait-free algorithms is to encode within the FLEX specification the novel instructions that are used within these algorithms. Then any algorithm can be specified and quickly verified given that the state space is not huge[4].

## 4 Conclusion

In this report we have highlighted the difficulties normally encountered by designers of concurrent algorithms to ensure the correctness of these algorithms. We have pinpointed a tool (SMV) that is

[3] While it is normally assumed that one never calls release unless one has the lock, SMV quickly picks up on the gross negligence in this assumption and shows how this simple algorithm fails when this is not assumed.

[4] Normally concurrent algorithms are quite short in the number of instructions, thus ensuring a limited state space and thus verification can be completed within a reasonable amount of time within SMV.

able to verify these algorithms and we have constructed an implementation that would satisfactorily verify these algorithms for both uni-processor machines running a non-deterministic scheduler and a multi-processor machine. In addition we have presented a description language using which a designer can specify his algorithms without having to resort to the use of the SMV language. Using our parser this description is automatically translated to its SMV counterpart which in its turn is directly given to the SMV system.

While this work is still in its infancy and many more improvements can be applied, we are confident that using these ideas, many complex concurrent algorithms can be verified and guarantees reasonably given on these algorithms.

## References


1. Gérard Berry. The foundations of esterel. In *Proof, Language and Interaction: Essays in Honour of Robin Milner, G. Plotkin, C. Stirling and M. Tofte*. MIT Press, 1998.
2. E.M. Clarke and E.A. Emerson. Design and synthesis of synchronisation skeletons using branching time temporal logic. In *Logic of Programs, Volume 131 of Lecture Notes in Conputer Science*, pages 52–71. Springer-Verlag, 1981.
3. J. Cordina. Fast multithreading on shared memory multiprocessors. B.Sc. I.T. Final Year Project, Department of Computer Science and Artificial Intelligence, University of Malta, June 2000.
4. K. Debattista. High performance thread scheduling on shared memory multiprocessors. Master's thesis, University of Malta, February 2001.
5. Gerald J. Holzmann. *The Spin Model Checker: Primer and Reference Manual*. Addison Wesley Publishing Company, Sep 2003.
6. R. Nickson K. Bauer, L. Groves and M. Moir. Machine-assisted reasoning about concurrency and dynamic memory. Victoria University of Wellington, School of Mathematical and Computing Sciences, December 2001.
7. K. L. McMillan. *Symbolic model checking - an approach to the state explosion problem*. PhD thesis, SCS, Carnegie Mellon University, 1992.
8. K.L. McMillan. The SMV system. Cernegie Mellon University, November 2000.
9. S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In Deepak Kapur, editor, *11th International Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, Jun 1992. Springer-Verlag.
10. N. Halbwachs P. Caspi, D. Pilaud and J. Plaice. Lustre: a declarative language for programming synchronous systems. In *14th ACM Conf. on Principles of Programming Languages.*, Munich, January 1987.
11. Vern Paxson. Flex, version 2.5. URL http://www.gnu.org/software/flex/.
12. K. Vella. CSP/occam on networks of workstations. In C.R. Jesshope and A.V. Shafarenko, editors, *Proceedings of UK Parallel '96: The BCS Parallel Processing Specialist Group Annual Conference*, pages 70–91. Springer-Verlag, July 1996.
13. K. Vella. *Seamless Parallel Computing on Heterogeneous Networks of Multiprocessor Workstations*. PhD thesis, University of Kent at Canterbury, December 1998.
14. K. Vella and P.H. Welch. CSP/occam on shared memory multiprocessor workstations. In B.M. Cook, editor, *Proceedings of WoTUG-22: Architectures, Languages and Techniques for Concurrent Systems*, volume 57 of *Concurrent Systems Engineering*, pages 87–120. IOS Press, April 1999.

# TTS Pre-processing Issues for Mixed Language Support

Paulseph-John Farrugia[1]

Department of Computer Science and AI,
University of Malta

**Abstract.** The design of an open domain Text-to-Speech (TTS) system mandates a pre-processing module that is responsible for preparing the input text so that it can be handled in a standard fashion by other TTS components. There are several pre-processing issues that are common across input domains, such as number, date and acronym handling. Others may be specific to the type of input under examination.

In designing a TTS system supporting Maltese for SMS messages in the local context, specific issues are also encountered. In terms of language issues, the practical use of Maltese is heavily characterised by *code-switching* into other languages, most commonly in English. While the resulting language may not be considered 'Maltese' in the strictest sense of the language definition, it creates a state of affairs that cannot simply be ignored by a general purpose system. In respect of the SMS domain, the use of various shorthand notation and lack of phrase structure is encountered.

This paper describes these specific issues in further detail and discusses techniques with which they may be addressed.

## 1 Introduction

The human reading process (an overview of whose intricacies is given in [1]) is quite a complex one. It is simplicity with which the human mind can easily process even heavily misspelt texts is impressive. Fig. 1 provides an interesting intuitive example of this. Implementing the same kind of flexibility programmatically for a TTS system, however, is not a simple task.

> The following text is taken from an email commonly forwarded around the Internet:
>
> "*Aoccdrnig to rscheearch at an Elingsh uinervtisy, it deosn't mttaer in waht oredr the ltteers in a wrod are, olny taht the frist and lsat ltteres are at the rghit pcleas. The rset can be a toatl mses and you can sitll raed it wouthit a porbelm. Tihs is bcuseae we do not raed ervey lteter by ilstef, but the wrod as a wlohe. Fnnuy how the mnid wroks, eh? . . .*"
>
> Irrespective of the validity of its message, it does provide an amusing example of the facility with which the human mind can easily interpret and read out even heavily misspelt texts.

**Fig. 1.** Reading Misspelled Text

[1] This work is supported by MobIsle Communications Ltd. (http://www.go.com.mt)

Handling part of this task is the role of the *pre-processing* module. This is the first stage of input processing, organising the input text into a standard format that the following modules can process more easily ([2]). Amongst other things, it is generally responsible for converting numerals and acronyms into their textual interpretation and resolving punctuation. In specialised systems, other tasks may be required of it. For example, in an email to speech system, the pre-processor will be required to remove unnecessary email headers and superfluous formatting. (See [3] for examples.)

In designing a TTS system supporting Maltese for SMS messages in the local context, pre-processing issues specific to this domain are encountered. These will be illustrated in some detail and techniques with which they may be addressed will be discussed.

## 2 Mixed Language Support

In the domain of concern, the specific pre-processing issues arise out of the mixed language nature of the text, the mechanisms used to artificially shorten the messages, orthographic errors and device limitations. The two main concerns, *code switching* and character encoding support, are discussed below. Another pre-processing issue is the handling of *smileys*, such as `;->`, and other abbreviations typical of the SMS domain. This however, can be handled by a simple rewrite process from well-known lists of such common shortcuts.

### 2.1 Bilinguality and Code Switching

The Maltese Islands are officially bilingual, with both Maltese and English as official languages of the country. English is generally understood as British English, while an official version of Maltese, often termed as *Standard Maltese* (see [4] and [5]) is commonly taught in schools. The latter is distinguished from dialects of Maltese that may be found in certain villages and localities, which differ subtly at various levels of structure (including syntactic, lexical, grammatical and intonational.)

In practice, however, this bilingual situation brings forth a heterogeneous mixture of language use. For various geographical and sociological reasons ([4]), the use of Maltese, whether spoken or written, is often interspersed with words or phrases in English[2], a phenomenon known as *code switching*. Conversely, the local use of English is often marked by certain grammatical and phonetic differences, to the extent that some may want to refer to it as a "Maltese English" dialect ([6]).

For an appreciation of the above, consider, for instance, the following (real-life) SMS text message sample:

> *"jaqaw bdejt tibza tixel il car? xorta qajjimt il qattusa u issa qed tajjat wara il bieb. bring that btl of last time plz qalbi :)"*

The extent to which such code switching is acceptable as proper use of the Maltese language (or whether "Maltese English" may be truly considered an English dialect as opposed to a result of a lack of linguistic sophistication) is a matter of near religious debate ([6]). Rather than going into the respective merits of such considerations, a pragmatic approach is preferred whereby this state

[2] And, to a lesser extent, with words in other languages, most typically in Italian.

of affairs is addressed and techniques that can help solve the issues that arise are identified. In this case, a means is required to identify the switching from one language to another at the word level[3].

The intuitive solution would be to resort to a lookup process within extensive dictionaries for the languages involved. This approach by itself, however, is bound to have limited success. Firstly, in an open domain, no dictionary is ever expected to be complete, and hence the resolution for unknown words remains a problem. Secondly, SMS messages often contain a variety of spelling mistakes, rendering a lookup into a classic dictionary prone to failure. However, it is not excluded that the use of dictionary can be a supplementary aid to the language tagging process.

Attention is changed, then, to stochastic approaches for identifying the language of a text, of which two main trends can be identified ([7].) The first involves the use of frequencies of occurrence of short words (such as *u* and *ta* in Maltese) in order to determine the general language of a given corpus. This is not a directly applicable approach here, since the interest lies in identifying the language of origin of each individual word.

A second technique, which has been used for language ([8]) and subject ([9]) classification, is based on *n-gram* probability matching on characters. In essence, for each of a candidate set of languages, an n-gram probability profile is created from a training corpus. Input text is then classified according to the profile that best matches its n-gram probabilities.

An extension is hereby proposed to handle the issue of code switching. Assuming *bigrams* are being used, the procedure can be expressed as follows. Let $\mathbf{L} = \{L_1, L_2, \ldots, L_n\}$ be a set of $n$ candidate languages. For each $L_i$, let $C_i = \{c_1, c_2, \ldots, c_m\}$ be the set containing those characters belonging to the language $L_i$, taken from a universal set of characters $\mathbf{C}$.

Given any significantly large corpus known to be entirely (or primarily, since the interest here lies in statistical significance) in a language $L_i$, it is possible to compute $0 \leq P_{L_i}(a, b) \leq 1$, the probability that the bigram $ab$ occurs in free text written in the language $L_i$, where $a, b \in \mathbf{C}$. The probability of a word $\mathbf{w} = w_1 w_2 \ldots w_k$, where $w_j \in \mathbf{C}$, belonging to $L_i$ can then be defined as:

$$Pw_{L_i}(\mathbf{w}) = \prod_{j=0}^{k} P_{L_i}(w_j, w_{j+1})$$

where $w_0$ and $w_{k+1}$ are defined as the equivalence class of characters in the set $\mathbf{C} \setminus C_i$, that is, those characters not belonging to language $L_i$ (which include spaces and punctuation.) The most likely classification of $\mathbf{w}$, then, is that it belongs to the language $L_i$ that maximizes $Pw_{L_i}(\mathbf{w})$ over all languages in $\mathbf{L}$. In practice, it would also be possible to take the probabilities of the surrounding words for further disambiguating power.

### 2.2 Character Set Support

Another issue to be tackled arises from the character set required to encode the Maltese alphabet. Proper writing of Maltese mandates the use of certain characters, namely *ċ*, *Ċ*, *ħ*, *Ħ*, *għ*, *Għ*, *ż*, *Ż*, that are not found in the ASCII character set. Electronic input devices—from keyboards to mobile phones—in the local market are usually intended for an English audience (perhaps supporting some other main European languages), and the support for the full Maltese character set ranges from

[3] Ideally, this should be at the morpheme level, since it is also common to process loan words using Maltese morphological rules. Also, to a certain extent, it may be desirable to phonetize other orthographic elements, such as numerals, in another language, in order to conversely mimic this tendency toward code-switching.

limited (as is the case with most common computer keyboards in use[4], for which specific shortcuts may be required) to non-existent (as in the case of mobile phones.) Given the use of English as a secondary language, this is generally not seen as a major usability issue, and the motivation to have such devices enabled is somewhat limited.

While the advent of Unicode ([10]) is now helping to provide a reference format standard for Maltese text processing (although it still poses some computational issues, as it does not admit the representation of digraphs, such as *ie* and *għ*, as single character units), the situation regarding electronic texts at this point is really split in two. As regards official documents or documents meant for public access, schemes have been utilised to represent the above characters using the ordinary ASCII encoding. The most typical is the use of particular fonts designed for Maltese typesetting (but incompatible with Unicode) whereby the glyphs of the above characters replace the entries for certain punctuation characters ([]). Under other schemes, such as that used in Maltilex ([11]), character escape sequences are used instead.

On the other hand, where the presentation of text is not critical, it is often the case that letters containing diacritical marks are written instead with those ASCII character counterparts that do not, which, corresponding to the above, would be *c*, *C*, *h*, *H*, *gh*, *Gh*, *z* and *Z*. Despite what would constitute intrinsic spelling mistakes, a native speaker can immediately identify and interpret the underlying message[5]. However, such text cannot be directly operated upon prior to resolving the resulting lexical ambiguities.

Resolving this issue is an exercise in spell checking, albeit a simplified one. One could envisage resolving it using a threefold approach (starting from the assumption that the word has already been tagged as a Maltese one.) First, certain re-write rules can be applied. For instance, *c* can be safely re-written as *ċ* in all cases, since the previous character does not exist in the Maltese alphabet. Secondly, use of a dictionary can sort out some entries, as it will list *żarbun* but not *zarbun*. Finally, one could consider using stochastically based re-write rules trained on a corpus written in two fashions, once in the appropriate encoding and one using the ASCII character set only.

## 3 Language Classification Implementation

In the following, the process of developing a language classification mechanism for differentiating between Maltese and English words, based upon the ideas from the previous section, is described. Such a mechanism would form part of a more complete pre-processor module.

### 3.1 Corpora and Bigram Calculation

In order to estimate the n-gram probabilities, suitable corpora for the languages under consideration is required. In order for the probabilities to be meaningful, the corpora are required to be substantially large, and ideally from the same domain as the text that needs to be classified.

Unfortunately, corpora consisting solely of SMS messages already organised as Maltese or English are not readily available, and deriving substantially sized ones would be a very time consuming exercise. An alternative textual corpus is available in the form of the Laws of Malta ([12]). These provide a suitably large and balanced corpora for both Maltese and English. (The fact that the

[4] A Maltese keyboard standard has been defined but is not in widespread use.

[5] Essentially, this is a more natural occurrence of the phenomenon illustrated in Fig. 1.

corpora are translations of each other is of peripheral interest here. However, it is useful in providing balance for the frequencies being calculated.)

The laws are available in PDF format, and hence not directly suitable for machine processing. The text contents were thus extracted to plain text files in order that they may be easily processed programmatically. As a result of the extraction process, the files were not completely coherent with respect to the originals, containing some spurious symbols and characters (such as used for formatting or page numbering). However, given that these made up only a very small percentage of the overall corpora, they make negligible effect on the overall frequency calculation results.

The Maltese laws seem to have been encoded with the legacy 'Tornado' fonts which necessitated a mapping from the replaced glyphs to the proper Maltese characters in the Unicode set. The text files were saved in UTF-8 format. Another version of these files was created, where the non-ASCII Maltese characters were replaced by their ASCII counterparts. This was necessary, since, for the reasons explained in the previous section, in the short to medium term, textual input from mobile devices is expected to contain this ASCII version of Maltese. Unless the n-grams are calculated on the ASCII based frequencies, language identification would be expected to show a preference towards English.

An application was written to extract the bigram frequencies from the corpora thus created and save them into an XML file format. For Maltese, the bigram frequencies were calculated on both the Unicode and ASCII text corpora. These were then used as the basis for a language classification application that tags whitespace separated tokens according to the maximal $Pw_{L_i}(\mathbf{w})$.

### 3.2 Preliminary Results

The language classifier thus created was used to identify a set of random SMS messages[6] consisting of 1000 whitespace separated tokens (mostly words or numbers, but also containing shortcuts and punctuation which were processed in the same manner.) The results were then hand-checked to verify correctness. With this basic, unpolished, process, a 76% accuracy ratio was obtained.

While being correct three fourths of the time seems a reasonable start, analysing the results one finds that there is room for substantial improvement in the approach. Firstly, given that no attempt at pre-filtering the input was done, some of the cases (such as smileys and punctuation) could not be clearly classified English or Maltese. In this case, the tagging was considered as incorrect, but it is expected that in future revisions a pre-filtering process would take care of these tokens. Secondly, a certain amount of words were actually in languages other than Maltese and English. Again, these were marked as incorrect, even though they contradict the implicit assumption that the input can be accordingly classified.

However, certain misidentification errors do arise out of the limitations of the probabilistic approach taken. One of the notable problems was the word 'I', necessarily an English one, being consistently[7] identified as Maltese. This arises out of the characteristics of corpus chosen for frequency calculation, which would hardly contain any instance of this word, even though in general English it is quite a common one. To solve this, it is necessary to improve the n-gram frequencies by calculating them from corpora that contain this word with a reasonable frequency.

[6] To preserve anonymity, all header information, including sender number, receiver number and timestamps, were stripped off prior to usage, and only the message text was maintained. Messages were also sequentially randomized, removing any implicit temporal information.

[7] Obviously, this basic approach does not permit otherwise.

Another prominent issue was the tagging of the word 'u' as Maltese. This word is in fact used both as a shortening of the English word 'you' and also as the proper form of the Maltese 'u' ('and'). In this case, the error is not a result of incorrect training and it is necessary to take the surrounding context into consideration in order to resolve the ambiguity.

### 3.3 Conclusions and Indications for Further Work

The implementation described above provides a a basic infrastructure for tackling the problem of code switching. However, it is clear that there is still room for further improving the results. One possible improvement would be to further train the language classificator by calculating frequencies on text from the domain concerned itself. Creating the corpus could now be simplified by, for instance, using $|Pw_{Maltese}(\mathbf{w}) - Pw_{English}(\mathbf{w})|$ of the tagged text as a confidence level indicator, and consequently manually check only those results which fall below a specific threshold. In this manner, one should expect the results to improve iteratively.

Still, the basic approach fails to account properly for words that can occur in more than one of the languages under consideration. Identifying the situation by itself necessitates a dictionary lookup, but resolving it would require taking the surrounding context into consideration. However, since the surrounding language context is itself statistically inferred, it is possible to imagine a multi-pass approach that resolves ambiguities left open in previous passes. How to best account for context in this manner still remains to be determined.

## References

1. Dutoit, T.: An Introduction to Text-To-Speech Synthesis. Volume 3 of Text, Speech and Language Technology. Kluwer Academic Publishers, P.O. Box 322, 3300 AH Dordrecht, The Netherlands (1997)
2. Farrugia, P.J.: Text-to-speech technologies for mobile telephony services, University of Malta, Computer Science Department (2003)
3. Black, A.W., Taylor, P., Caley, R.: The Festival Speech Synthesis System. University of Edinburgh. 1.4 edn. (2002)
4. Borg, A.: Ilsienna. Albert Borg (1988)
5. Borg, A., Azzopardi-Alexander, M.: Maltese. Descriptive Grammars. Routlege, London and New York (1997)
6. Vella, A.: Prosodic Structure and Intonation in Maltese and its Influence on Maltese English. PhD thesis, University of Edinburgh (1995)
7. Grefenstette, G.: Comparing two language identification schemes. In: 3rd International Conference on Statistical Analysis of Textual Data, Rome (1995)
8. Beesley, K.: Language identifier: A computer program for automatic natural-language identification of on-line text. In: Languages at Crossroads: Proceedings of the 29th Annual Conference of the American Translators Association. (1988) 47–54
9. Cavnar, W.B., Trenkle, J.M.: N-gram-based text categorization. In: Proceedings of (SDAIR)-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, US (1994) 161–175
10. Unicode homepage. (http://www.unicode.org)
11. Maltilex homepage. (http://mlex.cs.um.edu.mt)
12. Laws of Malta Homepage. (http://justice.gov.mt)

# Low-Latency Message Passing Over Gigabit Ethernet Clusters

Keith Fenech

Department of Computer Science and AI,
University of Malta

**Abstract.** As Ethernet hardware bandwidth increased to Gigabit speeds it became evident that it was difficult for conventional messaging protocols to deliver this performance to the application layer. Kernel based protocols such as TCP/IP impose a significant load on the host processor in order to service incoming packets and pass them to the application layer. Under heavy loads this problem can also lead to the host processor being completely used up for processing incoming messages, thus starving host applications of CPU resources. Another problem suffered by inter-process communication using small messages is the latency imposed by memory-to-memory copying in layered protocols as well as the slow context switching times in kernel-level schedulers required for servicing incoming interrupts. All this has put pressure on messaging software which led to the development of several lower latency user-level protocols specifically adapted to high-performance networks (see U-Net[18], EMP[16], VIA[3], QsNET[15], Active Messages[19], GM[13], FM[14]).
The aim of this paper is to investigate the issues involved in building high performance cluster messaging systems. We will also review some of the more prominent work in the area as well as propose a low-overhead low-latency messaging system to be used by a cluster of commodity platforms running over Gigabit Ethernet. We propose to use the programmable Netgear GA620-T NICs and modify their firmware to design a lightweight reliable OS-bypass protocol for message passing. We propose the use of zero-copy and polling techniques in order to keep host CPU utilization to a minimum whilst obtaining the maximum bandwidth possible.

## 1 Introduction

The performance of off-the-shelf commodity platforms has increased multi-fold over the past two decades. The same can be said for the continuous increase in demand for computational speed by modern applications, especially in areas involving scientific and engineering problem simulations and numerical modeling. In most cases such computation can be performed by multi-threaded applications running in parallel on separate processors. The increased availability of high bandwidth networking hardware which is currently capable of transmitting at gigabit/sec speeds has enabled the development of Networks of Workstations (NOWs) [5] which are turning out to be more affordable replacements to single large-scale computers.

The inclination towards the use of clustered computing for high performance computation has put forward new requirements that cannot be delivered by conventional communication systems used in commodity platforms. The major challenge in using NOWs for parallel computation is that the communication architecture should be as transparent as possible to the user-level applications and must not act as a performance bottleneck. Thus, the aim is to provide a high-throughput and low-latency message passing mechanism that enables fine grain multi-threaded applications to communicate efficiently whilst keeping processing overhead on host machines to a minimum.

Although commodity networks are still dominated by IP over Ethernet, these protocols were not intended for networks running at gigabit speeds due to several inefficiencies that will be discussed later. This led to the development of other interconnect architectures such as Myrinet [2], SCI [10], Giganet cLAN [7], Quadrics [15] and Infiniband [1] which were targeted towards achieving higher bandwidth reliable communication without incurring the processing overheads of slower protocols. Our work will be geared towards commodity platforms running over a Gigabit Ethernet architecture, since these are one of the most common and affordable cluster architectures available.

## 2 High Performance Networking Issues

Various performance issues arise when utilizing the standard Ethernet framework for gigabit communication within a cluster of commodity platforms. Although modern Ethernet hardware promises gigabit line-rates, this performance is hardly ever obtainable at the application layers. This is due to several overheads imposed by interrupt generation [8] as well as conventional layered communication protocols such as TCP/IP which although provide reliable messaging, also add significant protocol processing overheads to high bandwidth applications. First of all we will take a look at the steps involved in sending messages over a network using conventional NICs and generic protocols such as TCP/IP.

### 2.1 Conventional Ethernet NICs

Conventional network cards are cheap and usually characterised by little onboard memory and limited onboard intelligence which means that most of the protocol processing must be accomplished by the host. When the Ethernet controller onboard the NIC receives Ethernet frames, it verifies that the frames are valid (using CRC) and filters them (using the MAC destination address). When a valid network packet is accepted, the card generates a hardware interrupt to notify the host that a packet has been received. Since hardware interrupts are given a high priority, the host CPU suspends its current task and invokes the interrupt handling routine that will check the network layer protocol header (in our case being IP). After verifying that the packet is valid (using a checksum) it also checks whether that particular IP datagram is destined for this node or whether it should be routed using IP routing tables. Valid IP datagrams are stored in a buffer (IP queue) on host memory and any fragmented IP datagrams are reassembled in the buffer before calling the appropriate transport layer protocol functions (such as TCP/UDP). Passing IP datagrams to upper layer protocols is done by generating a software interrupt and usually this is followed by copying data from the IP buffer to the TCP/UDP buffer.

### 2.2 Problems with Conventional Protocols

Since traditional operating systems implement TCP/IP at the kernel level, whenever an interrupt is generated the host would incur a vertical switch from the user to the kernel level in order to service the interrupt. This means that for each packet received the host would experience several switching overheads. Also due to the layered protocol approach, each packet would require several memory-to-memory buffer copies before it reaches the application layer. Both memory-to-memory copies as well as context switching (between kernel and user space) are expensive to the host CPU which means that the higher the rate of incoming packets, the more host processing power is required to service these packets.

Servicing these interrupts on a 100Mbps fast ethernet network would be manageable by the receiving host using around 15% CPU utilization, however when projecting this to a 1Gbps network using the same host we would easily get to 100% CPU utilization before reaching 600Mbps [8]. This weakness in traditional protocols can lead to a *receiver livelock* scenario which can be easily exploited to cause denial of service attacks. This scenario is brought about since hardware interrupts generated by incoming packets are given a higher priority than software interrupts generated by higher layer protocols. Also user level applications which are waiting for the data are given the lowest priority. This means that under high network load the applications processing the incoming data would be continuously interrupted by incoming packets and thus would not have enough processing time available to service the incoming data. Unserviced packets would remain waiting in the respective protocol queues and this could easily result in the overflow of protocol buffers. In case of overflow, all newly received packets would be discarded, however these would still be using up host processing time due to the interrupts they generate. This would result in the host machine consuming all its processing resources for servicing and discarding incoming packets while starving user applications from CPU time.

### 2.3 Current Solutions

There have been several attempts to achieve better throughput while maintaining low CPU utilization. U-Net [18] was one of the first systems that removed the kernel from the communication path and enabled user-level access to the communication hardware which in their case worked over ATM. This OS-bypass mechanism was also exploited by Myrinet GM [13] which uses a technique to allocate and register a common chunk of host memory which is then used by the hardware for sending/receiving messages to/from user-space. Myrinet [2] interfaces have an onboard processor that continuously loops through what is referred to as a *Myrinet Control Program* that allows user-applications to communicate with the hardware without using system calls, thus eliminating the need for vertical switches during communication.

Projects like U-Net also make it possible for protocol processing to move into user-space. Thus, unlike traditional kernel based protocols, U-Net delivers flexibility to applications that want to customize their own protocols and interfaces tailored to application specific requirements. U-Net also moves all buffer management to user-space which gives the illusion that each process owns the network, however it also gives rise to issues dealing with data protection between processes using the same network. U-Net also supports a zero-copy architecture in which data can be passed between the application data structures and the network interface without any intermediate buffering. In order not to involve the kernel in the critical path, U-Net uses an interrupt-free mechanism and transmits data by pushing message descriptors onto a queue. A polling mechanism enables the NIC to retrieve the data from the buffer and transmit it over the network. Receiving messages is done in a similar way using events.

Other performance enhancements can be achieved through interrupt coalescing so that hardware interrupts are not generated for each received packet but packets are batched together and passed to the host in larger chunks, thus incurring less DMA startup overheads. Interrupts can be generated when an incoming buffer treshold is exceeded or when a timer expires or else using a hybrid approach between these two. On switches and NICs that support a non-standard maximum transmission unit (MTU) we can instruct the NIC to use larger MTUs (9KB Jumbo frames) instead of the standard 1.5KB Ethernet frames in order to reduce the number of transmitted Ethernet frames as well as reduce the interrupts generated when using applications requiring high throughput.

The techniques used by U-Net, Myrinet GM and other architectures such as QsNet [15] were only possible by utilizing programmable network interface cards. These type of cards having one or more onboard processors and memory which make it possible for them to interact directly with

user-applications and thus offload message passing overheads from the host. Another overhead that may be alleviated from the host processor is that of calculating checksums for each transmitted network packet. Making use of the NIC CPU it is possible to instruct the firmware on the card to perform these checksums before passing them on to the host application.

One of the first host-offloading systems to be developed for the Gigabit Ethernet architecture was EMP [16]. This was the first NIC-level implementation of a zero-copy OS-bypass messaging layer that used programmable NICs to perform the entire protocol processing onboard the NIC. EMP made use of the NIC's DMA facilities for transferring packets to/from the host, thus allowing the CPU to remain available for application processing.

Several issues arise when using DMA to copy data directly from the user-space to the NIC memory or vice-versa without any kernel interaction. First of all user applications make use of virtual memory addresses provided by the OS whilst the DMA engine on the NIC requires physical memory addresses in order to perform data transfers to/from the host. Thus before using DMA, the architecture is required to implement some sort of mechanism for mapping virtual to physical addresses. One way of doing this is by keeping a copy of the kernel page-tables on the NIC however due to the limited memory onboard the NIC this would impose constraints on the amount of memory that can be mapped. Another option is to have the user application inform the NIC about the physical addresses of the pages involved in the transfer. Since user-space does not have access to physical memory addresses, these would have to be retrieved from the kernel through a system call. This is the technique used by Shivam *et al.* in EMP who also argue that it is not too computationally expensive to perform such system calls for translating memory addresses. Another issue arising when DMAing directly to/from user-space is that modern operating systems use complex Virtual Memory management mechanisms in order to allow applications to use more memory than physically available. This allows memory pages to be paged out from physical memory and stored in virtual memory automatically by the OS. This would provide a problem for the NIC DMA engine if pages are removed from memory whilst a DMA transfer is taking place. To solve this EMP make use of UNIX system calls that instruct the OS not to allow paging for specific memory regions. Such memory locking would allow zero-copy message transfers between the NIC and user-space. Tezuka *et al.* [17] also implemented a zero-copy message transfer in their Myrinet PM network driver which they call *Pin-down Cache*. Their implementation provides an API that allows the user application to pin-down and release physical memory to be used for zero-copy messaging. However pinned down areas are managed by the system and are reused in order to reduce the number of pin-down and release primitives as well as to protect physical memory from being exhausted by malicious users.

Another issue to consider when using DMA for data transfers is that the pages involved in the transfer are not always aligned in one contiguous block which compels a problem for the DMA engine which uses physical address. To avoid expensive buffer alignments prior to using DMA data transfers we can make use of scatter/gather techniques where fragmented packets can be retrieved from different parts in memory without having to perform additional memory copies to align the buffer data into a contiguous chunk. This can be accomplished by performing multiple DMA transfers for each packet.

Each time a transfer occurs through the PCI bus, a startup latency is inherently induced by the bus, however once the transfer is started there are no further overheads unless the bus operations are interrupted by the host CPU or other connected devices [9]. Consequently increasing the size of data transfers over the PCI bus would lead to a higher bandwidth due to less setup latency as noted in tests done by Cordina [4] and Wadge [20]. On similar lines, Gilfeather and Underwood [8] proposed transferring larger MTUs (64K) over the PCI bus in order to reduce interrupt overhead. This was done by modifying the NIC firmware to enable fragmentation of large datagrams into smaller frames on the NIC before being transmitted over the network. On the receiving side, the NIC re-assembles the datagrams before sending them through the host PCI bus.

### 2.4 Short messaging

Traditional network performace is usually measured by the bandwidth achieved for continuous streams, however many applications relying on short messages are more sensitive to the communication latency which can be described as the round-trip time for a message to travel between two processes running on networked nodes. This is the case with fine-grain multi-threaded applications which communicate very frequently by sending small messages usually under 1KB. There have been several attempts to minimize the latency incurred by small message passing over a cluster. One such attempt is Illinois Fast Messaging FM [14] for Myrinet which offers a low-latency reliable messaging layer designed specifically for sending small messages over tightly coupled clusters.

Intel[1], Microsoft[2] and Compaq[3] have joined forces in an effort to standardize an architecture for the interface between computer systems and high performance hardware. First drafted in 1997, the Virtual Interface Architecture (VIA) [3] aims to improve the performance of distributed applications by reducing the software processing required to exchange messages. The VI architecture can be said to have been built upon U-Net, FM and Active Messages [19] and provides each consumer process with a protected and directly accessible virtual interface (VI) to the shared network hardware. The VIA only uses the OS to setup data structures and mappings for the user-level network (ULN) so the kernel can be removed from the critical path. This implies that the network adapter must have a certain degree of intelligence in order to perform all the data transfer scheduling and de/multiplexing usually performed by the OS. VIA uses a system of buffer descriptors and doorbells to pass messages between the NIC and user-space. This event driven system uses polling to avoid interrupt and system call overheads for message passing.

### 2.5 The Alteon Acenic Tigon 2 NIC

The work done by Gilfeather and Underwood [8], EMP [16], Cordina [4] and the tests done by Wadge [20] were all implemented by using Alteon Networks' Acenic[4] [12, 11] NIC which is a PCI programmable Gigabit Ethernet card having two onboard 32-bit/88 MHz RISC processors. Unlike ROM based NICs, the firmware is uploaded onto the card during initialization which means that it can be modified and reloaded onto the card at any time. The card also has 1MB of SRAM which is used for holding the firmware as well as any other data buffers. The Alteon Acenic is also equipped with 2 DMA engines and supports a list of extra features such as the ability to perform onboard IP/TCP/UDP checksum calculations which is useful for alleviating such a load from the host processor.

One of the reasons why the Alteon Acenic is popular is due to the fact that Alteon Websystems have provided complete documentation for their NIC API and the Tigon 2 firmware is available as open source. This allows us to optimize the firmware for non-standard protocols in search for optimal performance results. Unfortunately in the early 2000 the Alteon Acenic was discontinued and Alteon was taken over by 3Com who replaced the Tigon 2 by an improved design; the Tigon 3. Unlike Alteon, 3Com decided not to provide documentation for the new NIC which makes it very difficult to reprogram the card.

[1] http://www.intel.com/

[2] http://www.microsoft.com/

[3] http://www.compaq.com/

[4] Also known as the Tigon 2

## 3 Our Proposal

Since the Alteon Acenic is no longer in production it was very difficult to acquire a pair of these cards for experimenting with customizable firmware, however we did find the Netgear GA620-T which is a similar model that makes use of the same Tigon 2 chipset. One of the major differences between the two is that the GA620-T has only 512KB of onboard memory compared to the 1MB of the Alteon Acenic. Using the documentation provided by Alteon for their Tigon 2 we will attempt to use the Netgear programmable NICs to develop an architecture for high-throughput low-latency message passing between processes running over a Gigabit Ethernet cluster.

Our primary aim is to eliminate any OS interaction during communication by building a user-level network similar to the VI architecture that gives user-level applications direct access to the NIC. This would involve creating a pinned down area in host memory that would be shared between the NIC and the host processes. Our system will use polling and events in replacement of interrupts in order to keep host CPU utilization to a minimum. We will use the Netgear's DMA facilities together with a zero-copy technique in order to reduce latency for small messages.

We will also modify the firmware to experiment with different size MTU's both for DMA and Ethernet frames as well as enable large PCI transfers in order to increase the overall throughput for applications requiring high bandwidth. We plan to design a lightweight but reliable user-level protocol for fast messaging that takes into consideration the reliability of the underlying physical network in order to provide the highest bandwidth possible with the lowest CPU utilization. This would be aided by offloading part of the checksum and protocol processing onto the NIC as far as the card's onboard CPUs allow us.

The final goal would be that of providing an API that can be seamlessly integrated with a user-level thread scheduler such as SMASH [6] developed by Debattista. This would allow users to develop fine-grain multi-threaded applications whose threads can be efficiently scheduled over SMP nodes connected on a gigabit cluster. Our designs will be implemented over the Linux platform.

## 4 Conclusion

We have highlighted various bottlenecks that arise when using conventional networking protocols for high-performance computing as well as illustrated several issues involved in developing high-performance communication protocols. We have analysed several attempts by previous projects aimed towards increasing throughput and decreasing latency for message passing over different gigabit network architectures. Finally we have stated our aim to use Netgear GA620-T programmable NICs together with the Linux OS in order to provide an API that allows efficient messaging between multi-threaded user-level applications running in a distributed environment over a cluster of SMPs.

## References

1. Infiniband Trade Association. Infiniband Architecture. http://www.infinibandta.org/ibta/.
2. Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic, and Wen-King Su. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*, 15(1):29–36, 1995.
3. Microsoft Compaq, Intel. *Virtual Interface Architecture Specification*, draft revision 1.0 edition, December 1997.

4. Joseph Cordina. High Performance TCP/IP for Multi-Threaded Servers. Master's thesis, University of Malta, 2002.
5. D. Culler, A. Arpaci-Dusseau, R. Arpaci-Dusseau, B. Chun, S. Lumetta, A. Mainwaring, R. Martin, C. Yoshikawa, and F. Wong. Parallel Computing on the Berkeley NOW. In *Ninth Joint Symposium on Parallel Processing*, 1997.
6. Kurt Debattista. High Performance Thread Scheduling on Shared Momory Multiprocessors. Master's thesis, University of Malta, 2001.
7. Emulex. clan. http://www.emulex.com/ts/legacy/clan/index.htm.
8. Patricia Gilfeather and Todd Underwood. Fragmentation and High Performance IP. University of New Mexico.
9. PCI Special Interest Group. *PCI Local Bus Specification*, revision 2.1 edition, June 1995.
10. Davib B. Gustavson. The Scalable Coherent Interface and Related Standards Projects. *IEEE Micro*, 12(1):10–22, 1992.
11. Alteon Networks Inc. *Tigon/PCI Ethernet Controller*, revision 1.04 edition, August 1997.
12. Alteon Networks Inc. *Gigabit Ethernet PCI Network Interface Card, Host/Nic Software Interface Definition*, revision 12.4.13 edition, July 1999.
13. Myricom Inc. Myrinet GM – the low-level message-passing system for Myrinet networks.
14. Scott Pakin, Mario Lauria, and Andrew Chien. High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet. 1995.
15. Fabrizio Petrini, Wu chun Feng, Adolfy Hoisie, Salvador Coll, and Eitan Frachtenberg. Quadrics Network (QsNet): High-Performance Clustering Technology. In *Hot Interconnects 9*, Stanford University, Palo Alto, CA, August 2001.
16. Piyush Shivam, Pete Wyckoff, and Dhabaleswar Panda. EMP: Zero-copy OS-bypass NIC-driven Gigabit Ethernet Message Passing. 2001.
17. H. Tezuka, F. O'Carroll, A. Hori, and Y. Ishikawa. Pin-down Cache: A Virtual Memory Management Technique for Zero-copy Communication. pages 308–315.
18. T. von Eicken, A. Basu, V. Buch, and W. Vogels. U-Net: a user-level network interface for parallel and distributed computing. In *Proceedings of the fifteenth ACM symposium on Operating systems principles*, pages 40–53. ACM Press, 1995.
19. Thorsten von Eicken, David E. Culler, Seth Copen Goldstein, and Klaus Erik Schauser. Active Messages: A Mechanism for Integrated Communication and Computation. In *19th International Symposium on Computer Architecture*, pages 256–266, Gold Coast, Australia, 1992.
20. Wallace Wadge. Achieving Gigabit Performance on Programmable Ethernet Network Interface Cards. University of Malta, 2001.

# Semantic Web-Services or Semantic-Web Services?

Matthew Montebello

Department of Computer Science and AI,
University of Malta

**Abstract.** The emergence of the Semantic Web together with the promise of Web Services, a new level of service on top of the current web is envisaged as the new silver bullet and considered as the next great cure-all for IT's ill. However, in order to employ their full potential, appropriate description techniques for such services need to be developed, and even though the future looks bright, promising and interesting, one should proceed with caution as potential pitfalls lie ahead. In this paper a brief overview of how to best ensue such issues is given, together with a personal clarification to which this paper is entitled.

## 1 Introduction

Academic and industrial bodies have been considering the issue of Web Services as being the next step forward in the area of eBusiness / eCommerce over the World Wide Web (WWW). A number of efforts have been made and are evolving, through the World Wide Web Consortium (W3C), to define standards, specifications and architectures for the spreading of this new breed of web applications. Tim Berners-Lee, Director of the W3C, referred to the future of the current WWW as the "Semantic Web" — an extended Web of machine-readable information and automated services that extend far beyond current capabilities. The explicit representation of the semantics underlying data, programs, pages, and other Web resources, will enable a knowledge-based Web that provides a qualitatively new level of service. Automated services will improve in their capacity to assist humans in achieving their goals by "understanding" more of the content on the Web, and thus providing more accurate filtering, categorization, and searches of information sources. This process will ultimately lead to an extremely knowledgeable system that features various specialized reasoning services. These services will support users in nearly all aspects of daily life — making access to information as pervasive, and necessary, as access to electricity is today. In this short paper we will highlight the main issues concerning a current debate around this research area, namely the uncertainty of differentiating between Web Services that are optimized using current semantic research enabled over the web (referred to as Semantic Web-Services) and Semantic Web enabled Web Services (referred to as Semantic-Web Services).

## 2 Where is the Dilemma?

Web Services basically involve three main issues, namely, the description language of such services, WSDL (Web Services Description Language); a protocol for communicating with the services, SOAP (Simple Object Access Protocol); and the UDDI, which is a registry where services are published. On the other hand Semantic Web offers an easier way to publish data that can be accessed and re-purposed as needed, thereby creating an ideal environment for higher level services over the web to become a reality. What is surely required, is a markup language that is descriptive enough that a computer can automatically determine its meaning. The following is a list of tasks such a language would be required to perform:

- Discovery: A program must first be able to automatically find, or discover, an appropriate Web service. Neither WSDL nor the UDDI allows for software to determine what a Web service offers to the client. A Semantic Web-Service describes its properties and capabilities so that software can automatically determine its purpose, while on the other hand a Semantic-Web service would be automatically understood in an already semantically-enabled web.
- Invocation: Software must be able automatically to determine how to invoke or execute the service. For example, if executing the service is a multi-step procedure, the software needs to know how to interact with the service to complete the necessary sequence. A Semantic Web-Service provides a descriptive list of what an agent needs to do to be able to execute and fulfill the service whereby the inputs and outputs of the service are discribed differing from a Semantic-Web Service.
- Composition: Software must be able to select and combine a number of Web services to complete a certain objective. The services have to interoperate with each other seamlessly so that the combined results are a valid solution.
- Monitoring: Agent software needs to be able to verify and monitor the service properties while in operation.

When Web markup languages evolve to the point that they can perform the above tasks, Web Services can begin to prosper. Semantic Web-Services can be considered to be all those Web Services that have and will continue to be developed before the Semantic Web itself becomes a reality. Once this second generation web is in place, then we can start developing real Semantic-Web Services.

## 3 Conclusion

There are many ways in which the two areas of Web Services and the Semantic Web could interact to lead to the further development of higher value-added Web Services. Berners-Lee has suggested that both of these technologies would benefit from integration that would combine the Semantic Web's meaningful content with Web Services' business logic.

Areas such as UDDI and WSDL are ideally suited to be implemented using Semantic Web technology. In addition, SOAP could use Resource Description Framework (RDF) payloads, remote RDF queries and updates, and interact with Semantic Web business rules engines, thereby laying the foundation for Semantic Web Services.

The W3C is engaged in building the Pyramid of Web Markup Languages, which starts with HTML and XML and continues upward to include RDF and the most recent Web Ontology Language (OWL). The off-spring of OWL is OWL for Services (OWL-S).

However, the technology issues of the Next Generation Web create many problematic questions like logic loops, syntax errors and trust-worthy published information, that must be solved before the full power and capability of the Semantic-Web Services are available.

## References

1. Abela C., and Montebello M., *DAML enabled Web Services and Agents in the Semantic Web,* in proceedings of WS–RSD'02, Erfurt Germany, October 2002.
2. Abela C., and Montebello M., *DAML enabled Agents and Workflow Components Integration,* in proceedings of the IADIS International Conference WWW/Internet 2002 , Lisbon Portugal, November 2002.

3. Abela C., and Solanki M., *The Landscape of Markup Languages for Web Service Composition,* May 2003.
4. Ankolenkar, A., Burstein, M., Hobbs, J.R., Lassila, O., Martin, D.L., McDermott, D., McIlraith, S.A., Narayanan, S., Paolucci, M., Payne, T.R. and Sycara, K., *DAML-S: Web Service Description for the Semantic Web* in Proceedings of The First International Semantic Web Conference, 2002.
5. Berners-Lee, T., Hendler, J. and Lassila, O. *The Semantic Web,* Scientific American, May 2001.
6. Cadoli, M., Palopoli, L. and Lenzerini, M., *Datalog and Description Logics: Expressive Power* in Proceedings of 6th International Workshop on Database Programming Language, 1997.
7. Fikes, R. and McGuinness, D. *An Axiomatic Semantics for RDF, RDF-S, and DAML+OIL* (March 2001), 2001. `http://www.w3.org/TR/daml+oil-axioms`.
8. Grixti W., Abela C., and Montebello M., *Name Finding from Free Text using HMMS,* submitted at the IADIS International Conference WWW/Internet,Madrid Spain, October 2004.
9. OWL `http://www.w3c.org/2004/OWL/`
10. RDF `http://www.w3c.org/RDF/`
11. Scicluna J., Abela C., and Montebello M., *Visual Modelling of OWL-S Services,* submitted at the IADIS International Conference WWW/Internet, Madrid Spain, October 2004.
12. SOAP `http://www.w3.org/TR/soap/`
13. UDDI `http://www.uddi.org/`
14. W3C `http://www.w3c.org`
15. WSDL `http://www.w3.org/TR/wsdl/`

# Automatic Document Clustering Using Topic Analysis

Robert Muscat

Department of Computer Science and AI,
University of Malta

**Abstract.** Web users are demanding more out of current search engines. This can be noticed by the behaviour of users when interacting with search engines [12, 28]. Besides traditional query/results interactions, other tools are springing up on the web. An example of such tools includes web document clustering systems. The idea is for the user to interact with the system by navigating through an organised hierarchy of topics. Document clustering is ideal for unspecified search goals or for the exploration of a topic by the inexpert [21]. Document clustering is there to transform the current interactions of searching through a large amount of links into an efficient interaction where the interaction is navigation through hierarchies. This report will give an overview of the major work in this area, we will also propose our current work, progress and pitfalls which are being tackled.

## 1 Introduction

We propose document clustering as an alternative to traditional web searching where the user submits a query to a search engine and is provided a ranked list of documents from which he has to choose. The user traverses the list of links (usually aided by a description which is usually a document snippet for each link).

With document clustering a query is optional, and initially the user is presented with an overall structure of the topics related to the domain. Selecting a particular topic from those provided will reveal further topical subdivisions which intuitively reside beneath the selected topic e.g. selecting *science* will reveal *mathematics, physics, computing, chemistry* together with other documents which fit under *science* but which do not fit under any other subcategory in *science*.

A number of terms are used to describe document clustering including text categorisation or document classification. Even though the results of each process are in the same spirit described above, the approach to solving the problem is different. Categorisation/classification methodologies use a machine learning (statistical approach) where a learning algorithm is applied on a manually built data set to allow the algorithm to find patterns within the data which allow classification. We consider manually built data sets an issue due to the fact that typically indexers are not able to agree on the majority of their indexing decisions [28, 30]. Typical text/document clustering on the other hand uses text analysis and models to decide how documents are grouped. Both these methods have their own pros and cons which we will illustrate.

CATEGORISATION

**Pros**
⇒ The structure of the topics within documents is known before hand.
⇒ Straight forward to implement after the documents have been appropriately vectorised. A number of accepted vectorising methods is already available.
⇒ Typically, machine learning systems make common assumptions across all data set, which is not always correct.
**Cons**
⇒ Requires a training period which may be lengthy at times.
⇒ Selection of training data is crucial.
⇒ Adding new topics to the collection requires reconfiguration of the training data and retraining.

CLUSTERING

**Pros**
⇒ The structure is generally detected automatically, without (or with minimal) human input.
⇒ Does not require any previous knowledge, the approach extracts all required information from the domain.
**Cons**
⇒ The structure is usually less intuitive and semantically nodes are less related than when manually built.
⇒ Requires a lot of processing.
⇒ Assumption are not made, but everything is seen from the model's perspective.

Throughout this report we will assume some knowledge of basic models used in information retrieval. We follow with a small introduction to clustering methodologies (domain independent discussion). The rest of the report describes work directly related to document clustering/classification, followed by evaluation methodologies and proposed techniques.

### 1.1 Data Clustering

The main types of clustering algorithms are *agglomerative* and *divisive*. As the name suggests, the former assumes that a domain of elements are initially separate clusters, during the clustering process elements are combined into clusters until finally all elements are in one all inclusive cluster. Divisive on the other hand is a top down approach, starting with one large cluster and ending up with each element in a separate clusters.

The principle agglomerative algorithms are *single link* [27], *complete link*[6] and *average link*[31]. On the assumption that initially all elements are clusters, each algorithm compares all clusters together to decide which will be joined (to form a single cluster) by comparing each element of the cluster. The two clusters with the minimum distance are then joined. The distance varies depending on the algorithm used:

**Single-Link** The distance between two clusters is the *minimum* distance between any two elements from the two clusters.
**Complete-Link** The distance between two clusters is the *maximum* distance between any two elements from the two clusters.

**Average-Link** The distance between two clusters is the *average* distance between all elements from the two clusters.

The above algorithms, by nature, are a traversal across dichotomies. To reach a particular number of clusters it is required to traverse all stages before reaching a target number of clusters. In most applications this might be an overhead and alternatives to this approach have been suggested.

One such alternative is *k-means* [22]. The advantages are that the algorithm allows $k$ clusters to be generated immediately ($k$ is the function parameter). Initially $k$ random elements are selected and assumed to be centroids. Centroids are points in the cluster which are the closest to all other elements in the cluster. All others elements are assigned to the nearest centroids and a new centroid is recalculated. The process is reiterated until no further elements move from one cluster to another. Other variations on this algorithm exist [14, 17, 19].

## 2 Related Work

In this section we review some of the major approaches document clustering relates to our approach. Some common evaluation methodologies will also be discussed.

### 2.1 Document Clustering

*Scatter/Gather* [10] was one of the first attempts to group document into clusters. The interaction was quite simple. Initially the user is presented with a set of groups which have been determined by the *Fractionation* algorithm. The user selects groups from the list presented, these groups are then joined and reclustered using another algorithm *Buckshot* (which is faster but less accurate). By iterating this process and selecting different clusters each time the user is shown different views of the domain [4, 10, 3]. Both of the two algorithms used are an adapted version of *k-means*. Further work on scatter/gather has improved the speed at which the user navigates through hierarchies (by using the *Cluster Refinement Hypothesis* which essentially states that most of the elements in two similar clusters will end up in the same cluster higher up in the hierarchy).

Another important approach was that taken Zamir and Etzioni [32, 33]. Their algorithm *Suffix Tree Clustering* lies between an information retrieval system (search engine) and the user, and linearly analyses the results of the query to provide a structure to the stream of data. The algorithm divides the clustering process into three stages, *preprocessing*, *base cluster identification* and the *combine base clusters* stage before it actually provides the user with a list of clusters. The main advantage of the algorithm is that it starts generating data as soon as the stream is being processed, so this means that the user can be provided with the data as soon as he makes a request. As the name implies, the *preprocessing* stage processes the stream for analysis by removing tags, punctuation and transforming terms into stems. The second phase, *base cluster identification* builds a suffix tree out of the stems from the first phase. Each of these nodes is assigned a score which increases with the number of documents which hit the nodes and decreases linearly with the length of the phrase in the node. The final phase, *combine base clusters* analyses the nodes to join similar ones with edges and spare the user from seeing duplicate data. Finally, the set of available graphs are scored and the top ten are returned to the user. Grouper [33] was an online application of the *suffix tree algorithm* and used results from *HuskySearch* [33] a meta-search engine which was under development by the same team.

Other work by Lerman [19] and Schütze and Silverstein [25], takes a more standard approach to clustering but concentrate, on the preprocessing of the document space. They reduce the dimensions

of documents to make clustering faster without deteriorating quality. Schütze tests two approaches, document truncation (taking the top 20 and 50 terms) and Latent Semantic Indexing (LSI is a method to project documents in a smaller dimension space while retaining the maximum information possible). Their results show only a difference in time efficiency which was not expected, since LSI (Latent Semantic Indexing [7, 5], discussed later) has generally improved performance of information retrieval system. Lerman [19] shows that LSI does in fact improve cluster quality. Her work shows that for this to happen, the dimension size selected must be optimal (or near optimal). [19] suggests a method to select an ideal dimension by selecting the point in the vector space where the largest weight gap occurs. Her work shows improvement on cluster quality (rather than time efficiency) with LSI using a simple single link algorithm.

### 2.2 Evaluation

Evaluating document clustering is still a grey area with no accepted technique available, especially for cases where the hierarchy is unknown and has to be generated.

The most common evaluation applied is using standard information retrieval values like *recall* (percentage of relevant retrieved documents over total retrieved) and *precision* (percentage of relevant retrieved over total retrieved). This is ideal for techniques where the hierarchical structure is known beforehand, and we can compare the entries for each cluster from the test against the correct data. To be able to calculate recall and precision clusters must be labelled a priori.

The issue is more complex when the overall hierarchy is not available. A number of attempts have been made to define measures which objectively allow clustering algorithms to be compared but there still is no accepted standard. Refer to [20] for a review. In most cases a function is defined which determines the quality of a cluster but this is not objective and is only an independent measure which is not gauged against any other reference except itself.

A number of datasets have emerged out of which a few are growing to become accepted standards, but still these do not take in consideration algorithms which generate the hierarchy themselves. Currently the `Reuters-21578`[1] data is the most widely used. Work is underway to add another Reuters data set which is the `Reuters Corpus`[2] which is a larger version and officially released by Reuters. The `Reuters Corpus` is being promoted to replace the `Reuters 21578` data set.

The closest method which is relevant to our requirements is that used by Mandhani et al. in [23]. The authors combine two methodologies to evaluate their work, the first considers each cluster as a single entity and a measure is used to analyse the quality of its content (the two suggested are *entropy* and *purity*). Secondly, they analyse the resulting tree, hierarchically at each level, by looking at the number of nodes per level, the purity at each level and by comparing the generated node labels at each level. We think that this kind of hybrid analysis is the best approach which can be applied to automatic document clustering. However, this approach generates a large number of results (separate values per level). An obvious enhancement would integrate all these separate results in fewer (ideally one).

## 3 Proposed Approach

In this section we propose our approach to document clustering. A number of techniques borrowed from other research areas in information retrieval to propose a method with which we can automatically generate clusters and topic hierarchies.

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/

[2] http://about.reuters.com/researchandstandards/corpus/

### 3.1 Clustering Methodology

Our document clustering solution is based on the assumption that a document can refer to more than one topic/concept. We will use these concepts to build the hierarchy and assign each document to the part of the tree which relates mostly to it.

The first challenge involves subdividing each document into topics. A number of topic extraction algorithms exist which help detect shifts in the topic of discussion in the text stream. A number of techniques are already available with the two main ones being *TextTiling* [8, 9, 11] and *C99* [2].

*TextTiling* determines topic shifts by first assuming that the document is divided up into blocks. Usually these blocks are taken to be paragraphs or blocks of $n$ terms. `TFxIDF` (Term Frequency and Inverse Document Frequency) is calculated across the blocks (assumed to be separate documents) and the terms in each block. After each term is assigned a weight, pairs of sequential blocks of $k$ terms are taken and compared using a distance algorithm. This will generate a sequence of distances for all pairs. This sequence is then smoothed using two techniques, one based on an average similarity value and another using simple median algorithm.

These sections will be analysed and keywords will be extracted from each section. The terms to select will require the analysis of the term weightings which have been assigned globally. We need to investigate other term weightings schemes to see which weightings best help determine the topic terms. Currently the suggested weightings are `TFxIDF`, document frequency and term frequency. We will also suggest local term and block frequency but these latter have not been tested yet. This phase is important and from preliminary tests we noticed that it can determine the quality of the overall system, since the quality of the tree is determined by the terms selected.

When a number of terms are selected they are used to build a concept hierarchy using specialised algorithms. These algorithms analyse the relations between terms and determine which term, semantically, lies beneath another. The idea is that of detecting relations of sub-topics, sub-categories and co-similarity e.g. "physics" and "nuclear physics", "science" and "physics", and "chemistry" and "physics". Given a set of such relations, we then build a full fledged topic hierarchy which is what the user will be navigating through at the very end. Documents are assigned to the topics which are deemed most similar to the topic in the hierarchy.

A number of algorithms exist which detect topic relationship. Besides Lawrie *et al.* which bases his approach on the *Dominating Set Problem* [18] we will describe the work by Sanderson and Croft in [24] where they describe the subsumption algorithm. The algorithm analyses the co-occurrence of a particular term $A$ with term $B$. To find a relation we assume that $A$ subsumes $B$ if $P(A|B) = 1$ and $P(B|A) < 1$.

By applying these ideas we claim that we will be able to automatically build a document hierarchy, with clusters containing documents which discuss similar topics. With this approach, we will not be able to compare with statistical/machine learning techniques but at least we generate an acceptable tree which can be traversed by a user.

All our comparisons will be performed using Latent Semantic Indexing (LSI) which is a technique used to map a term-document index into a different (usually smaller to make comparisons faster) space. LSI maps the index into a smaller dimension whilst retaining the information in the original index [7, 5].

The final step would be to assign a naming to each cluster by looking at its contents. We intend to extract phrases which are common inside each document in the cluster and use that as a cluster identifier.

### 3.2 Evaluation

As we have discussed in Section 2.2 evaluation is an issue which has been tackled a number of times but not yet settled, since the standards available are only useful for statistical/machine learning approaches. We suggest a method for evaluating automatic document clustering where the hierarchical structure is not known a priori.

We propose three test bases to be able to evaluate the quality of document hierarchies,

**Tree Distance** we will be using a tree distance algorithm in order to calculate the edit distance between the tree from the data set and the generated hierarchy. The edit distance is a measure which is based on a weight scheme assigned to each operation (insertions, deletions and edits) which transforms tree $A$ into tree $B$. A number of algorithms exist which carry out this task. Zhang and Shasha's algorithm returns the minimum edit distance between two trees [34].
Two tests will use the tree edit distance.
- An overall tree edit distance which will measure the two trees as a whole.
- An iterative test where starting from the leaves cluster nodes are replaced iteratively with the cluster content moving up in the hierarchy. For each iteration the tree distance is calculated.

More work related to tree edit distances is found in [1, 15, 16, 29, 26, 13].

**Cluster Quality Measures** This stage will measure cluster cohesiveness and quality. These measures are internal measures and they only base their calculations on the contents of the cluster without comparing it to any external source of information (like the original data set).

**Recall and Precision** are the standard measure used in information retrieval. They will be used to test each cluster. An overall average can be used to measure the overall system *recall* and *precision*. Refer to Section 2.2 for details.

Note that since our clusters will not be labelled, we will be matching clusters between source and generated clusters by taking the two with the shortest distance between them.

## 4 Results

At the moment no results are available. Zhang and Shasha's algorithm resulted in extreme computation times especially since the algorithm requires a regular expression to find patterns in strings representing the trees. We are at the moment trying to find a way to speed up this process. We are also considering a tree edit distance algorithm which does not guarantee a *minimum* edit distance but does allow us to compare tree structures in tractable times.

This hurdle stopped the development process of our ideas until we have fixed our evaluation algorithms. At the moment we are concentrating our work on the evaluation process to get it up and running.

## 5 Conclusion

We do not expect our results to compare with statistical/machine learning approaches since the hierarchical structure is manually constructed beforehand. Our work should, however be able to automatically build structures which are navigable by the average user. We intend to show this by doing a user study to determine navigability.

The most critical stage in the process is the term selection to automatically identify topic terms. Part of the research will be dedicated to the fine-tuning of the current models to enable better term selection. This model is also critical in identifying cluster titles.

At this stage work is in progress to make the evaluation process as fast, accurate and easy as possible to allow future reference to our work and allow use of the same evaluation models we suggest in our system.

## References


1. Philip Bille. Tree edit distance, alignment distance and inclusion. Technical report, IT University of Copenhagen, March 2003.
2. Freddy Y. Y Choi. Advances in domain independent linear text segmentation. In *Proceedings of NAACL-00*, 2000.
3. Douglass R. Cutting, David Karger, and Jan Pedersen. Constant interaction-time scatter/gather browsing of very large document collections. In *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 126–135, 1993.
4. Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM Press, 1992.
5. Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
6. D. Defays. An efficient algorithm for a complete link method. *The Computer Journal*, (20):346–366, 1977.
7. Susan T. Dumais, George W. Furnas, Thomas K. Landauer, Scott Deerwester, and Richard Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the Conference on Human Factors in Computing Systems CHI'88*, 1988.
8. Marti A. Hearst. Texttiling: A quantitative approach to discourse segmentation. Technical Report S2K-93-24.
9. Marti A. Hearst. Multi-paragraph segmentation of expository text. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 9 – 16, New Mexico State University, Las Cruces, New Mexico, 1994.
10. Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 76–84, Zürich, CH, 1996.
11. Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 59–68. ACM Press, 1993.
12. Bernard J. Jansen, Amanda Spink, Judy Bateman, and Tefko Saracevic. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998.
13. Taeho Jo. Evaluation function of document clustering based on term entropy. In *The Proceedings of 2nd International Symposium on Advanced Intelligent System*, pages 302–306, 2001.
14. J.H. Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, (58):236–244, 1963.
15. Philip Klein, Srikanta Tirthapura, Daniel Sharvit, and Ben Kimia. A tree-edit-distance algorithm for comparing simple, closed shapes. In *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 696–704. Society for Industrial and Applied Mathematics, 2000.
16. Philip N. Klein. Computing the edit-distance between unrooted ordered trees. In *Proceedings of the 6th Annual European Symposium on Algorithms*, pages 91–102. Springer-Verlag, 1998.
17. Jouko Lampinen. On clustering properties of hierarchical self-organizing maps. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks, 2*, volume II, pages 1219–1222, Amsterdam, Netherlands, 1992. North-Holland.

18. Dawn Lawrie, W. Bruce Croft, and Arnold Rosenberg. Finding topic words for hierarchical summarization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 349–357. ACM Press, 2001.
19. Kristina Lerman. Document clustering in reduced dimension vector space. http://www.isi.edu/ lerman/papers/Lerman99.pdf (last visited 09/02/2004), January 1999.
20. David D. Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 246–254. ACM Press, 1995.
21. Xin Liu, Yihong Gong, Wei Xu, and Shenghuo Zhu. Document clustering with cluster refinement and model selection capabilities. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–198. ACM Press, 2002.
22. J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Symp. Math. Statist. and Probability, 5th, Berkeley*, volume 1, pages 281–297, Berkeley, CA, 1967. AD 66981. 157., 1967. Univ. of California Press.
23. Bhushan Mandhani, Sachindra Joshi, and Krishna Kummamuru. A matrix density based algorithm to hierarchically co-cluster documents and words. In *Proceedings of the twelfth international conference on World Wide Web*, pages 511–518. ACM Press, 2003.
24. Mark Sanderson and Bruce Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–213. ACM Press, 1999.
25. H. Schutze and H. Silverstein. Projections for efficient document clustering, 1997.
26. Stanley M. Selkow. The tree to tree editing problem. *Information Processing Letters*, 6(6):184–186, 1977.
27. R. Sibson. Slink: an optimally efficient algorithm for a complete link method. *The Computer Journal*, (16):30–34, 1973.
28. Associate Professor Amanda H. Spink, S. Ozmutlu, and H. C. Ozmutlu. A day in the life of web searching: An exploratory study. *Information Processing and Management 40.2*, pages 319–45, 2004.
29. Kuo-Chung Tai. The tree-to-tree correction problem. *J. ACM*, 26(3):422–433, 1979.
30. D. Tarr and Harold Borko. Factors influencing inter-indexing consistency. In *Proceedings of the American Society for Information Science (ASIS) 37th Annual Meeting*, volume 11, pages 50–55, 1974.
31. E. M. Voorhees. Implementing agglomerative hierarchic clustering algorithms for use in document retrieval. *Information Processing & Management*, 22(6):465–476, 1986.
32. Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.
33. Oren Zamir and Oren Etzioni. Grouper: a dynamic clustering interface to Web search results. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1361–1374, 1999.
34. K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.

# Monadic Compositional Parsing with Context Using Maltese as a Case Study

Gordon J. Pace

Department of Computer Science and AI,
University of Malta

**Abstract.** Combinator-based parsing using functional programming provides an elegant, and compositional approach to parser design and development. By its very nature, sensitivity to context usually fails to be properly addressed in these approaches. We identify two techniques to deal with context compositionally, particularly suited for natural language parsing. As case studies, we present parsers for Maltese definite nouns and conjugated verbs of the first form.

## 1 Introduction

Combinator-based programming, where a basic set of objects are defined, together with operators to combine them to produce more complex instances has frequently been advocated. This approach essentially embeds a domain-specific language, which allows the used to use a combination of the embedded language and the host language to design a solution. The high-level of abstraction offered by modern functional programming languages such as Haskell and ML make them a perfect host language for such an approach. Various complex domains tackled in this manner can be be found in the literature (see, for example, hardware [2], stage lighting [12] and financial contracts [6]).

One domain in which the combinator approach has been successfully applied is that of parsing. In [15], Wadler presented a number of parser combinators for Haskell [9], using which one can compose complex parsers from simpler ones. Furthermore, the core-combinator code for parsing is itself very simple and easy to follow, making it possible to change and add new basic-parsing combinators as other parsing requirements not originally needed. Various later work (see, for example, [3–5, 7, 13, 14]) is based on this classic paper.

Ljunglöf [8] proposed the use of lazy functional programming languages (in particular, Haskell) to parse natural language texts. The paper uses the parser combinators from [15] and [5] to construct a Montague-style parser from English into logic statements. The parser is expressed very concisely in Haskell in easy-to-follow code, which makes the claim of the paper rather convincing. However, a number of more complex, yet interesting issues are not discussed. One such issue is that of sentence context-awareness, where certain words in a sentence depend on the context in which they appear in the sentence. A typical example is verb conjugation which must match its subject. Although these issues did not arise in the parser presented in [8], they arise immediately when the same approach is applied to a more comprehensive grammar, or to other languages such as Maltese.

Not much work has been done on the automatic parsing of Maltese. Of particular note, is the Maltilex project [11], which aimed at producing a computationally tractable lexicon of the Maltese language. Rosner [10] also treats the problem of parsing the Maltese article in conjuction with prepositions, using finite state automata, which is closely related to the domain of application we adopt for this paper.

In this paper, we give an overview monadic parser combinators (from [5]), and proceed to build the machinery to express grammars which are context dependent. We then propose solutions for problems arising with context-dependent grammars, in particular the case of natural language parsing. We use the parsing of simple Maltese grammatical constructs to illustrate our approach. At this stage, our primary aim has been the extension of parser combinators to deal with context. It remains to be established how useful our approach is on more substantial natural language grammar subsets.

## 2 Haskell Parser Combinators

Haskell is a strongly typed language, and types can be used, not just to identify certain problems at compile-time, but also to shape and direct a solution. It thus makes sense to start by asking what the type of a parser is to be. A parser in a function which reads an input stream of tokens of a particular type. After consuming part of the input, we would expect it to return back the object parsed, and the unconsumed part of the input stream. Actually, since the parser may match the input in more than one way, it would return a list of such possibilities:

```
newtype Parser a b = Parser ([a] -> [(b,[a])])
```

One may think that in some cases this extremely general parser type is overkill — producing the complete list of matches, when we may sometimes be only interested in a possible solution. However, lazyness comes to the rescue in this case. If we are only interested in any parse of the input, we just take the first element of the output list of the parser, and lazyness will take care of the rest.

Since the tag `Parser` stops us from applying the parser directly, we will define a function to 'peel' this tag away:

```
parse (Parser x) = x
```

We can now proceed to construct a number of simple parsers. The simplest parser is the one which returns a constant value, but leaves the input stream intact:

```
constantParser :: a -> Parser b a
constantParser x = Parser (\is -> [(x,is)])
```

Another simple parser is one which simply fails:

```
failParser :: Parser a b
failParser = Parser (\_ -> [])
```

Using these, we can define a parser which fails when a given boolean condition is false:

```
check :: Bool -> Parser a ()
check True  = constantParser ()
check False = failParser
```

Another useful parser, is one which checks that the input stream has been fully consumed:

```
fin :: Parser a ()
fin = Parser (\is -> case is of { [] -> [((),[])]; _ -> [] })
```

Parsing a single object at the beginning of the input:

```
one :: Parser a a
one = Parser (\is -> case is of { [] -> []; (x:xs) -> [(x,xs)] })
```

Parsers can be easily packaged as monads, which offer a reusable structure for functional programs.

```
instance Monad (Parser a) where
  -- return :: b -> Parser a b
  return = constantParser
  -- (>>=) :: Parser a b -> (b -> Parser a c) -> Parser a c
  parser1 >>= fparser2 =
    Parser $ \is ->
      let results1 = parse parser1 is
      in  concat [ parse (fparser x) is' | (x,is') <- results ]
```

The first offers a means of creating a parser which returns a given value, while the second allows us to run two parsers sequentially.

This now allows us to combine parsers to obtain compound parsers such as:

```
p1 <*> p2 =
  p1 >>= (\x1 -> p2 >>= \x2 -> return (x1, x2))
```

Haskell provides two alternative ways to make monad code look prettier:

```
p1 <*> p2 =
  do
    x1 <- p1
    x2 <- p2
    return (x1,x2)
```

```
p1 <*> p2 =
   do { x1 <- p1; x2 <- p2; return (p1,p2) }
```

Another useful parser is one which reads a symbol off the input stream, succeeding only if it satisfies a given property:

```
parseSat :: (a -> Bool) -> Parser a a
parseSat property = do { x <- one; check (property x); return x }
```

Parsing a particular character or string off the input stream can now be defined in terms of this combinator:

```
parseChar c  = parseSat (c==)

parseString "" = return ""
parseString (c:cs) = do { parseChar c; parseString cs; return (c:cs) }
```

Similarly, one can define parsing a vowel or consonant:

```
isVowel     c = c `elem` ['a','e','i','o','u']
isConsonant c = isAlpha c && not (isVowel c)

parseVowel     = parseSat isVowel
parseConsonant = parseSat isConsonant
```

Another useful combinator is the `ifnot` operator which tries one parser, and if it fails uses another:

```
p1 `ifnot` p2 = Parser $ \is ->
    case (parse p1 is) of
      [] -> parse p2 is
      rs -> rs
```

This last operator resolves potential non-determinism between the parsers by giving priority to the first operand. In fact, although our parser type can handle non-determinism, we have only, as yet, used them in deterministic ways. Non-deterministic choice between parsers can be easily defined in this framework:

```
p <+> q = Parser (\is -> parse p is ++ parse q is)
```

Parsing any word from a list of possibilities can now be expressed recursively as:

```
parseAnyOf []     = mzero
parseAnyOf (p:ps) = p <+> anyOneOf ps
```

Parsing a character or a word from a list of possibilities can now be expressed as:

```
anyCharOf   = parseAnyOf . map parseChar
anyStringOf = parseAnyOf . map parseString
```

One can now also define regular expression-like operators on parsers such as star (to accept any number of repetitions of a parser):

```
star :: Parser a b -> Parser a [b]
star parser =
 do { x <- parser; xs <- star parser; return (x:xs) } <+> return []
```

Parsing word space and space separated text has never been easier:

```
parseWordSpace = plus (parseSat isSpace)
  where
    plus p = p <*> star p

p1 <**> p2 = do { x1 <- p1; parseWordSpace; x2 <- p2; return (x1,x2) }
```

## 3 Parsing in Context: The Maltese Definite Article

We will now look at the Maltese article to see different approaches to parsing a natural language using the parser combinators presented in the previous section.

In English, parsing a definite noun using these combinators is trivial, since the article and noun are independent of each other.

```
parseArticle = parseString "the"
parseNoun = anyStringOf ["dog", "cat", "mouse", "computer"]
parseDefiniteNoun = parseArticle <**> parseNoun
```

The Maltese article is, however, more complicated.

### 3.1 The Maltese Article

The rules governing the Maltese article are quite intricate, combining morphological rules to aid pronunciation of the words with grammatical rules.

Indefinite nouns in Maltese have no extra signifier. The basic form of the definite article is constructed by adding *il–* before the noun. Hence, *kelb* (a dog), becomes *il–kelb* (the dog). Unfortunately, the matter is not so simple. Nouns starting with one of a number of letters (the *xemxin*[1] consonants: ċ, d, n, r, s, t, x, ż) have the initial letter of the noun assimilated into the article, replacing the *l* to accomodate pronunciation-based rules. *Serp* (a snake) becomes *is–serp* (the snake). Furthermore, nouns starting with a vowel sound (possibly preceded by a silent letter) drop the initial *i* in the article. *Orfni* (an orphan) becomes *l–orfni* (the orphan). Finally, words starting with two or three consonants, the first being an *s* or an *x*, take *l–*, but also precede the noun by an initial *i*. For example, *spazju* (a space) becomes *l–ispazju* (the space).

To make matters worse, if the word preceding the article ends in a vowel, the article loses the initial *i* (should it start with one). For example, *kiel il–kelb* (he ate the dog), but *qela l–kelb* (he fried the dog)[2]

As these intricate rules indicate, parsing definite noun in Maltese is not as straightforward as it is in English. The logical flow of consequence flows from the right (the noun) to the left (the article). In a limited sense, this is also the case in English when parsing an indefinite noun, and finding *a* or *an*. However, whereas finding *an* in English, one can look for a noun starting with a vowel, in Maltese the situation is more complex since without matching against a lexicon, it is impossible to deduce whether *l–iskola* (the school) is the definite form of *skola* or *iskola*.

Furthermore, Maltese definite nouns reveal another important issue. Usually, we would like to have compositional parsers, that is, if I write a parser for the article, and a parser for nouns, I should be able to simply combine the two to parse a noun with an article. This is what was done in the case of the English definite noun. In the case of Maltese, this is not possible, since one would have to add constraints ensuring that the article and noun match. A simpler situation would arise when parsing the indefinite article (*a* or *an*) in English. We propose a simple enrichment of the Haskell parser combinators enabling us to define such constructs compositionally, by refering to the context of a parsed object.

### 3.2 Compositional Context Parsing

A parser with look-ahead can differentiate between equivalent input by looking at input still to arrive. We propose an approach in which look ahead can be expressed compositionally in terms of normal parser combinators.

#### 3.2.1 Looking into the future

With the current `Parser` structure, we can already handle looking ahead into the future input context. We can generalise the look-ahead parser into one which fails if the former failed, but succeeds returning no information (via the unit type `()`) and leaves the input untouched:

```
lookAhead :: Parser a b -> Parser a ()
lookAhead p = Parser $ \is ->
    case (parse p is) of
      [] -> return []
      _  -> return [((), is)]
```

[1] Literally *sun letters*. Singular *xemxija*.

[2] An alternative grammatical view is that the article consists of *l–*, which acquires an initial *i* if it is not preceded by a vowel, and the noun it is attached to starts with a consonant.

This can be used to write a compositional parser for the indefinite English article, or the Maltese definite article:

```
articleEnglish = parseString "an" <*> lookAhead startsWithVowel
  where
    startsWithVowel = parseWordSpace <*> parseVowel

articleMalteseXemxin = do
                         parseChar 'i'
                         c <- parseSat isXemxija
                         parseChar '-'
                         lookAhead (parseChar c)
                         return ('i':c:"-")
```

Note that the Maltese article parser works for nouns starting with one of the *xemxin* consonants ċ, d, n, r, s, t, x and ż. It can be extended to deal with the other cases in a similar manner.

Using this technique, we can describe future context in terms of the same combinators, hence resulting in a compositional parser. Parsing a definite Maltese noun can be expressed in terms of two separate parsers as: `articleMaltese <**> nounMaltese`, and the matching of the article with the noun is transparently taken care of.

### 3.2.2 Looking into the past

Although the technique described in the previous section enables us to describe all the different cases of the Maltese definite noun, one nuance still has not been catered for — the initial *i* of the article is dropped if the preceding word ends in a vowel. Ideally, we would like to have a past context parser transformer to be able to specify the condition as:

```
weak_i = lookBack (parseVowel <*> parseWordSpace) `ifnot` parseChar 'i'
```

All occurrences of matching the initial *i* in Maltese definite article would then be replaced by the above parser, solving the problem. However, the parser type we have been using, does not hold any past information, and needs to be enriched to enable us to refer to past context:

```
type Stream a = ([a],[a])
newtype Parser a b = Parser (Stream a -> [(b,Stream a)])
```

We have basically just changed our notion of a stream from a sequence of unconsumed symbols to a pair of sequences of symbols — enabling us to recall not only the unconsumed symbols, but also the consumed ones. This does not change the use of any of the composed parsers. However, it may involve slight changes in the foundational parser combinators. For example, parser `one` would need to be redefined as follows[3]:

```
one :: Parser a a
one = Parser $ \(ws,is) ->
         case is of {[] -> []; (i':is') -> [(i',(ws++[i'],is'))] }
```

We can now define `lookBack` using this additional information:

```
lookBack :: Parser a b -> Parser a ()
lookBack p = Parser $ \(ws,is) ->
```

[3] Interestingly, the definitions of the monad operators $\gg=$ and `return` remain unchanged, despite the fact that the variables now refer to different objects!

```
case (parse (star one <*> p <*> fin) ([],ws)) of
  [] -> return []
  _  -> return [((), (ws,is))]
```

Note that, since we would like our parser to match the final part of the consumed input stream, we compose it after `star one` which consumes the first part of the stream. Clearly, this raises major efficiency issues.

However, our definition of `star` ensures that the order of the possible parses is longest first. Since most natural language look-backs would only look at a short part of the stream, and we are only interested whether there is at least one match, lazyness ensures that the match is quickly found, and the search stopped. Part of the overhead of using `star one` (the past stream is recomposed through the repeated calls to `one` by `star`) can be avoided by splitting the input and passing it directly to the parser, for some performance improvement.

Another solution to improve efficiency can be used if the parser applied to the consumed input is sufficiently simple to be able to express its reverse of the parser (which matches the same strings but reversed). Is such cases, we can simply apply the reversed parser to the reversed past, with no other parsers composition. If the programmer would be required to provide the reversed parser, the solution would lead to unreadable and confusing code. One alternative would be to enrich the parser library to be able to identify a subset of parsers which can be automatically reversed, and apply the reverse of the parser before 'looking back'.

## 4 Further Context in Parsing: Maltese Verbs

The approach presented in the previous section allows us to express parsers with conditions on the context of the input stream where they are applied. The approach works well in the case of very local context, and is thus ideal to apply when one in word morphology based on phonological rules which depend on the immediate context. However, the approach is not so well suited for certain applications, where the context may be set by other complex rules based on text residing far from the location being parsed or information gathered as a result the parsing itself. Furthermore, any context parsing means duplicating the parsing effort since the information may be parsed multiple times, as a context and as the main text.

Another approach to enable adapting to the context, is to modify the global state in order to pass information which will be used later. The unstructured use of such an approach lead to non-reusable code. However, Haskell being a pure functional language with no side-effects, means that we have to use a monad (or equivalent) to pass the state around. This enables us to tailor the use of the state to a limited repertoire of operators, which ensure that the functionality is available without the possibility of abuse.

### 4.1 Maltese Verbs of the First Form in the Present Tense

The equivalent of the infinitive of Maltese verbs is the past third person singular conjugated form of the verb. Hence, the equivalent of *to ride* would be *rikeb* (he rode). Maltese verbs of semitic origin fall into one of ten forms (types). In the first and main form, one finds verbs based on three consonants (*trilitteri*) as in the case of *rikeb*. The other forms are derived from this first form. In this paper we will only be dealing with regular verbs of the first form.

The conjugation of regular verbs of the first form is rather straightforward, and follows the following pattern:

| | Example (rikeb) | General case[4] ($c_1v_1c_2v_2c_3$) |
|---|---|---|
| First person singular | nirkeb | $n\ v_1\ c_1\ c_2\ v_2\ c_3$ |
| Second person singular | tirkeb | $t\ v_1\ c_1\ c_2\ v_2\ c_3$ |
| Third person singular male | jirkeb | $j\ v_1\ c_1\ c_2\ v_2\ c_3$ |
| Third person singular female | tirkeb | $t\ v_1\ c_1\ c_2\ v_2\ c_3$ |
| First person plural | nirkbu | $n\ v_1\ c_1\ c_2\ c_3\ u$ |
| Second person plural | tirkbu | $t\ v_1\ c_1\ c_2\ c_3\ u$ |
| Third person plural | jirkbu | $j\ v_1\ c_1\ c_2\ c_3\ u$ |

We make two remarks on the table above (i) the infinitive of the verb can be reconstructed from all the singular conjugations, but we lack the second vowel in the plural forms; and (ii) encountering *tirkeb* leaves an ambiguity whether we are referring to the second person singular, or the female form of the third person singular.

Based on these rules, one can easily check a given verb infinitive against a list of known verbs:

```
verbKnown  verb = verb `elem` ["rikeb", "kiteb" ...]
```

### 4.2 Attributes

One recurring non-local dependency in natural language parsing, is that of matching gender, tense or case between different parts of the same sentence. To enable such matching, all we need is a means of setting attributes with particular values. An attribute may be set multiple times, but we would have to ensure that the value is always the same. In this manner, both parsing the noun and the verb may set the gender attribute of the subject of the sentence. However, unless the gender set by the two matches, the parsing of the phrase will fail. In the case of attributes which may take one of a number of possibilities, we use non-determinism to try all the cases. For example, when encountering the Maltese verb *tikteb*, we can deduce that the subject is either second person singular, or female third person singular. By having a non-deterministic parse, yielding either of the two, we ensure that any context deciding the gender or person of the subject to one of the possibilities will not fail, but simply reduce the non-determinism.

The other operations we would like to perform on attributes are reading and renaming. Reading is necessary to enable us to act according to a value set elsewhere. Renaming of an attribute may be necessary to be able to adapt the attributes as we acquire more information. For example, parsing a noun would set the attributes of the noun, while a simple sentence would set the attributes of the subject, verb and object. Therefore, once we discover that a particular noun phrase is the subject phrase, we would be able to rename the `NounGender` attribute to `SubjectGender`. We merge the state monad into the parser monad to handle attributes:

```
type State = [(AttributeName, AttributeValue)]
newtype Parser a b = (State, Stream a) -> [(b, (State, Stream a))]
```

As in the previous case, we would need to redefine some of the basic parser combinators, but the derived combinators would still work unmodified. Once again, the code defining the operators in the classes `Monad` remains unchanged.

We also define a number of functions which update and read the attribute list:

[4] The letters $c_i$ indicate the consonants of the verb, while $v_i$ indicate the vowels. Other symbols represent actual letters occurring in the verb.

```
setAttribute :: (AttributeName, AttributeValue) -> Parser a ()
setAttribute attribute =
  Parser $ \(memory, is) ->
    if noClash attribute memory
      then [((), (updateState attribute memory, is))]
      else []

getAttribute :: AttributeName -> Parser a String
getAttribute attribute_name =
  Parser $ \(memory, is) ->
    if isDefined attribute_name memory
       then [(lookup attribute_name memory, (memory, is))]
       else []
```

Using similar definitions, we also define `setAttributes` (which updates a list of attributes), `renameAttribute` (which renames an attribute) and `renameAttributes`.

Before we define a parser for Maltese verbs, we must decide which attributes we will be setting. From the verb, we can only derive information about the subject. There are three pieces of information that can be gathered: the grammatical person, gender and number. We thus set three attributes, `SubjectPerson`, `SubjectGender` and `SubjectNumber` (respectively).

Parsing the first letter of the verb indicates the person. To simplify the presentation, we split the cases into two — singular and plural. The following in the code for the singular case. The plural case is similar.

```
parsePersonSingular =
  do setAttribute ("SubjectNumber", "Singular")
     c <- anyCharOf ['n','t','j']
     case c of
        'n' -> setAttribute   ("SubjectPerson" "First"
        'j' -> setAttributes [("SubjectPerson","Third"), ("SubjectGender","Male")]
        't' -> setAttribute   ("SubjectPerson","Second") <+>
               setAttributes [("SubjectPerson","Third"), ("SubjectGender","Female")])
```

Using these functions, we can now express how to parse a conjugated verb. As in the previous case, we separate parsing verbs conjugated in the singular and the plural form. The following is the singular form:

```
parseVerbSingular =
   do  parsePersonSingular
       v1 <- parseVowel
       c1 <- parseConsonant
       c2 <- parseConsonant
       v2 <- parseVowel
       c3 <- parseConsonant
       check (verbKnown [c1,v1,c2,v2,c3])
```

Parsing a verb is now simply the non-deterministic choice between the parsing a singular or plural verb:

```
parseVerb = parseVerbSingular <+> parseVerbPlural
```

Now that we have the parsing of a simple verb in place, and a parser for nouns (setting attributes `Gender`, `Person`, `Number`), we can construct a simple sentence parser for transitive verbs as follows:

```
parseSubject =
  do n <- parseNoun
     renameAttributes [("Noun"++x, "Subject"++x) | x <- ["Gender", "Person", "Number"]]
parseObject =
  do n <- parseNoun
     renameAttributes [("Noun"++x, "Object"++x) | x <- ["Gender", "Person", "Number"]]
parseSentence = parseSubject <**> parseVerb <**> parseObject
```

Although in the example given here we use strings both for the tag names and values stored within, it may be clearer to use a structured type allowing us to group together certain attributes (in this case, all three attributes refer to the subject — one would also need identical attributes for the object of the phrase). We use strings only for ease of presentation.

## 5 Conclusions

We have presented two approaches which can be used to introduce context-sensitivity for parsing natural language compositionally using parser combinators. The first allows us to refer to the context using parsers, while the second is an instantiation of the state monad with operators refined for natural language parsing.

The combinator-based approach to programming is extremely versatile and introduces a level of abstraction which enables program re-interpretation. Although in this paper we store each parser a function which actually performs the parsing (its behaviour), by changing the basic combinators we can store its structure (for example, the fact that the parser for the Maltese definite noun is built as the 'sequential' composition of two other parsers). No information is lost, since we can reconstruct the parsing function from the description. Having access to the structure, however, would enable us to use analyse and perform operations of the grammar which would have otherwise been impossible to perform[5]. The following are three avenues we would like to explore using this approach:

**Reversing a parser:** We have already discussed the efficiency issue when looking back at the consumed input stream. One possible improvement we mentioned in passing is by reversing a parser and applying it to the reversed past stream. Clearly, however, not all parsers can be reversed. With access to the structure, we can define a partial parser reverse function. The `lookBack` function can then use the reversed parser when possible, resulting in more efficient simple look-backs.

**Optimisation:** Other optimisations also become possible when we have access to the structure. An obvious example is analysing the given grammar structure, and construct a minimised automaton if it turns out to be a regular language.

**Language generation:** If we have access to the structure of the grammar, we can also use it to generate, or complete phrases, as opposed to parsing. We have already constructed an experimental phrase generator based on parsing combinators. Combining the two together would expose further the duality of grammars as means of checking sentence correctness and as tools to structure and generate sentences.

[5] This distinction between structure and behaviour is very important in other domains such as hardware description. If a combinator library for hardware description is only used to simulate the circuit, then just having a circuit described as a function from inputs to outputs will be sufficient. If, however, we would like to count the number of gates in the circuit, or measure the longest combinational path, clearly we require access to the underlying structure.

The work presented in this paper is mainly concerned with dealing compositionally with context, with particular emphasis on natural language processing. We feel that the examples presented in this paper are encouraging and plan to apply the techniques on a non-trivial portion of Maltese grammar to assess their real-world effectiveness.

## References


1. L–Akkademja tal-Malti, *Tagħrif fuq il–Kitba tal–Malti II*, Klabb Kotba Maltin, 2004.
2. K. Claessen, M. Sheeran and S. Singh, *The Design and Verification of a Sorter Core,* in Correct Hardware Design and Verification Methods 2001, Lecture Notes in Computer Science 2144, 2001.
3. J. van Eijck, *Parser Combinators for Extraction*, 14th Amsterdam Colloquium, 2003.
4. G. Hutton and E. Meijer, *Monadic Parser Combinators*, J. of Functional Programming, 2(3):323–343, 1992.
5. G. Hutton, *Higher-Order Functions for Parsing*, J. of Functional Programming, 8(4):437–444, 1998.
6. S. Peyton Jones and J.-M. Eber and J. Seward, *Composing contracts: an adventure in financial engineering*, ACM SIG-PLAN Notices, 35(9):280–292, 2000.
7. Peter Ljunglöf, *Pure Functional Parsing*, Licentiate thesis, Technical Report no. 6L, Department of Computer Science and Engineering, Chalmers University of Technology, 2002.
8. Peter Ljunglöf, *Functional Programming and NLP*, Department of Computer Science and Engineering, Chalmers University of Technology, 2002.
9. S. Peyton Jones, J. Hughes et al., *Report on the Programming Language Haskell 98, a Non-strict, Purely Functional Language,*, Available from `http://www.haskell.org`, 1999.
10. M. Rosner, *Finite State Analysis of Prepositional Phrases in Maltese.* In G. Pace and J. Cordina, editors, University of Malta Computer Science Annual Workshop 2003 (CSAW '03), 2003.
11. M. Rosner, J. Caruana, and R. Fabri. *Maltilex: A Computational Lexicon for Maltese.* In M. Rosner, editor, Computational Approaches to Semitic Languages: Proceedings of the Workshop held at COLING-ACL98, Université de Montréal, Canada, pages 97–105, 1998.
12. M. Sperber, *Developing a stage lighting system from scratch,* in Proceedings of the sixth ACM SIGPLAN international conference on Functional programming, 2001.
13. D. Swierstra and L. Duponcheel, *Deterministic, error-correcting combinator parsers.* In Advanced Functional Programming, volume 1129 of LNCS, 1996.
14. S. Doaitse Swiestra, *Combinator Parsers: From Toys to Tools.* In Graham Hutton, editor, Haskell Workshop, pages 46–57, 2000.
15. P. Wadler, *How to Replace a Failure by a List of Successes.* In Jean-Pierre Jouannaud, editor, FPCA, volume 201, pages 113–128, 1985.

# Matrix Decomposition Algorithms for Feature Extraction

Derrick Pisani

Department of Computer Science and AI,
University of Malta

**Abstract.** Clinical decision support software is a delicate system which, can potentially be the physician's closest friend. The aim of such systems is to be able to cleverly recommend a list of treatment options which closely matches the patient. We envisage a system which learns from experts without ever needing to ask them for feedback, and thus one which learns from past patient encounters. The system needs to be adaptive as well as dynamic, since all patients are different even if they may exhibit very similar symptoms. This paper proposes using matrices to capture such data, and algorithms using Singular Value Decomposition to predict treatments.

## 1 Introduction

The aim of the project is that of achieving the level of machine intelligence required for delivering clinical decision support[6]. The traditional meaning of learning is the gaining of knowledge or information about an area, and thus the acquiring of an understanding of the topic[7]. An understanding is the product of the learning process, and it is very often an interpretation of the knowledge gained. As a result, in complex areas such as those found in the medical field, learning does not lead to all experts, the physicians, to have a common understanding. An analysis of the actions of several physicians, however, reveals trends or patterns of prescriptions to tests and medications given to patients. Although this is not equivalent to an understanding of the area, which is satisfactory to substitute the expert, such knowledge can be used as training data for clinical decision support systems. This paper introduces techniques which use singular value decomposition to extract this training data dynamically.

## 2 Basic Definitions

Observations of a working system or natural phenomenon may be described by a set of simultaneous equations. These, in turn, may be represented in matrix notation.

In the case of a system with $m$ simultaneous equations in $n$ variables, we may write

$$\mathbf{A}x = b \tag{1}$$

where $x$ is an unknown solution column vector and $\mathbf{A}$ is an $(m \times n)$ co-efficient matrix. Thus:

$$x = \mathbf{A^{-1}}b$$

For a matrix $\mathbf{A}$, where $b = 0$, the system is said to be *homogenous*. The solution here is apparently $x = 0$, but when the $det\mathbf{A} = 0$, the inverse $\mathbf{A}^{-1}$ does not exist, and thus the solution for $x$ is non-trivial. This is also always the case when $m > n$. Matrix $\mathbf{A}$ is then said to be *singular*.

The *rank* of a matrix is the maximum number of linearly independent rows or columns. When the $det\mathbf{A} = 0$, the rows of $\mathbf{A}$ are linearly dependent and thus $rank(\mathbf{A}) < n$. This is termed *rank-deficiency*, and leads to the system potentially having infinitely many solutions. Solving a system where the coefficient matrix is rank-deficient is equivalent to solving $m$ simultaneous equations in $n$ unknowns where $m < n$, and thus traditionally considered intractable.

Let us finally consider the same system of simultaneous equations described by equation (1) when:

$$b = \lambda x$$

$\lambda$ is a scalar and its values are called *eigenvalues*, whilst the corresponding solutions are called *eigenvectors*. Using the above substitution for $b$ in $\mathbf{A}x = b$:

$$\begin{aligned} \mathbf{A}x &= \lambda x \\ (\mathbf{A} - \lambda I)x &= 0 \end{aligned}$$

Therefore, the *characteristic equation* for systems with non-trivial solutions is:

$$det(\mathbf{A} - \lambda I) = 0 \tag{2}$$

Its solution gives the eigenvalues of $\mathbf{A}$ and, in turn, the solutions of $x$.

## 3 Singular Value Decomposition

Most systems of simultaneous equations may be solved using algebraic manipulations involving elementary row operations. In those cases where Gaussian Elimination or Triangular (LU) Decomposition do not succeed to give satisfactory results, we can use *Singular Value Decomposition (SVD)* to diagnose and possibly solve the problem[8].

SVD techniques deal with singular matrices, or ones which are very close to being singular. They are an extension of eigen decomposition to suit non-square matrices. Any matrix may be decomposed into a set of characteristic eigenvector pairs called the component factors, and their associated eigenvalues called the singular values [3].

The SVD equation for an $(m \times n)$ singular matrix $\mathbf{A}$ is:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathbf{T}} \tag{3}$$

where $\mathbf{U}$ is an $(m \times m)$ orthogonal matrix, $\mathbf{V}$ is an $(n \times n)$ orthogonal matrix and $\mathbf{\Sigma}$ is an $(m \times n)$ diagonal matrix containing the singular values of $\mathbf{A}$ arranged in descreasing order of magnitude.

A vector in an orthogonal matrix, which can be expressed as a linear combination of the other vectors. The vectors in this space are thus, also mutually independent, and thus a solution for $x$ may be now calculated.

The equation for Singular Value Decomposition needs to be further explained by showing the significance of the eigenvectors and the singular values.

Each of the orthogonal matrices is achieved by multiplying the original $(m \times n)$ data matrix $\mathbf{A}$ by its transpose, once as $\mathbf{A}\mathbf{A}^{\mathbf{T}}$ to get $\mathbf{U}$, and once as $\mathbf{A}^{\mathbf{T}}\mathbf{A}$ to get $\mathbf{V}^{\mathbf{T}}$. The data is then available in square

matrices, on which eigen decomposition may be applied. This involves the repeated multiplication of a matrix by itself, forcing its strongest feature to predominate. In SVD, this is done on both the orthogonal matrices. The pairs of eigenvectors are the rows in $\mathbf{U}$ and the columns in $\mathbf{V}$. These are sorted in order of their strength; the respective singular value acts as the required numerical indicator as further explained below. [3]

Let us first explain the relationship between the singular values and the co-efficient matrix's eigenvalues:

$$\begin{aligned}\mathbf{A^T A} &= (\mathbf{U\Sigma V^T})^T(\mathbf{U\Sigma V^T}) \\ \mathbf{A^T A} &= \mathbf{V\Sigma^T U^T U\Sigma V^T} \\ \mathbf{A^T A} &= \mathbf{V\Sigma^T \Sigma V^T} \\ \mathbf{A^T A} &= \mathbf{V\Sigma^2 V^T}\end{aligned}$$

$\mathbf{\Sigma^2}$ contains the eigenvalues of $\mathbf{A^T A}$. Similarly, it may be shown that they are also the eigenvalues of $\mathbf{AA^T}$. They may be arranged in decreasing order of magnitude:

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq ... \geq \lambda_n \geq 0$$

Singular values of a matrix $\mathbf{A}$ are defined as the square root of the corresponding eigenvalues of the matrix $\mathbf{A^T A}$:

$$\sigma_\mathbf{j} = \sqrt{\lambda_j}$$

As with the eigenvalues, they are sorted according to their magnitude, and the corresponding eigenvectors in the orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ follow the same order. The SVD equation (3) takes the form:

$$\mathbf{A} = \begin{pmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1m} \\ u_{21} & u_{22} & u_{23} & \dots & u_{2m} \\ u_{31} & u_{32} & u_{33} & \dots & u_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & u_{m3} & \dots & u_{mm} \end{pmatrix} \begin{pmatrix} \sigma_{11} & 0 & 0 & \dots & 0 \\ 0 & \sigma_{22} & 0 & \dots & 0 \\ 0 & 0 & \sigma_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_{nn} \end{pmatrix} \begin{pmatrix} v_{11} & v_{21} & v_{31} & \dots & u_{n1} \\ v_{12} & v_{22} & v_{32} & \dots & u_{n2} \\ v_{13} & v_{23} & v_{33} & \dots & u_{n3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{1n} & v_{2n} & v_{3n} & \dots & u_{nn} \end{pmatrix}$$

An alternative way to what was initially described, exists for defining the size of the matrices: $\mathbf{U}_{m\times n}$, $\mathbf{V}_{n\times n}$ and thus $\mathbf{\Sigma}_{n\times n}$ [1]. This is due to the fact that when $m > n$, then $n$ rows are sufficient to solve for $x$, and the main diagonal of $\Sigma$ still crosses the top left $(n \times n)$ subsection of the matrix.

Some singular values are equal or very close to zero. A singular value $\sigma_j$ describes the importance of $u_j$ and $v_j$, and when its value approaches zero it indicates that these associated vectors are less significant.

A proof that SVD can be used to decompose singular matrices may be found in Hourigan and McIndoo's work [1].

[1] Hourigan J S, McIndoo L V. The Singular Value Decomposition. http://online.redwoods.cc.ca.us/instruct/darnold/laproj/Fall98/JodLynn/report2.pdf

# 4 Applications in Clinical Decision Support

## 4.1 The Dataset

The dataset being utilised for this project originates from collections of anonymous patient-encounters from various acute care hospitals during the year 2001. The hospitals selected are all medium to large ones and geographically sparse over a very large country. As shown in Table 1, they have accommodated thousands of patients.

**Table 1.** Hospitals in Dataset

| Hospital | Size in no of beds | Patient Encounters |
|---|---|---|
| A | 200-399 | 80,000 |
| B | 200-399 | 55,000 |
| C | 400+ | 175,000 |
| D | 400+ | 90,000 |

For the scope of this study only inpatients who have spent at least one night in hospital are being considered. The sex distribution is about 4 males to 6 females, and there is a very even and broad distribution of ages.

Each patient is represented by a unique identifier that masks the person's identity, but real demographic and medical information is available. This includes sex, age, diagnosis in ICD-9CM format, procedure information in ICD-9CM and CPT-4 format, and hospital billing information. Since hospital-specific codes are used to describe the latter, when patient-encounters from more than one hospital are used, the data needs to be standardised in advance.

## 4.2 Use of SVD Techniques

Singular Value Decomposition is an excellent unsupervised learning tool to process source data in order to find clusters, reduce dimensionality, and find latent variables. This means that field or experiment data can be converted into a form which is free of noise or redundancy and that it can be better organised to reveal hidden information.

Although a lot of work on information retrieval (IR) has been done using SVD, most of it is in the areas of genetic patterns and semantic analysis of text and hypertext documents. Clinical decision support software affects human lives, and thus, since SVD techniques are still being tested with less delicate applications, no current attempts at using it to help the physician's decision making process are known to the author. The many current efforts to harness SVD techniques in the field of IR have been instrumental for this project, and good examples are provided by Wall's work [9–11] and others such as [2, 4].

The various prescriptions (orders) received by a hypothetical number of patients exhibiting a certain condition can be visualised in Fig. 1. The orders are however not mutually exclusive, and no information is given regarding which orders were prescribed together to which patients. SVD is not a straightforward statistical technique, and manages to mine such information.

In the following sections we will briefly describe the benefits of using Singular Value Decomposition to process the source data originating from patient encounters.

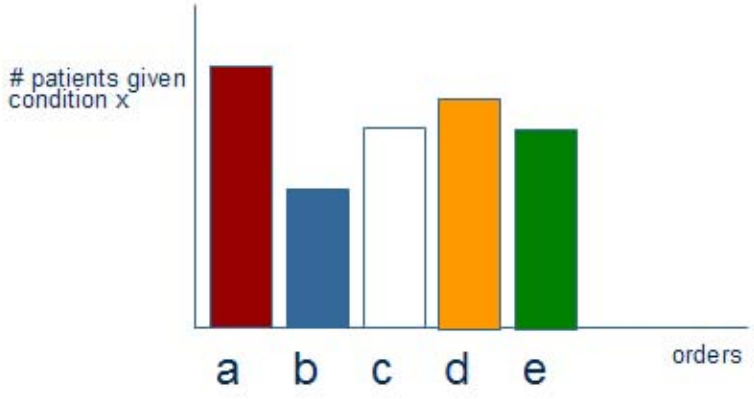



**Fig. 1.** Different Orders for a Particular Condition

**Redundancy Reduction:** The diagram in Fig. 2 shows the process of converting field data into matrix format. The chosen rows are those of patients who fit the demography and symptom profile chosen by the physician.

Since we are using field data, and not one specifically collected for this project, the matrix containing the patient-encounter data contains a degree of noise and redundancy. The sorted singular values in a Singular Value Decomposition help represent the original matrix in less rows, by pushing the less significant rows to the bottom of the orthogonal matrices. The matrix may thus be smoothed by eliminating those vectors whose respective singular value is equal to, or approaches zero. As explained in section 3, these are the vectors which least impact on the original matrix.

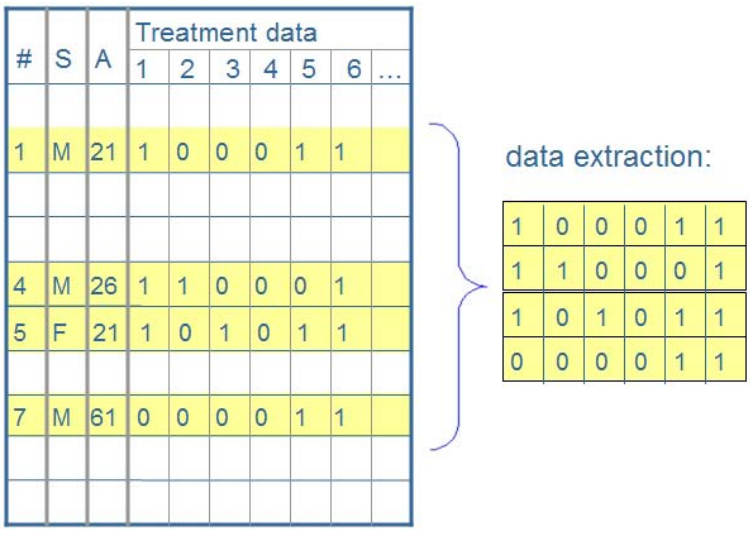



**Fig. 2.** Source Data converted to Matrix-format

**Exposing Components:** As already explained in Section 3, Singular Value Decomposition results in sorted eigenvectors (components). These are very useful in analysing patient encounters and revealing the most predominant components. These are representative of the procedures which most significantly describe the typical patient's treatment for the chosen patient profile.

We believe that the clusters formed when the source data is projected in the dimensions of the first two components is indicative of their constituents. An example is seen in Fig 3.

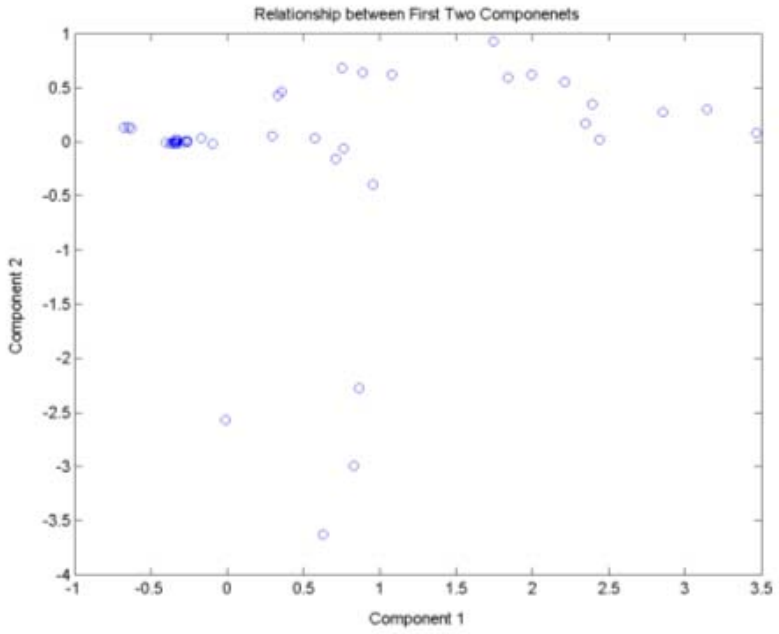



**Fig. 3.** Clustering when Source Data is Projected in two Dimensions

**Understanding Latent Variables:** Latent variables are hidden sources in the original data; they explain the common traits in patient encounters, but cannot be measured directly. For the observed data represented by matrix $\mathbf{A}$, in $n$ variables, the latent variables may be expressed in $l$ dimensions where $l << n$. If we assume that the initial $n$ variables capture the whole domain, the product of SVD will be a set of variables which includes the latent ones. What may initially be evident as a prescription to a set of tests and medications for particular symptoms may thus have been due to a combination of latent variables including demography, diagnosis, or the *'school of thought'* of the physician or expert. [5, 4, 2]

Following SVD, this hidden information will not become transparent, but it will be captured by the system as shown in Fig. 4. The 'Controls' act for the Latent variables and for every patient where the same conditions apply, the relevant ones are toggled on. In this sense, every 'Control' is a description of the effects of a set of Latent Variables, and certain medical procedures go ordered when a control is toggled on.

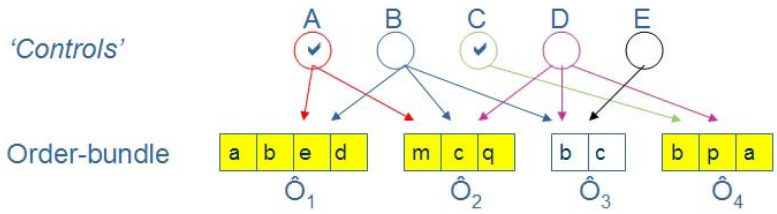



**Fig. 4.** Hidden sources may be represented as 'Controls'

## 5 Conclusion

SVD techniques will be used to predict a patient's treatment based on all the information known about the patient, without having to assign static weights to a particular characteristic. This means that the importance of age or a particular symptom, for instance, may be very significant in one encounter and disregarded in another as the system's decision support is based on a non-trivial dynamic distance measure between the current patient profile and past cases.

## References

1. Glover D M, Jenkins W J, Doney S C. Modeling, Data Analysis and Numerical Techniques for Geochemistry (Notes). Massachusets, USA: Woods Hole Oceonographic Institution; 1998,2000,2002
2. Gong Y, Liu X. Generic Text Summarization Using Relevance Measure and Latent Semantic Analysis. In: SIGIR'01; 09 December 2001. San Jose, USA: NEC USA, C & C Research Laboratories; 2001
3. Gorrell G. Word Co-occurrence and Singular Value Decomposition. Sweden: Institutionen for dataveteskap, Linkopings universitet. http://w3.msi.vxu.se/ nivre/teaching/gslt/mlpapers/gorrell.ps
4. Jiangsheng Y. Singular Value Decomposition With Applications to IR and Text Clustering. Beijing, China: School of Electronics Engineering and Computer Science, Peking University; 2003. p12-14
5. Landauer T K, Foltz P W, Laham D. An Introduction to Latent Sematic indexing. Discourse Processes, 25, p259-284; 1998
6. Pisani D. Achieving Intelligent Clinical Decision Support through Orderset Management. In: Pace G. and Cordina J., editors. CSAW 2003; 10 July 2003; Malta: Department of Computer science and AI, University of Malta; 2003. p129-132.
7. Porter N editor. Webster's Revised Unabridged Dictionary. 1998 edition (MICRA, Inc.). Springfield, Massachusetts, USA: C. & G. Merriam Co.; 1913.
8. Press W H, Flannery B P, Teukolsky S A, Vetterling W T. Numerical Recipes in C : The Art of Scientific Computing. 2nd ed. Cambridge: Cambridge University Press; 1992
9. Wall M E, Rechsteiner A, Rocha L M. Singular value decomposition and principal component analyis. In: Berrar D P, Dubitzky W, Granzow M., editors. A Practical approach to Microarray Data Analysis; Kluwer: Norwell Massachusetts, USA; 2003. p91-109
10. Wall M E, Dyck P A, Brettin T S. SVDMAN - Singular value decomposition analysis of microarray data. Los Alamos, USA: Bioscience Division, Los Alamos National Laboratory; 2001
11. Wall M E. Validation of Singular Value Decomposition analysis of Global expression Data. Los Alamos, USA: Bioscience Division, Los Alamos National Laboratory

# Corpus-Driven Bilingual Lexicon Extraction

Michael Rosner

Department of Computer Science and AI,
University of Malta

**Abstract.** This paper introduces some key aspects of machine translation in order to situate the role of the bilingual lexicon in transfer-based systems. It then discusses the data-driven approach to extracting bilingual knowledge automatically from bilingual texts, tracing the processes of alignment at different levels of granularity. The paper concludes with some suggestions for future work.

## 1 Machine Translation

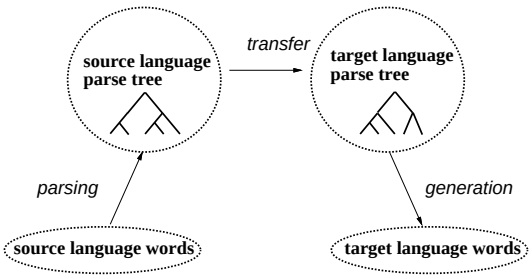



**Fig. 1.**

The Machine Translation (MT) problem is almost as old as Computer Science, the first efforts at solving it dating from the beginning of the 1950s. At that time, early enthusiasm with the novel and apparently limitless capabilities of digital computers led to grandiose claims that the MT problem would be cracked by purely technological approaches.

However, researchers not only failed to appreciate the computational complexity of the problem but also the role of non-technological factors: syntactic and semantic knowledge, and also subtle cultural conventions that play an important role in distinguishing a good translation from a meaningless one.

By ignoring these factors the systems developed by the early researchers were hopelessly inadequate in terms of both coverage and quality of translation. As a consequence, almost all of the funding for MT dried up in the mid-1960s with a recommendation that more investigation into fundamental linguistic issues and their computational properties was necessary if MT was to move ahead.

This certainly happened. For the next twenty years the emerging field of computational linguistics concentrated upon the development of grammar formalisms – special purpose notations created with the twin goals of (i) encoding linguistic knowledge and (ii) determining computations.

The emphasis on linguistic issues, and on the machinery required to handle them, left its impact upon the proposed architectures for MT systems.

Most Machine Translation (MT) systems are transfer based and figure 1 shows the architecture of a typical transfer-based MT system[1]. The oval items on the left and right represent source and target texts, whilst the arrowed lines denote the three different translation phases.

- **Parsing.** A representation (which need not necessarily be a parse tree, as indicated in the figure) of the surface text is computed, as represented by the left edge of the triangle.
- **Transfer.** The source level representation is transferred to a target representation.
- **Generation.** A surface text in the target langauge is generated from the target level representation.

The essential feature of transfer based systems is that the representation level is not abstract but *linguistic*: a representation from source language S still contains elements (e.g. words) from language S.

A great deal of the character of the system depends upon the *depth* of the representation chosen for transfer. A shallow representation (i.e. one that is close to the surface text), will be relatively easy to compute. From such a representation, it will be relatively easy to generate text. However, precisely because it is shallow, slight variations in the surface form will have to be dealt with by the transfer component. Conversely, a deeper representation, which abstracts away from much of the surface detail, will cause the transfer component to shrink.

Different levels of representation yield a spectrum of different solutions to the MT problem. At one extreme there is Example Based Machine Translation (EBMT) in which there is no representation apart from the text itself. In such systems, the transfer component is essentially lookup (with appropriate indexing) into a database containing pairs of sentences or segment fragments.

At the other extreme are systems like Rosetta (Landsbergen [6]), in which the representation is interlingual (i.e. language free encoding of meaning - see figure 1). Here, the there is no transfer component at all, but the analysis and generation phases are substantial.

### 1.1 Transfer

The majority of systems fall somewhere in between the two extremes. Systran, for example, which is the technology underlying Babelfish as used by Alta Vista and Google, uses a low level representation (see Wheeler [9]) that includes a limited amount of syntactic information.

The transfer component involves two somewhat different processes.

- Structural transfer deals with the syntactic transformations that are necessary to translate the syntactic conventions of the source language to that of the target. Figure 1.1 shows a simple example that reorders adjectives and nouns (cf. English and Maltese).
- Lexical transfer concerns the actual translations between words. The system component that deals with this kind of equivalence is the bilingual lexicon, as discussed in the next section.

[1] Figure due to Jurafsky and Martin [5]

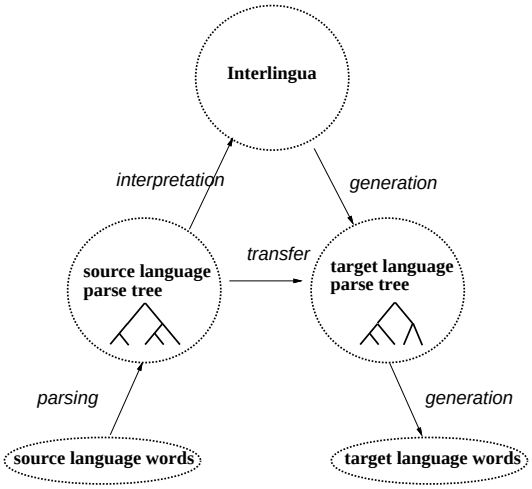



**Fig. 2.**

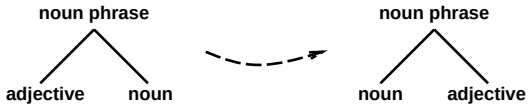



**Fig. 3.**

### 1.2 The Bilingual Lexicon

A key element of any translation system is a bilingual lexicon, which, as its name suggests, is essentially a lookup table which encodes mappings between words in the two languages under consideration. The construction of such a lexicon is both non-trivial and highly labour-intensive. The non-triviality arises from several sources. First is the inherently difficult nature of the equivalence relation between words in different languages: for a given word there may be zero to N different possible translations - and where N is greater than 1, the choice may depend upon complex aspects of the context. Second is that the number of words is large: a typical dictionary contains c. 100,000 entries. Even if we ignore the problem of multiple-word expressions and morphological complexity, the creation of such a lexicon is a prodigious task: if each entry takes 15 mins to create (probably an underestimate) this corresponds to 12.5 man-years of work.

In short, then, the creation of a bilingual dictionary constitutes a knowledge acquisition bottleneck, and computational linguists have turned to data-driven rather than handcrafted approaches to lexicon construction.

## 2 The Data Driven Approach

Professional translators know that the richest source of translation knowledge are texts or text fragments that have already been translated into another language. The question is whether this intuition can be harnessed by an automatic procedure for extracting translation knowledge in general, and a bilingual lexicon in particular, from particular instances of translated texts.

In order to develop this idea, we will regard a text S as a set of *segments*. Segments are simply linguistic entities of a certain type, where the types vary according to the level of analysis. Thus at a given level of analysis, a segment might be a sequence of words, a sequence of paragraphs, or an entire chapter. Furthermore, other relations might be defined over a set of such segments. Within a sentence text, for example, sequence-of-word segments will be ordered. However, for the present we will not be concerned with either the internal character of segments nor with the relations that might exist over them. The important intuition is that the text is a *set*.

## 3 Alignment

Within the framework under discussion, bilingual knowledge is derived from *correspondences* between a text S and its translation T, that is, between the respective segments in $S = \{s_1, s_2, \ldots, s_n\}$ and $T = \{t_1, t_2, \ldots, t_m\}$. These correspondences are expressed using the notion of *alignment*, defined by Isabelle and Simard [4] as a subset of the Cartesian product $S \times T$.

To give an example, suppose

$$S = \{s_1, s_2, s_3, s_4, s_5\}$$

and

$$T = \{t_1, t_2, t_3, t_4, t_5\}$$

A particular alignment might be

$$A = \{(s_1, t_1), (s_2, t_2), (s_2, t_3), (s_3, t_4), (s_4, t_5), (s_5, t_5)\}$$

This associates the segment $s_1$ with segment $t_1$; the segment $s_2$ with segments $t_2$ and $t_3$; the segment $s_3$ to segment $t_4$; and the segments $s_4$ and $s_5$ to the same segment $t_5$.

A bilingual text, or "bitext" is a triple (A,S,T) where A is an alignment, S is a text, and T is its translation.

Input to an alignment system is normally the bitext ({(s,t)},{s},{t}), i.e. comprising a single, large text segment, its translation , and the alignment that is the one-to-one mapping between the two.

In order to derive bilingual lexical information from this initial input, a bitext at the next level must be constructed by

- identifying the subsegments of S and T
- computing the best alignment of the subsegments of S and T.

This process is repeated until subsegmentation reaches the word level where, potentially every subset of words in the source text can be aligned with every subset of words in the target text.

At this point we inspect the best alignment and count occurrences of similar pairs. The highest ranking pairs are frequently occurring n-gram-to-n-gram translations and therefore good candidates for the bilingual lexicon.

Of course, this is a deliberate simplification of the process. When, at a given level, the number of subsegments is large, the cost of computing the possible alignments of all possible subsegments is prohibitively high so in practice certain heuristic assumptions must be incorporated to reduce this cost, e.g.

– Subsegments are defined in such a way that that the number of them at a given level is reasonable. This is achieved by staging the alignment process to deal, in order, with alignments of sections, paragraphs, sentences, and finally words.
– Limits are imposed on the set of candidate alignments that will actually be considered. When performing subsentence alignment, for example, one can put an upper bound on their length. Hence, most work in this area in fact deals with single words. We shall see other examples below.

### 3.1 Paragraph Alignment

Typically, segmentation of paragraphs is carried out in two stages: anchor alignment and paragraph alignment, as described originally by Brown et. al. [2]. A slightly simplified procedure was adopted for the Maltese/English work by Bajada [1].

The two texts are inspected and a set of *anchor point types* manually identified. An anchor point is a point that is easily recognised in both source and target texts such as a chapter heading, title, or other mark that is characteristic of the text type under investigation. The identification of anchor points divides the two texts into *sections* containing one or more paragraphs, as shown in figure 4. A table of anchor point types together with their respective translations is compiled and stored in memory (to be used for computing alignment costs).

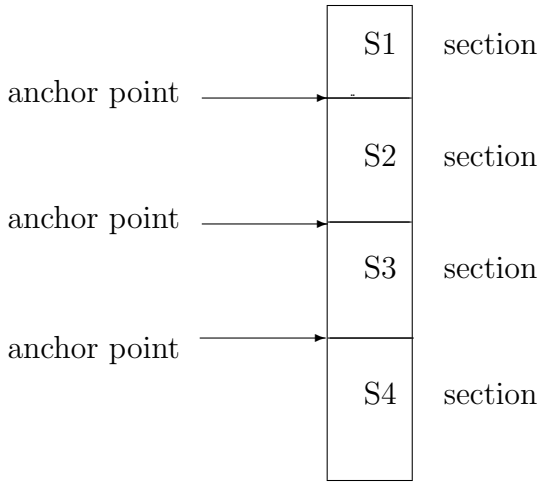



**Fig. 4.**

For the most part, corresponding anchor points are present in both source and target texts. However, there are typically omissions of one or other anchor and/or minor departures from the exact anchor translations as stored in the table. Hence, the automatic alignment of anchors is not entirely straightforward.

Both Brown et. al. and Bajada adopted the technique of assigning a cost to the alignment of a proposed source/target anchor pairing. If the proposed pair is in the table, the cost is zero. If one element of the pair is an omission, a fixed cost is assigned. Otherwise, rules can be supplied for dealing with concrete cases. In the case of Bajada, a dynamic programming algorithm to find the least cost alignment is based on a limited pattern of possibilities for the *next* anchor pair.

Specifically, two lists of anchor points are created, one for the source and one for the target. The ordering within these list preserves the ordering of anchor points in the texts. The algorithm iterates through each source anchor point for which the first N target anchor points are compared, where N is a small integer, and the one with the least cost assigned. This is an example of limiting the set of candidate alignments that will actually be considered, as mentioned above. In the work carried out by Bajada, a value of 3 was chosen for N, since it was established empirically that for the texts under consideration, the anchor points were never out of alignment by more than 3.

The quality of the results obtained depends largely on the reliability of the anchor points identified by hand as indicators of "section-ness". Although these can be very reliable in formalised text such as legal documents, the problem is harder in the case of, for example, government circulars or communications from the local Water Services. Simard et. al [8] have proposed using *linguistic cognates* as the basis for recognising potential anchor points automatically.

Anchor points provide a rough alignment of source and target sections each containing several unaligned paragraphs.

The next stage aligns the paragraphs within sections. For each aligned anchor pair, the algorithm first retrieves and then counts the number of source and target paragraphs they enclose. Let N and M be the respective counts.

Following the strategy of limiting the set of possible alignments, Bajada allows the following alignment options, for $n, m > 1$:

$$0:0, 1:1, 1:0, 0:1, 1:n, n:1, n:n, n:m(n<m), n:m(m>n), 1:0, 0:1$$

where X:Y means that X source paragraphs are aligned with Y target paragraphs. Clearly, the choice is fully determined for all but the last two cases, for which all possible combinations are enumerated and ranked by considering the length, in characters, of individual paragraphs or concatenations of paragraphs. This strategy proved to be tractable because the number of paragraphs under consideration was always small for the texts being considered.

The underlying assumption is that the length ratio of correctly aligned paragraphs is approximately constant for a given language pair. Hence the cost of a given alignment is proportional to the sum, over all source/target pairs in the alignment, of the difference in character length between each source/target pair, and the least-cost alignment is that which minimises the sum.

### 3.2 Sentence Alignment

Like paragraph alignment, the sentence alignment algorithms proposed by Brown et. al. [2], and Gale and Church [3], are based on the length of sentences in characters. The latter was reimplemented by Bajada [1]. Like paragraph alignment, both algorithms assume that longer sentences tend to be translated by longer sentences and vice versa. The operation of the algorithm is summarised by Bajada as follows:

> "The algorithm calculates the cost of aligning sentences by assigning a probabilistic score to each possible pair of sentences which make up an alignment. The score is assigned according to the ratio of the lengths of the two sentences and on the variance of this ratio for all sentences. Dynamic programming is then used to find the maximum likelihood alignment of the sentences."

### 3.3 Word Alignment

In a classic paper that outlines the basic tenets of statistical machine translation, Brown et. al. [7] develop a model in which for estimating Pr(f|e), the probability that f, a target (French) sentence, is the translation of e, a source (English) sentence. This is of course defined in terms of alignments expressed as sets of what they term *connections*.

A connection, written $\mathrm{con}_{j,i}^{f,e}$, states that position j in f is connected to position i in e. The model is directional, so that each position in f is connected to exactly one position in e, including the special "null" position 0 (where it is not connected to any word). Consequently, the number of connections in a given alignment is equal to the length of f.

The translation probability for a pair of sentences is expressed as

$$\Pr(f|e) = \Sigma_{a \in A} \Pr(f, a|e)$$

where A is the set of all possible alignments for the sentences f and e. In other words, each possible alignment contributes its share of probability additively. The question is, how to estimate $\Pr(f, a|e)$. Brown et. al. propose 5 different models.

The first model assumes that $\Pr(f, a|e)$ depends purely on the translation probabilities $t(\phi|\epsilon)$ between the *word* pairs $(\phi, \epsilon)$ contained in the connections of a, and is obtained by multiplying these probabilities together. The individual values for $t(\phi|\epsilon)$ are estimated using a training set comprising a bitext that has been aligned at sentence level.

The second model improves upon the first by taking into account the observation that, under translation, and for certain language pairs at least, words tend to retain their relative position in the sentence: words that are near the beginning of the source sentence tend to appear near the beginning of the target sentence.

### 3.4 Conclusion

Bajada's FYP report [1] implemented a complete framework for the extraction of bilingual lexical equivalences that included all of the other alignment phases described above, including the two versions of word alignment mentioned above. Using training and test material based on an abridged version of the Malta-EU accession treaty, results for section, paragraph and sentence alignment were encouraging, with precision and recall above 90% over the test material.

The results for word alignment were somewhat less satisfactory, with precision and recall figures closer to 50%. It was shown that the second model for word alignment described above was an improvement on the first.

Amongst the strategies under consideration for improving the performance of the system at word level are:

- Investigation of larger datasets in order to determine the rate of improvement for a given increase in size.
- Use of a lemmatiser to reduce the vocabulary size and hence the frequency of individual connections between source and target words (this will be of particular interest on the Maltese side).
- Investigation of possible equivalences between frequent n-grams of words for given n.

## References

1. J. Bajada. Investigation of translation equivalences from parallel texts. *University of Malta, BSc. Final Report*, 2004.
2. P. Brown, J. Lai, and R. Mercer. Aligning sentences in parallel corpora. In *29th Meeting of the Association for Computational Linguistics*, pages 169–176, 1991.
3. W. Gale and K. Church. A program for aligning sentences in bilingual corpora. In *29th Meeting of the Association for Computational Linguistics*, pages 177–184, 1991.
4. Pierre Isabelle and Michel Simard. Propositions pour la reprsentation et lvaluation des alignements et des textes parallles. *Rapport technique du CITI.*, 1996.
5. D. Jurafsky and J. Martin. *Speech and Language Processing*. Prentice-Hall, 2000.
6. J. Landsbergen. Isomorphic grammars and their use in the rosetta translation system. In M. King, editor, *Machine Translation Today*. Edinburgh University Press, 1997.
7. Brown P., J. Cocke, S. Della Pietra, V. Della Pietra, F. Jelinek, and R. Mercer. A statistical approach to language translation. *Computational Linguistics*, 16.2:79–85, 1990.
8. Michel Simard, George F. Foster, and Pierre Isabelle. Using cognates to align sentences in bilingual corpora. In *Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research*, pages 1071–1082. IBM Press, 1993.
9. P. Wheeler. Systran. In M. King, editor, *Machine Translation Today*, pages 192–208. Edinburgh University Press, 1997.

# Visual Modeling of OWL-S Services

James Scicluna, Charlie Abela and Matthew Montebello

Department of Computer Science and AI,
University of Malta

**Abstract.** The Semantic Web is slowly gathering interest and becoming a reality. More people are becoming aware of this and are trying to embed Semantic Web technologies into their applications. This involves the use of tools that can handle rapid ontology building and validation in an easy and transparent manner. In the area of Semantic Web Web Services (SWWS) an OWL-S specification defines a set of ontologies through which a semantic description of the service can be created. At times this is not an easy task and could result in an incorrect specification of the description or even lead the fainthearted user to resort to some other type of description language. This paper describes the OWL-S editor tool that provides two methodologies in which such a web services description can be developed without exposing the developer to the underlying OWL-S syntax. These methodologies are based on a mapping from WSDL to OWL-S and on modeling a composite service using standard UML Activity Diagrams.

## 1 Rationale

The Web Services paradigm is becoming an important area due to the distributed nature of this technology. In recent years we have seen the birth of a number of languages that somehow provide the developer with ways and means to describe and advertise services on the Web. Some of these languages have already been accepted as standards while others are still evolving and striving to become accepted by the general public.

WSDL [20] is one such language and has been accepted as a standard by the W3C. It is also at the base of other Web Service composition languages, such as BPEL4WS [11], WSCI [19] and OWL-S [15]. These provide the Web Service developer with constructs to create a composition of services, by possibly reusing and or extending existing ones. Creating Web Service descriptions with any of these languages requires a thorough understanding of the language thus making them unpopular with the inexperienced developer. As such developers strive to find tools that could help or even automate the process of creating service descriptions in an easy and transparent manner. A number of tools such as BPWS4J [11] and .NET [14] provide such functionality to both the experienced and inexperienced user. In the area of SWWS though, tools are an important issue since several related technologies are still evolving making things somewhat more difficult.

OWL-S is at present the only language that marries the area of the Semantic Web with that of Web Services. A typical service description in this language consists of four ontologies, namely, Service, Service Profile, Service Model and Service Grounding. The upper Service ontology links to the other ontologies that make up the semantic description. The Profile ontology gives an overview of the service by exposing the inputs, outputs, preconditions and effects as defined in the process model. Other types of information such as a human readable textual description and business contact information are also expressed in this ontology. The Service Model defines the processes that make up the service together with a definition of their respective parameters. Additionally, processes may be of a composite nature in which a choreography of other processes is described

using control constructs defined in the Service Model ontology. Finally, the Service Grounding ontology defines the mapping of the atomic processes and their parameters to the operations and message parts defined in the WSDL of the service.

To create an OWL-S description, the user must be very confident about the underlying markup language. One of the major problems in this case is that the OWL-S specification is still under development and many aspects of this language are still being developed. Such aspects include expressing conditions, effects and dataflow binding mechanisms. DRS [3] and SWRL [9] seem to be the markup languages through which conditions will be expressed. Such ontologies enable to express logical expressions in OWL and hence allow defining also preconditions and effects in the OWL-S Service Model ontology. These evolutions in the language are not simple to grasp and define, as they require the user to be knowledgeable in other logics related areas as well as in OWL-S.

The OWL-S Editor tool that we present in this paper provides the user with a simplified user interface so that the above-mentioned concepts can be developed graphically without exposing the user to manually edit the underlying OWL-S constructs. Massimo Paolucci et al. describe how a primitive DAML-S description can be obtained from an existent WSDL. Since OWL-S essentially builds over DAML-S the concepts expressed in that paper equally apply. The conversion method considered by Paolucci assumes a one to one mapping between a DAML-S Atomic Process (or an OWL-S Atomic Process) and a WSDL operation. This operation will result in a complete Service Grounding specification and a partial definition of the Service Model and Service Profile ontologies. A complete specification of all the ontologies that comprise OWL-S cannot be obtained from a WSDL because the former is richer and expresses a more detailed description of the service. We implemented this methodology in the OWL-S Editor tool through a wizard (OwlsWiz), which abstracts the details of the underlying conversion and lets the user to specify the missing information through a graphical user interface.

Composing an OWL-S service through control constructs defined in the Service Model ontology, is another difficult task. Although tools such as Protégé [16], through its OWL plug-in, and the Web Service Composer [4], enable visual editing of ontologies and services respectively, there exist no tools which enables users to visually create a composite service by using a standard diagrammatic technique without exposing the users to technical information related to the underlying markup. UML Activity Diagrams is a standard diagrammatic technique which is well suited for manipulating the constructs and concepts defined in an OWL-S Service Model ontology. As part of OwlsWiz our tool implements a visual composer which makes use of UML Activity Diagrams [17] to enable a visual composition of the service and the automatic generation of the process markup. This abstracts away the underlying structural details of the Service Model ontology enabling a fast and easy creation of a composite service. The user must still have to know the basic key features behind OWL-S, such as knowledge about preconditions and effects, but there is no need for an in-depth understanding of how these concepts are expressed in the markup.

The rest of the paper is structured as follows. We first give an overview of our tool and then we continue by giving a more in-depth view of both the OwlsWiz and the Visual Composer by making references to a Health Information System service. We discuss issues related to the adoption of a diagrammatic representation in the composer and also other solutions related to the definition of conditions and effects. We then evaluate the tool and make references to other related work. Finally we describe some planned future work and some concluding remarks.

## 2 Overview

Most of the effort was dedicated to the development of the wizard and the visual composer, nonetheless we are also proposing a general architecture, depicted in Figure 1, of a system in which OWL-S

descriptions can be created, validated and also visualized. The idea is to consider an OWL-S description as if it were an application project. Thus the user wont have to load the OWL-S ontologies one by one but rather open a single file and define the paths and file names of the ontologies that comprise the description. Furthermore, the experienced user will be given the facility to directly edit the markup if desired.

The user may either create a description from an existent WSDL or from an empty template as desired. Any required non-local ontologies are fetched from the Web and validated before further processing is done. Such validation is performed with the Jena Toolkit, version 2.1 [7]. During the validation stage, the classes and property identifiers are extracted and maintained in a concepts list. The markup editor uses this list to perform a simple markup completion procedure during editing. The resulting ontologies can be validated and viewed either as a directed graph or as a table of triples. This method is an adaptation of the RDF Validation service [18].

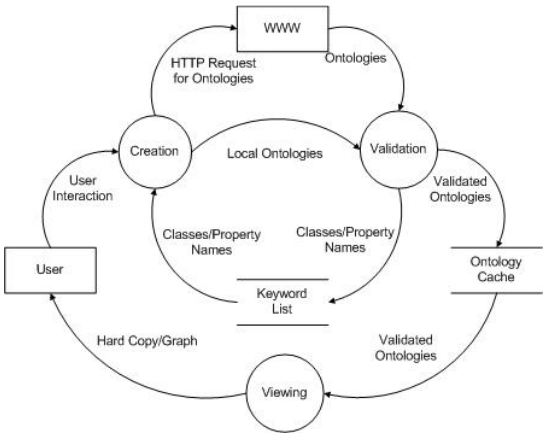



**Fig. 1.** General architecture of the OWL-S Editor

## 3 OWLSWIZ

The core feature in the OWL-S Editor is the wizard, which enables to create a full set of OWL-S ontologies from a WSDL description. This wizard presents to the user a step-by-step process in which the respective OWL-S ontologies are created. Initially, the user has to enter three basic types of information:

- A path to the input WSDL file
- The local target directory where the generated OWL-S description will be saved
- The base URL from where the description will be deployed.

The base URL directory is very important because it will be used to create the URI references of the various ontologies that comprise the OWL-S description. If this URL is incorrect, then the validation will not succeed but the description is created just the same. The second step involves the creation of the upper Service Ontology. Here the user can enter a general comment about the service and also add specific reference to imported URIs if the service is to make use of other ontologies. During the Profile generation step the user has to enter four basic types of information:

– A full human-readable Service Name
– A detailed human readable description of the service
– Business contact information of the entity providing the service
– A URL to the services rating

The business contact information is expressed as a modified VCard ontology. This ontology was simplified to enable to express simple types of information such as name, title, role, email and telephone number. For the service ratings, we are considering that there is some other service that provides rating information about services.

When creating the Process Model, the atomic processes and their inputs and outputs are extracted from the WSDL, but this process requires further interaction with the user, as shown in Figure 2 below. At this stage the user can make use of the Visual Composer to create a composite process by plugging together the extracted atomic processes. It is also possible for the user to visualize a tree representation of the extracted atomic processes and their parameters.

During the creation of the Grounding ontology, no user interaction is required because this ontology is completely generated from the WSDL. However, as in the previous step, the user may view a tree representation of the mappings between the process model and WSDL information. The final step in the wizard simply informs the user that the OWL-S description has been created successfully and shows also the information related to filenames of the ontologies.

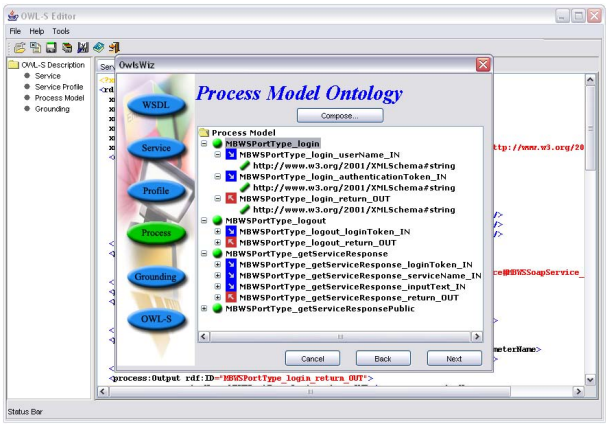



**Fig. 2.** Screenshot of the OwlsWiz at the Process Model generation step

## 4 Visual Composer

The Visual Composer is accessed from the wizard during the generation of the Service Model. The idea is to use the extracted atomic processes and their inputs/outputs to compose them graphically and automatically generate the markup. The graphical representation adopted in the composer involves the use of UML Activity Diagrams and the constructs implemented so far are Sequence, If-Then-Else and Split. These constructs were mapped to specific components in this diagrammatic representation. In order to differentiate between the different types of processes in OWL-S, some graphical changes had to be made to the Action component.

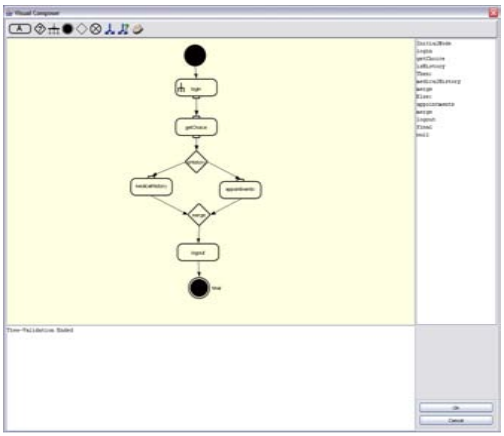


**Fig. 3.** A screenshot of the Visual Composer tool

To illustrate how the diagrams are mapped to the respective OWL-S control constructs and to processes, we will use a fictitious Health Information Service (HIS) [8] whose service composition is shown in Figure 3. A patient may log into the system and access two different services; a service which will return the patients medical history and another which returns the list of medical appointments.

Processes such as login, getChoice, medicalHistory, appointments and logout are represented as Action diagrams. Whenever a dataflow exists between processes, the notion of pins is used. The getChoice process for example has both an input and an output dataflow. OWL-S processes may also have preconditions and effects. We represent these with the rake symbol. This symbol is normally used to indicate that there are some sub-activities underlying the particular action. The If-Then-Else construct is represented using a Decision symbol. Such a construct must however by accompanied by a Merge in order to adhere to UML 2.0 specification which states that no action component can have more than one incoming edge. Although the example in Figure 3 does not use a split control construct, the latter is simply mapped to a fork symbol in the same representation.

### 4.1 Expressing Conditions and Effects

Currently, there is no specification in OWL-S that clearly defines how conditions and effects are to be expressed. This is somewhat changing in OWL-S 1.1 where an Expression construct defines the union between an AtomList (in SWRL) and a Formula (in DRS).

The Visual Composer adopts a subset of these languages in order to present the user with simple scenarios in which conditions, preconditions and effects can be expressed. Given the short time to develop these aspects, we couldnt implement full functionality of the mentioned languages but rather adopt adequate solutions for the most common cases in which the concepts expressed are used.

The condition of an If-Then-Else construct compares an output parameter of a process to a value. At present the wizard does not carry out the transformation from XSD complex datatypes to OWL concepts and this limits the comparison of parameters to atomic datatype values as expressed in the XML Schema Datatype specification. Hence, it is up to the user to make sure that the value to which the parameter is to be compared is of the same datatype. Optionally, the condition in an If-Then-Else may compare the outputs of two process parameters. In both cases, the conditions are

based upon a comparison of an equivalence mathematical operator (such as *equal to* or *greater than*) and the user must also make sure that the parameters are comparable with the selected operator. This type of condition is expressed as a DRS Atomic Formula construct where the subject is a reference to the process output parameter, the predicate is a reference to an equivalence operator and the object is either defined as a value or as a reference to another output parameter. The markup defined in the Process Model ontology at [8] is equivalent to the one generated for the condition of the isHistory If-Then-Else construct in Figure 3. The formula declares a variable of the same type as for the output parameter of the getChoice process and compares this to a value 0 as specified by the user. If the condition holds true, then the service lets the patient access the medicalHistory service.

For preconditions and effects, the composer adopts a rather different solution from the one above. We are assuming that there exist an ontology which defines the predicates and classes needed for the preconditions and effects defined by the particular service. Considering the login process in the HIS service, a typical precondition would involve a valid user name to be provided.

Depending on what the precondition or effect describes, a predicate may have two arguments rather than one. For example, the login process may have a loginUser effect which uses the user name and the respective authentication to let the user access the system. The idea is to use an IndividualPropertyAtom construct as defined in SWRL. This construct references to a propertyPredicate and to the arguments of the predicate, namely argument1 and argument2. In both cases, the respective variables must be declared.

### 4.2 Expressing Dataflow

The current version of OWL-S provides a simple way of handling dataflow. The idea is to use a sameValues property which defines ValueOf constructs which map the parameters of the processes. However, the problem with this technique is that if the same process instance is used in different parts of a choreography, the mapping of the parameters from the different calls to the process can be distinguished. The draft proposal for Version 1.1 is addressing this issue through the use of the Perform control construct which however was introduced late in the development stage of the Visual Composer at which point we had already opted for a different solution. The concept is basically the same as for the above mentioned construct but the idea of tags is used [2].

The use of this technique required an ontology which specifies the constructs to be used. The tag names are defined in the construct TagBind which is a sub-class of ControlConstruct. In the sequence control construct a set of Call constructs define a reference to the process to be executed and also to the associated tag name(s) that will be used to bind the processes parameters. Along with the set of Call constructs, another set of Dataflow construct is expressed. The number of such constructs will depend on how many dataflow bindings exist between the defined processes. For every single binding, a correspondent Dataflow construct must be expressed. This construct defines the properties source and destination. Both of these properties range over a ParameterInst class. The properties defined in this class include the tag name and a reference to the associated process and also whether the tag defines an input or an output flow. The markup of our fictitious HIS Process Model shows how such a technique can be used to define these types of dataflow.

## 5 Evaluation

Time constraints were very tight and there were several issues on which we had no control. Apart from that the API which transforms WSDL to OWL-S [13] was not available for download and

we had to adopt our own solution. We made the assumption that the input WSDL description doesnt define any types node. This limits the editor to handle atomic datatypes as defined in the XSD Schema Datatype Specification. However, the wizard was tested with extreme cases in which an input WSDL contained these types of elements. The result was that the information relative to atomic processes and their respective parameters was extracted correctly but without realizing the complex datatypes as OWL ontologies. This proved that the underlying engine performing this mapping is in fact robust.

The generated ontologies have been validated using the RDF Validation Service [18] and the OWL Ontology Validator [15]. When validating the base OWL-S ontologies, errors such as "Not a valid OWL DL subgraph" have been encountered. This error is generated because OWL-S falls under OWL-Full. As regards the process model ontologies we are still working on features regarding the way conditions, effects and dataflow are expressed. Most of the time we were basing our work on draft proposals that were described in papers (such as DRS and SWRL proposals) and mailing lists (Semantic Web Interest Group). Furthermore, one has to consider the new issues in OWL-S 1.1, specifically the expressions and dataflow binding mechanisms. We are still working on this part in order to generate ontologies that are compliant to OWL-S 1.1.

The use of standard APIs is also limited. During the initial stages of tool development, the Jena Toolkit [7] didnt support OWL Full. Thus we preferred to implement our own simple APIs in order to achieve our goals. Similarly, the OWL-S API [5] has been recently released at which point, most of the classes we needed were ready and was not feasible for us to switch to this API. However, we integrated the Jena Toolkit (version 2.1 with OWL Full support), which validates the ontologies created by the wizard.

The editor in general is limited in some editing features such as an efficient tokenizer for syntax highlighting purposes. Also, an efficient error handling and validation system is missing. The tool is limited to show errors in a separate window rather than pointing out in real time the errors or problems that the ontologies might have.

## 6 Related Work

At present, the number of tools available for creating OWL-S descriptions is limited though we expect that this number will increase in future months, as the OWL-S specification stabilizes and more research work is completed. We have found two tools that are somehow related to out editor: the Web Service Composer [5] and the Protégé OWL plugin [6].

The Web Service Composer consists of two modules, a visual composer and an inference engine. The former is used as an interface between the user and the engine and provides means to dynamically create a flow between a number of services. The inference engine is based on Prolog and handles OWL ontologies by converting the information into RDF triples and storing it in the composers knowledgebase. The engine uses a set of inbuilt axioms to provide service matching capability. It provides for two types of matches, namely an exact and a generic (subsumption) match. The composer makes use of a filtering mechanism to limit the number of services returned after the matchmaking process. The visual interface helps the user to create a flow between composable services and, generated automatically both the process model and the corresponding profile. The tool also provides for the execution of the composed service by invoking the individual services and passing data between the services according to the user-defined flow.

The Protg generic ontology editor enables to create individually the ontologies that make up an OWL-S service description. The tool presents graphically the properties, classes and individual which can be created. The user must be very confident with the constructs defined in the OWL-S

ontologies in order to create a whole description, and this is a bit of a drawback. When imports are added to the ontology, the asserted facts are also imported by making use of the in-built reasoner. These facts are listed in a tree view structure that makes it easy to understand the class hierarchies extracted from the imported ontologies. Furthermore, for service composition there is no standard graphical representation used.

## 7 Future Work

We plan to extend the tool to convert XSD complex datatypes into OWL concepts. This would require further work in other parts of the composer, such as in the parts where process parameters are compared to specific values.

A reasoner is another important feature that we are planning to add to the tool. This feature would enable classification and subsumption of the ontologies which are used by the service. It may also be used to automatically subsume the computed preconditions and effects of processes. This is however an area in OWL-S which is still under research and we hope that future version will fully tackle these issues.

The Visual Composer can be further extended to handle the other OWL-S control constructs. All the constructs defined in OWL-S can be represented using UML Activity Diagrams. Certain constructs are implicitly expressed in the visual composer (such as Sequence and dataflow mechanisms) while the others require the use of explicit constructs (such as Split, If-Then-Else, Split+Join etc). Also, the composer enables only to create a single composite process and hence it could be further extended to enable the creation of multiple composite processes.

Having the OWL-S API available, a slight re-design could be considered in order to make use of this standard official library. However, the OWL-S Editor presents some partial solutions with respect to conditions and dataflow that have not yet been implemented in this API. The idea is to extend the required classes of this library in order to maintain the features presented by the tool.

The user interface in general can be improved in various ways. A feature that will be added is a "help-in-context" system where the user may obtain the help needed at the particular step in the wizard. This would further ease the use of the tool and at the same time enable the user to get more familiar with OWL-S. Another feature is that of a markup completion tool where the editor suggests the lists of classes that can be used in some referenced ontology. Additionally, the reasoner may filter out these classes in order to provide only the classes and property names that are valid in the particular context.

## 8 Conclusion

The OWL-S Editor (OWL-S Editor site) tool is intended for users who need a fast way to create Semantic Web Service descriptions while using standard specifications such as WSDL and UML Activity Diagrams. The main objective behind this tool is to abstract away the underlying OWL-S construct details and to provide an easily accessible way of building complex descriptions. With this effort we wanted to consolidate on previous work and motivate others in adopting similar techniques and methodologies.

## References


1. DAML Services, (November 2003), OWL-S, (Online), Available from: http://www.daml.org/services/owl-s/
2. Drew McDermott, (13 October 2003), Surface Syntax for OWL-S-PAI, (Online), Available from: http://www.daml.org/services/owl-s/1.0/surface.pdf
3. Drew McDermott, (12 January 2004), DRS: A Set of Conventions for Representing Logical Languages in RDF, (Online), Available from: http://www.daml.org/services/owl-s/1.0/DRSguide.pdf
4. Evren Sirin, (19 March 2004), OWL-S API, (Online), Available from: http://www.mindswap.org/2004/owl-s/api/
5. Evren Sirin, James Hendler, Bijan Parsia, (April 2003), *Semi-Automatic composition of Web Services using Semantic Descriptions*, Web Services: Modeling, Architecture and Infrastructure workshop (ICEIS 03). Angers, France
6. Holger Krubnauch, (25 May 2004), Protégé OWL Plugin, (Online), Available from: http://protege.stanford.edu/plugins.html
7. HP Labs Semantic Web Programme, (February 2004), Jena: A Semantic Web Framework for Java, (Online), Available from: http://jena.sourceforge.net/index.html
8. HIS, Health Information Service Ontologies, (Online), Available from: http://staff.um.edu.mt/cabe2/supervising/undergraduate/owlseditFYP/his/
9. Ian Harrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, (19 November 2003), SWRL: A Semantic Web Rule Language combining OWL and RuleML, (Online), Available from: http://www.daml.org/2003/11/swrl/
10. IBM, (9 September 2002), BPWS4J, (Online), Available from: http://www.alphaworks.ibm.com/tech/bpws4j
11. IBM, (2002), Business Process Execution Language for Web Services, (Online), Available from: http://www-106.ibm.com/developerworks/library/ws-bpel1/
12. Martin Fowler, (2003), UML Distilled: A Brief guide to the Standard Object Modelling Language, 3rd ed, Addison-Wesley Pub Co
13. Massimo Paolucci, Naveen Srinivasan, Katia Sycara, Takuya Nishimura, (June 2003), Towards a Semantic Choreography of Web Services: From WSDL to DAML-S, Proceedings of First International Conference on Web Services (ICWS 03). Las Vegas, Nevada, USA, pp. 22-26
14. Microsoft. .NET Framework for Services, OWL-S Editor, (Online), Available from: http://staff.um.edu.mt/cabe2/supervising/undergraduate/owlseditFYP/OwlSEdit.html
15. Sean Bachofer, Raphael Votz, (2003), OWL Ontology Validator, (Online), Available from: http://phoebus.cs.man.ac.uk:9999/OWL/Validator
16. Stanford Medical Informatics, (12 May 2004), Protege 2000, (Online), Available from: http://protege.stanford.edu/aboutus.html
17. Unified Modeling Language, (2003), UML 2.0, (Online), Available from: ¡http://www.uml.org/
18. World Wide Web Consortium, (19 August 2003), RDF Validation Service, (Online),Available from: http://www.w3.org/RDF/Validator/
19. World Wide Web Consortium, (8 August 2002), Web Services Choreography Interface, (Online), Available from: http://www.w3.org/TR/wsci/
20. World Wide Web Consortium, (26th March 2004), Web Service Description Language (Online), Available from: http://www.w3.org/TR/wsdl20/
21. World Wide Web Consortium, (2001), XML Schema, (Online), Available from: http://www.w3.org/XML/Schema

# Non-Monotonic Search Strategies for Grammatical Inference

Sandro Spina and John Abela

Department of Computer Science and AI,
University of Malta

**Abstract.** Advances in DFA learning algorithms have been relatively slow over the past few years. After the introduction of Rodney Price's EDSM heuristic [4], pushing the limits of DFA learning appears to be a very difficult task. The S-EDSM heuristic proposed in [6, 1], manages to improve slightly on what EDSM can do. In this paper we outline our current research results, and propose the use of non-monotonic search strategies in order to improve the success rate of DFA inference.

## 1 Introduction

Grammatical Inference (GI) is an instance of inductive inference and can be described as the algorithmic process of discovering syntactic patterns from a corpus of training examples. GI addresses the following problem:

> **Given a finite set of strings that belong to some unknown formal language $L$, and possibly a finite set of strings that do not belong to $L$, we require a learning algorithm that infers $L$.**

Although GI addresses the whole of Chomsky's language hierarchy we are mainly interested in learning regular languages. Machine learning of grammars finds a variety of applications in syntactic pattern recognition, adaptive intelligent agents, diagnosis, computational biology, systems modeling, prediction, natural language acquisition, data mining and knowledge discovery.

However, while the final measure of success of GI algorithms is the performance of these algorithms on real-world data (as in the case of the examples mentioned above), it is important at times to separate ourselves from real world problems and to study the purely theoretical aspects of learning. These studies can aid research in understanding the limits of what can be learned efficiently by machines, as well as help researchers in the design of algorithms that not only perform well in theory, but also perform well in practice.

## 2 Preliminary Definitions

This section introduces the reader with the terms and definitions used throughout this paper. It is being assumed that the reader is already familiar with the definitions and results in set theory and formal languages, as well as the area of DFA learning in particular state merging DFA learning algorithms. If this is not the case the interested reader is referred to [3] and [6] for an exposure to formal languages and grammatical inference respectively.

### 2.1 Transition Trees

Transition trees represent the set of string suffixes of a language $L$ from a particular state. Transition trees are *mapped* onto each other by taking the state partitions of two transition trees and joining them into new blocks to form a new state set partition. The mapping operation is recursively defined as follows:

**Definition 1 (Map)** *A transition tree $t_1$ is **mapped** onto transition tree $t_2$ by recursively joining together blocks of the set partitions of $t_1$ and $t_2$, for each common string prefix present in $t_1$ and $t_2$. The result of this mapping operation is a set partition $\pi$, consisting of a number of blocks $b$. Each block in $\pi$, is a set of states which have been merged together.*

### 2.2 State Compatibility and Merges

States in a hypothesis DFA are either unlabeled or labeled as accepting or rejecting. Two state labels $A$,$B$ are **compatible** in all cases except when, $A$ is accepting and $B$ is rejecting, or, $A$ is rejecting and $B$ is accepting. Two states are **state compatible** if they have compatible labels. The set of all possible merges is divided between the set of **valid** merges, $\mathcal{M}_\mathcal{V}$, and that of **invalid** merges, $\mathcal{M}_\mathcal{I}$. A valid merge is defined in terms of the transition trees mapping operation as follows:

**Definition 2 (Valid Merge)** *A **valid merge** $M_V$ in a hypothesis DFA $H$ is defined as $(q, q')$, where $q$ and $q'$ are the states being merged, such that, the* mapping *of $q'$ onto $q$ results in a state partition $\pi$ of $H$, with a number of blocks $b$, such that for each block $b \in \pi$, all states in $b$ are **state compatible** with each other.*

## 3 Evidence Driven State Merging (EDSM)

The EDSM DFA learning algorithm developed by Price [4] emerged from the Abbadingo One DFA learning competition organised by Lang and Pearlmutter [4] in 1998. EDSM searches for a target DFA within a lattice of hypotheses (automata) enclosed between the augmented prefix tree acceptor (APTA) and the Universal Acceptor Automaton (UA) [2]. It is assumed that the target DFA lies in the search space of EDSM. It therefore follows, that at least one sequence of merges exists that will lead to the target DFA.

The algorithm starts by constructing an augmented prefix tree acceptor (APTA) and then progressively performing merges. The search is guided by an evidence heuristic which determines which pair of states are to be merged [4]. The heuristic is based upon a score, that computes the number of compatible states found in the transition trees of the two states under consideration. At each iteration, all possible pairs of states in the current hypothesis are evaluated. The pair with the highest evidence score (i.e. compatible states) is chosen for the next merge. This procedure is repeated until no more states can be merged.

## 4 Shared Evidence Driven State Merging (S-EDSM)

S-EDSM was developed with the aim of improving EDSM's heuristic, thus improving on the path traveled through the lattice of automata. S-EDSM is an attempt at a heuristic that tries to minimise

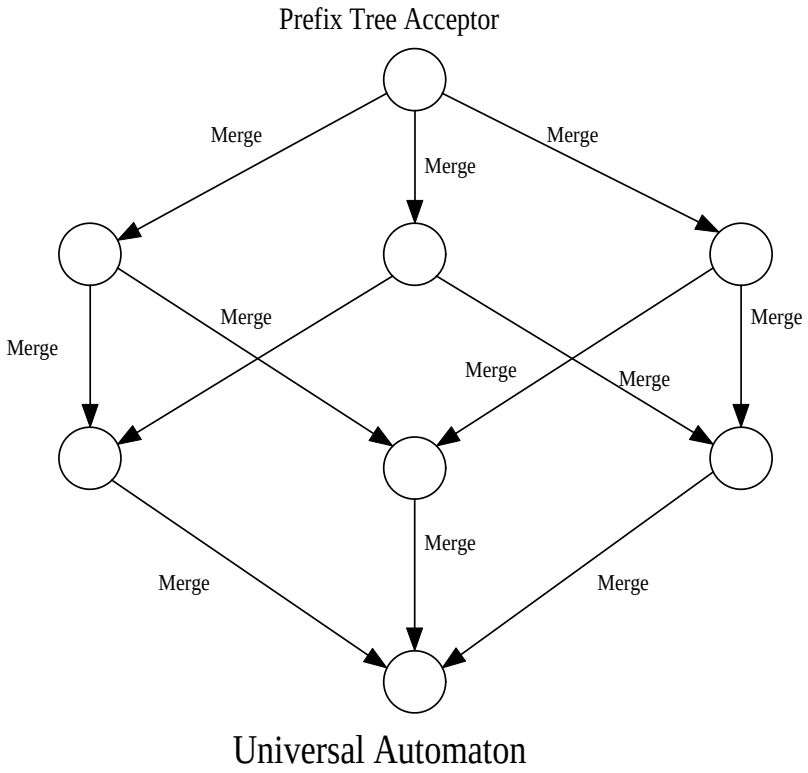



**Fig. 1.** GI Search Space — A Lattice of Automata

the risk of a merge. Undoubtedly there exist no risk free merges, however as opposed to EDSM, S-EDSM tries to share this risk across multiple merges. The main idea behind S-EDSM is that evidence can be enhanced (augmented) by making use of the knowledge gained when checking how valid merges interact. A heuristic is then developed based on some of the properties derived from this analysis.

The most basic interaction between two merges is compatibility. Merge $M$ is said to be *pairwise compatible* to merge $M'$ if after performing merge $M$, $M'$ remains a valid merge in the hypothesis automaton as changed by $M$. More formally two merges are pairwise compatible, if the following property holds:

**Definition 3 (Pairwise Compatible)** *Let $\pi_1$ and $\pi_2$ be the state partitions resulting from the application of the map operator to the two merges $M_1$ and $M_2$ on hypothesis $H$. Let $H_1$ and $H_2$ be the hypotheses resulting from $\pi_1$, $\pi_2$ respectively. $M_1$ and $M_2$ are **pairwise compatible** if for each state $s \in H$, $s \in H_1$ is state compatible with $s \in H_2$.*

The reader is referred to [6] were merge interactions are further discussed. In practice, S-EDSM has yielded a number of interesting results. These include:

- an improvement on the average classification rate for DFA learning, and
- an improvement on the final hypothesis size, whenever the final hypothesis achieves a classification rate of approximately 50%.

Both S-EDSM and EDSM are monotonic search algorithm. This means that, the algorithms do not have any mechanisms by which they can backtrack to previously visited nodes in the lattice and try to follow different paths. Clearly, if all merging decisions carried out throughout the learning process are correctly taken, then monotonic search is ideal. However, the problem lies in the fact that no algorithm can ever tell *when* a wrong decision has been taken (thus the need for the heuristic to optimise the merge choice). We believe that in order to be able to improve DFA learning algorithms, non-monotonic search strategies (backtracking algorithms) have to be used.

## 5 Non-Monotonic Search Strategies

GI algorithms are essentially graph-searching algorithms. Up till now our research has focused mainly on depth-first monotonic searches within the lattice of automata. Clearly, this has its advantages in terms of algorithm complexity. However, we also know that when the training sets being used are sparse, this strategy is not always able to appropriately learn the target language[1]. We will thus be introducing S-EDSM within the framework of a best-first search strategy. Consider the following algorithm, $A^*$, from [5]:

1. Create a search tree, $Tr$, consisting solely of the start node, $n_0$. Put $n_0$ on an ordered list called $OPEN$
2. Create a list called $CLOSED$ that is initially empty.
3. If OPEN is empty, exit with failure.
4. Select the first node on $OPEN$, remove it from $OPEN$, and put it on $CLOSED$. Call this node $n$.
5. If $n$ is a goal node, exit successfully with the solution obtained.
6. Expand node $n$, generating a set, $\mathcal{M}$, of successors. Install $\mathcal{M}$ as successors of $n$ in $Tr$ by creating arcs from $n$ to each member of $\mathcal{M}$
7. reorder the list $OPEN$, according to S-EDSM's heuristic scores and states reduction gradient
8. Go to step 3.

The expansion of any particular node $n$ in the search tree, creates the set $M$ which stores the possible merges which can be carried out from node $n$. These new merges (other nodes in the lattice) are ordered according to the heuristic used in the $OPEN$ list. It is therefore essential to create a heuristic which is able to model how well the lattice is being explored. One very important parameter, which will form part of the heuristic is the 'number of states' gradient[2]. From preliminary experiments, it seems that a heuristic needs to be developed which tries to optimise this gradient. The graph in figure 2, plots the number of states in the hypothesis DFA against the number of merges.

Monitoring the change in states reduction could be a key factor in order to determine when to backtrack and to which node in the search tree to backtrack to. This analysis is currently being carried out.

[1] Refer to [6] for an explanation of GI training sets

[2] Clearly with every merge, states are being absorbed, thus the number of states in the hypothesis DFA decreases

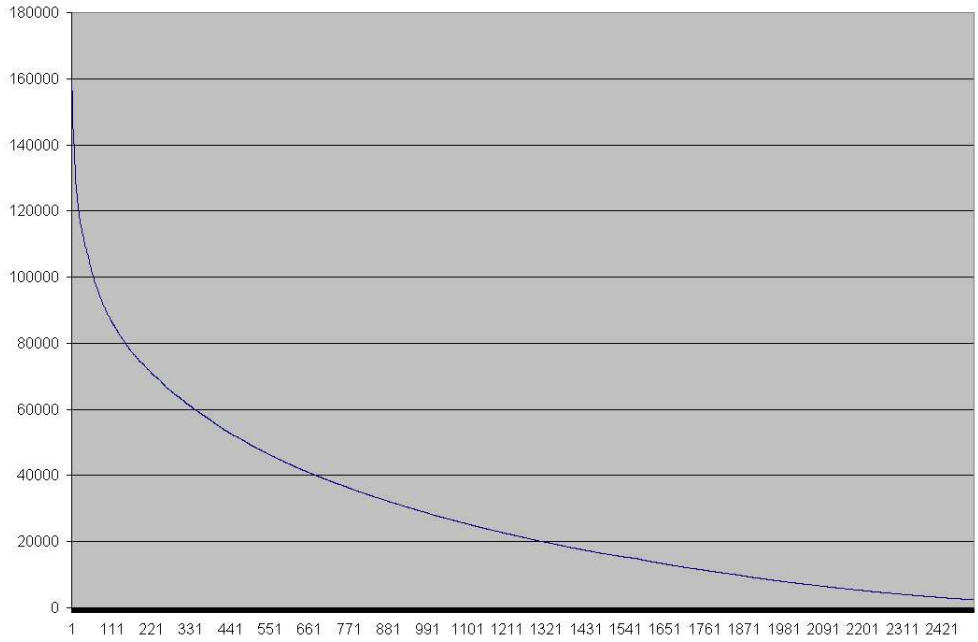



**Fig. 2.** States Reduction Graph

## 6 Conclusion and Future Perspectives

Apart from using other search strategies, the idea behind shared evidence has given rise to another algorithm - this time based on evidence of individual states. Different valid merges label states as either accepting or rejecting. Suppose a number of valid merges label a particular state $s$ as accepting. The idea is that of labeling state $s$ (without merging any states initially) as accepting, depending on the number of valid merges which contribute to this evidence. This state labeling continues until there are a sufficient number of states labeled in the APTA. S-EDSM (or EDSM) is then used to infer the target DFA. The study of this algorithm, with which we can gain further knowledge of where labeling errors are being done, is currently in progress.

Very often, real-world data suffers from the presence of noise. Another question which we'd like to pose is - can S-EDSM perform better by combining information between different merges, when noise is present in the training set? In theory, the extra information gathered from merge interactions with S-EDSM can be used to discover noise in the training set. This is definitely another research area, which can benefit from the analysis of merge interactions.

This paper has presented an overview of the research currently being carried out by the Grammatical Inference Research Group (GIRG) at the CSAI department. Our aim is that of designing better DFA learning algorithms which further push the boundaries of DFA learning.

## References

1. John Abela, Francois Coste, and Sandro Spina. Mutually compatible and incompatible merges for the search of the smallest consistent dfa. *Grammatical Inference: Emphasizing on Innovate Applications. ICGI 2004.*, 2004.

2. P. Dupont, L. Miclet, and E. Vidal. What is the search space of the regular inference ? In *Grammatical Inference and Applications, ICGI'94*, number 862 in Lecture Notes in Artificial Intelligence, pages 25–37. Springer Verlag, 1994.
3. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, Reading, Massachusetts, 1979.
4. K. Lang, B. Pearlmutter, and R. Price. Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm. *Grammatical Inference. ICGI 1998.*, LNAI 1433:1–12, 1998.
5. Nils J. Nilsson. Morgan Kaufmann, United States of America, 1998.
6. Sandro Spina. Merge heuristics : A new heuristic for automata learning, 2004.

# Regulating E-Commerce using XML

Kenneth J. Spiteri

Department of Computer Science and AI,
University of Malta

## 1 Specification of problems from an Electronic Commerce angle

E-commerce, which is short for 'Electronic Commerce', is simply the idea of replacing physical business transactions with electronic business transactions using the Internet. E-commerce cuts through boundaries such as time and geography to put business owners and customers into virtual contact with one another. A key idea behind E-commerce is the ability to take orders and receive payments through an electronic storefront. Philosophically, E-commerce is about openness, connectivity, and integration between businesses and customers.

From the business owner's perspective, E-commerce provides a way to instantly satisfy demand for products, services, and information of each customer individually. From the customer's perspective E-commerce offers convenience, variety, cost savings, and anonymity. Ultimately, E-commerce shifts the power from the merchant to the consumer. E-commerce also provides a way for the business to connect its customers, vendors, suppliers and employees all over the world. The business, as such, is now capable of reaching an infinite number of customers over the Web, seeking out potential markets that were once outside its traditional business boundaries. [5]

E-commerce is a growing area of business and the number of online buyers is increasing exponentially. Further to the issues highlighted previously, that effect the Internet structure as a whole, there are issues specific to the commercial use of the web, that hinder the use of E-commerce to its full potential.

- Lack of experience a major factor.
- Lack of knowledge is significant.
- Internet does not yet meet business needs
- Internet not yet an essential business tool
- Current E-commerce sites not always easily accessible

## 2 Research, Hypothesis, Aims and Objectives

Simplistically, the main hypothesis of this research is that as other areas of the Web have benefited through regulation and standardization, be it network systems, or simply web browsers, the same must hold true for e-commerce. In effect, to standardise e-commerce one must in fact standardise the actual business. Whilst the methodology of every business is specifically dependant on the physical business and as such impossible to standardise, the processes undertaken are not. This means that while all businesses have a procurement process, for example, it's the method this is carried out that varies. Moving down the levels, business processes function on an important element, that is, data. The data required is, in most cases, the same for the specific processes, it is the method of storage and retrieval that is the variable in the equation. Our hypothesis revolves round the

possibility of describing the data transferred in a transaction to satisfy a particular process, and if that description is than standardised, such that it is valid for every business indulging in commerce over the web, then the control on the transfer of data should result in the indirect regulation of e-commerce itself. The current technology evolving round the description of data is the extensible Mark-up language (XML). The application of the meta-language to business promises to be the basis for the description and standardization of data in e-commerce transactions.

### 2.1 Aims

The aims of the research is to:

1. Analyse current e-commerce standards based on XML
2. Design and develop a framework which provides a method for the standardisation of the data, describing the business and its processes, whatever the nature and size of the business.
3. Develop the framework in a way as to allow for its use in conjunction with various technologies and scenarios.
4. Analyse Business processes data requirements, based on Enterprise Resource Planning systems (ERP) for the structure of our framework
5. Develop an evolvable framework, allowing the description of particular data, required for specific transactions that are unique for particular business sectors.

### 2.2 Objectives

The objectives of the research are to show:

1. Current XML based standards, developments and research being done by other bodies on the subject of e-commerce regulation
2. Demonstrate how the framework we will be developing can be of benefit to an online business scenario, such as in a business-to-customer or a business-to-business environment.
3. Highlight principle back-end systems in use today, such as ERP systems, which we will be considering in conjunction with the framework, as the most possible source of data for business transactions in the physical business environment
4. Demonstrate how the developed framework interacts with other technologies, and how this can be used as a building block to more complex solutions.

### 2.3 Tasks

To achieve these objectives we will carry out the following tasks:

1. Develop a framework for the description and standardisation of business data, based on the requirements of business back-end systems, and able to cater for the complexity of an ERP system.
2. Develop and implement an online business system, based on a Business to customer environment with a virtual marketplace scenario, where through our E-commerce framework, multiple businesses can offer their services online, in a simple and cost effective manner. This online business system will generate a direct connection between the customer and the business the customer selected, allowing direct handling of transactions by the business's own back-end systems.

3. This prototype will be developed through the use of particular technologies, and the dot net development environment, with the objective of keeping the system as evolvable as possible, due to the nature of the aforementioned development environment in catering for a heterogeneous coding language base.
4. We will be mapping a number of businesses, with different back-end systems and complexity, to our framework, in this way effectively standardising the data required for their business processes
5. Present other scenarios and technology interactions with our framework, such as the use of SOAP as a platform for other online business scenarios.
6. Perform a full evaluation of the developed online business systems in handling the various back-end systems interactions and the customer transactions. We will also be analysing the performance of the developed framework in handling the various business transactions, in order to test the research hypothesis' validity

## References


1. J. Douglas Heflin, *Towards the Semantic Web: Knowledge representation in a dynamic, distributed Environment*, 2001.
2. W. Emmerich, C. Mascolo, A. Finkelstein, *Incremental Code Mobility with XML.* Dept of Computer Science, University College London, 1999.
3. A. Kumar, J. Leon Zhao, W*orkflow Support for Electronic Commerce Applications*, University of Colorado, University of Arizona, 1999.
4. W.M.P. van der Aalst and A. Kumar, *XML Based Schema Definition for Support of Inter-organizational Workflow,* University of Colorado and University of Eindhoven, 2000.
5. T. Buford, *What Exactly is E-commerce? and How Big is E-commerce?*, Webtomorrow (Url: `http://www.webtomorrow.com/ecommer1.htm`), 2001.
6. Neil Bradley, *The XML Companion*, Addison-Wesley, 1998.
7. Patrick Cauldwell, Vivek Chopra, et al, *Professional Web Services,* Wrox Press Ltd, 2001.
8. Daniel Amor, *The e-Business (r) Evolution,* second edition, Prentice Hall, 2002.
9. K. Sall, *IDM an introduction to XML,* Intranet Journal (Url: `http://www.intranetjournal.com/articles/200002/xml1a.html`).
10. D. Ince, *Issues and problems in E-commerce development. Distributed and E-Commerce applications*, (Url: `http://www.booksites.net/ince/ecomm/Chapters/C1.htm`), 2001.
11. Electronic Commerce Interest Group, *W3 Consortium* (Url: `http://www.w3.org/ECommerce/`), 2001.
12. Thomas Ekdah, *Mark-up Languages, Hypertext and Metaphors. History and future visions*, University of Oslo, 2000.
13. Peter H. Salus, *Paul Baran's Reports, Matrix News, March 2000,* (URL: `http://www.mids.org/pay/mn/1003/baran.html`).
14. FDDI Consortium, *Interoperability lab FDDI resources, InterOperability Lab* — University of New Hampshire, 1997, (URL: `http://www.iol.unh.edu/training/fddi/htmls/index.html`).
15. Piroz Mohseni, *An introduction to XML for Java programmers*, March 1999, (URL: `http://www.devx.com/upload/free/features/javapro/1999/03mar99/pm0399/pm0399.asp`).
16. *UDDI — Universal Description, discovery and integration* Technical White Paper, 2000.
17. Aaron Skonnard, SO*AP: The Simple Object Access Protocol Microsoft Internet Developer* (Url: `http://www.microsoft.com/mind/0100/soap/soap.asp`), 2000.
18. Don Box, et al: *Simple Object Access Protocol (SOAP) 1.1* May 2000 (Url: `http://www.w3.org/TR/SOAP/`).
19. Tarak Modi (*JavaWorld): Clean up your wire protocol with SOAP,* March 2001, (Url: `http://www.javaworld.com/javaworld/jw-03-2001/jw-0330-soap.html`).
20. Erik Christensen, et al, Web *Services Description Language (WSDL) 1.1*, 2001, (Url: `http://www.w3.org/TR/wsdl.html`).
21. Carlos C. Tapang *(Infotects): Web Services Description Language (WSDL) Explained,* July 2001, (Url: `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/wsdlexplained.asp`).

22. Romin Irani, *ebXML and Web Services — The Way to Do Business,* July 2001, (Url: http://www.webservicesarchitect.com/content/articles/irani03.asp).
23. Anders Grangard *(EDI France), et al, ebXML Technical Architecture Specification v1.0.4*, February 2001.
24. Madhu Siddalingaiah Overview of ebXML, August 2001, (Url: http://dcb.sun.com/practices/webservices/overviews/overview_ebxml.jsp).
25. Ariba Inc, *cXML User's Guide,* 2001, (Url: http://xml.cxml.org/current/cXMLUsersGuide.pdf).
26. XCBL.org, *XCBL FAQ*, 2000, (Url: http://www.xcbl.org/faq.html#2).
27. RosettaNet.org, *RosettaNet Overview*, 2001, (Url: http://www.rosettanet.org/rosettanet).
28. A. Keates, *The Key to RosettaNet E-Business,* 2000, (Url: http://www.myrosettanet.com/my).
29. Evan Prodromou, et al, *AuthXML: A Specification for Authentication Information In XML*, November 2000.
30. Microsoft Corp, *The Soap Toolkit*, May 2001, (Url: http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000523).
31. E-Commerce Training 4 business, *E-Commerce strategy,* June 2004 (URL: http://www.E-Commerce-training-4-business.co.uk).
32. Peterson Consulting Group, *Making ECommerce work for you*, June 2004, (URL: http://www.citivu.com/usa/y2k/).
33. Cunningham and Cunningham, *Web services and distributed computing*, April 2004, (URL: http://c2.com/cgi/wiki?WebServices).
34. E. Newcomer, *Introducing Web Services*, 2002 (URL: http://www.awprofessional.com/articles).
35. E. Newcomer, *Web Services versus Other Technologies*, 2002 (URL: http://www.awprofessional.com/articles).
36. W3C, *Soap messages with Attachments,* Dec 2000, (URL: http://www.w3.org/TR/SOAP-attachments).
37. Alan Kotok, *The e-business continuum: Web services, ebXML and EDI. WebServices. Org,* June 2003, (URL: http://www.webservices.org/).
38. Marios Alexandrou, *eCommerce Solutions and Custom Software Development* (URL: http://www.mariosalexandrou.com/).
39. Sue Hamilton, et al, *Entering the Dynamic New E-procurement Marketplace,* Paper 2002, (URL: http://www.ejiva.com/_pdfs/E-procurement%20white%20paper.pdf).
40. David A. Chappell, Vivek Chopra, Jean-Jacques Dubray, Pim van der Eijk, Colleen Evans, Betty Harvey, Tim McGrath, Duane Nickull, Marcel Noordzij, Bruce Peat, and Jan Vegt, *Professional ebXML Foundations*, Wrox Press, Birmingham, UK, 2001.
41. Dieter E. Jenz, *ebXML and Web Services — Friends or Foes?*, WebServices.Org, June 2003, (URL: http://www.webservices.org/index.php/article/articleview/451/1/22).
42. Licus Technologies Inc:, *The E-Commerce Story so far,* Dec 2002. (URL: http://www.licustech.com/Dec-2002.html.126).
43. D.Fensel and C.Bussler, *Web Service Modeling Framework*, (URL: http://informatik.uibk.ac.at/users/c70385/wese/wsmf.paper.pdf).
44. Tim Berners Lee, James Hendler, and Ora Lassila, *The Semantic Web,* May 2001. (URL: http://www.sciam.com/article.cfm).
45. Tim Bray, Jean Paoli, C.M. Sperberg-McQueen, and Eve Maler, eXtensible Mark-up Language XML 1.0, (URL: http://www.w3.org/TR/2000/REC-xml-20001006).
46. Ora Lassila and Ralph Swick, *Resource Description Framework RDF Model and Syntax Specification* W3C Recommendation (URL: http://www.w3.org/TR/REC-rdf-syntax).
47. Dan Brickley and R.V.Guha, *Resource Description Framework RDF Schema,* (URL: http://www.w3.org/TR/rdf-schema).
48. Tim Berners Lee, *Berners Lee and the Semantic Web Vision,* December 2000, (URL: http://www.xml.com/pub/a/2000/12/xml2000/timbl.html).
49. James Hendler and Deborah McGuinness, *The Semantic Web and its Languages* in IEEE Intelligent Systems Trends and Controversies, November/December 2000, (URL: http://www.ksl.stanford.edu/people/dlm/papers/ieee-daml01-abstract.html).
50. Doug Tidwell, *Introduction to XML,* developerWorks.
51. Doug Tidwell, *Web Services — The Web's Next Revolution,* developerWorks.
52. UN/CEFACT and OASIS, *Enabling Electronic Business with ebXML,* Dec 2000, (URL: http://www.ebxml.org/white_papers/whitepaper.htm).
53. Stuart Campbell, *Conducting Business via ebXML. Paper*, May 2001 (URL: http://www.gca.org/papers/xmleurope2001/papers/html/s03-2.html).

54. ebXML Requirements Team. ebXML Requirements Specification May 2001, (URL: http://www.ebxml.org/specs/index.htm).
55. Alan Kotok and David Webber, *ebXML: The New Global Standard for Doing Business on the Internet*, New Riders Press, USA, first edition, August 2001.
56. ebXML Technical Architecture Team, *ebXML Technical Architecture Specification*, February 2001, (URL: http://www.ebxml.org/specs/index.htm).
57. David Webber and Anthony Dutton, *Understanding ebXML, UDDI, XML/EDI.*, XMLGlobal, October 2000, (URL: http://www.xml.org/xml/feature_articles/2000_1107_miller.shtml).
58. James Rumbaugh, Ivar Jacobson, and Grady Booch, *The Unified Modelling Language Reference Manual*, Addison-Wesley, first edition, December, 1998.
59. OASIS, *ebXML Collaboration Protocol Profile and Agreement Technical Committee. Collaboration-Protocol Profile and Agreement Specification*, September, 2002, (URL: http://www.ebxml.org/specs/index.htm).
60. OASIS, *ebXML Messaging Services Technical Committee, Message Service Specification*, April 2002, (URL: http://www.ebxml.org/specs/index.htm).
61. Apache, SOAP Documentation, Apache Software Foundation, 2002.
62. Sang Shin, *SOAP Toolkits*, 2002, (URL: http://www.plurb.com/webservices/SOAPToolkits4.pdf).
63. Samudra Gupta, Using SOAP, 2002, (URL: http://javaboutique.internet.com/tutorials/SOAP).
64. IBM, *Soap Deployment Descriptors*, (URL: http://www7b.software.ibm.com/wsdd/WASInfoCenter/infocen).
65. Scoop Software Corp *ERP White Paper,* ERP Knowledge Base, (URL: http://erp.ittoolbox.com).

# Expanding Query Terms in Context

Chris Staff and Robert Muscat

Department of Computer Science and AI,
University of Malta

**Abstract.** Query expansion is normally performed using a thesaurus that is either generated from a collection of documents, or is otherwise language specific. We present a technique to discover associations between query terms that are synonyms based on past queries and documents common to multiple result sets, to enable query expansion to occur in context.

## 1 Introduction

Furnas describes the main obstacle to improved recall in Information Retrieval as the Vocabulary Problem [3], which arises from the small probability that authors and searchers of a concept describe that concept using the same terms.

Attempts to improve recall frequently utilise a manually or automatically constructed thesaurus [7, 4, 8]. The query is normally expanded prior to its submission and query expansion involves identifying as many alternative new terms as possible that express the same concept. When a term has many different word senses, choosing the correct synonym set can be challenging, as indiscriminate automatic query expansion may lead to a loss of precision [8].

When precision is low it is frequently because the terms specified by the user in the query do not have sufficient discriminatory power to distinguish between relevant and non-relevant documents containing the terms [6]. On the other hand, recall may be low because there may be several different terms that are used throughout a document collection to describe the concept in which the user is interested. However, the user has expressed fewer terms in the query than there are to describe the concept [1].

We explore a different approach to query expansion that assumes that the presence of the same document in the results sets of different queries indicates that some terms in the different queries may be synonyms. If we discover that they are, using WordNet [5], then we can construct a synonym set that will be used to expand a query when the results set of the original, unexpanded, query contains the document. This method allows us to discriminate between alternative candidate synonym sets when a term is ambiguous.

## 2 Scenario

Assume that two users searching for information related to the same concept C express the queries $Q_1$ and $Q_2$ respectively. Assume that no term is common to both $Q_1$ and $Q_2$. Let $R_1$ and $R_2$ be the results sets for $Q_1$ and $Q_2$ respectively. Let $R_{common}$ be the intersection of $R_1$ and $R_2$. Furthermore, let $R_{common}$ be small but non-empty. $R_{common}$ is the set of documents that are relevant to both user queries. Following [2] and [3], we would expect $R_{common}$ to be small if different terms in the original queries potentially describe the same concept but only a few documents contain both

terms. Incidentally, if $R_{common}$ is large, then it means that the majority of documents in the results set of either query contain both terms, in which case users using either term in a query will retrieve approximately the same set of documents. In this case, adding the other term to the query will not improve recall (though it may improve precision). However, if $R_{common}$ is small, and we can demonstrate that a term in $Q_1$ is a synonym of a term in $Q_2$, then in future queries that include either term we can successfully expand the query to include the other term to improve recall. Unlike thesaural expansion techniques, we also require that a document in the results set of the future query was previously seen in $R_{common}$ before we expand the terms.

Furnas recommended that an *adaptive index* is constructed by associating terms that users use with the terms that the system knows about [2]. We use terms supplied by users through queries to discover how the same concept may be described using different terms in other documents and user queries. Our assumption is that different users use different terms in their queries to describe the same concept; that there are documents in the collection that can satisfy each independent query; that some of these documents will contain more than one description of the same concept; and that we are able to automatically identify and associate these alternative descriptions, using, for example, WordNet [5].

## 3 Approach

We assume that a separate vector space-based information retrieval system provides document indexing and retrieval services. When a user query is submitted, the 100 top-ranking documents in the results set retrieved by the information retrieval system are processed to increment the number of times the document has ever been retrieved, and the number of times the document has been retrieved following a query containing each of the terms in the query. The rank at which the document is retrieved is also recorded. We keep a "bag of words" that contains all the terms that have ever been used to retrieve the document. Before adding a new term $q_i$ from the query to the "bag of words" for document $d_n$ we consult WordNet to check if the term is a synonym of any word/s $w_j$ in the "bag of words". For each pair $q_i, w_j$ that are synonyms, we update the synonym sets for $w_j, d_n$ and $q_i, d_n$, adding $q_i$ and $w_j$ respectively. This gives us synonym sets for $w_j$ and $q_i$ in the context of document $d_n$. We also record the word category and sense number of the terms in the synonym set (obtained from WordNet).

To expand some term $q_i$ in a query $Q$, we first submit $Q$ to obtain the initial ranked results set $R$. For each document $d_k$ in $R$, where $k$ is the rank of the document, we retrieve the synonym sets for $q_i, d_k$ along with the document's Inverse Document Relevance (IDR) for term $q_i$ (see next section). The IDR represents the relative frequency with which the document $d_k$ is retrieved in rank $k$ when term $q_i$ occurs in the query $Q$. The synonym set that will be selected for $q_i$ is based on a re-ranked results set based on each document's Document Relevance Weight.

## 4 Document Relevance Weight

There may be multiple documents in a query's initial results set $R$ that provide conflicting synonym sets for a term $t$ in the query. If $t$ is ambiguous or has many word senses, then we need to be able to select the most likely synonym set, otherwise we will provide the user with poor results. Let $\mathcal{W}_d$ be the number of times that document $d$ has appeared in any results set, and $\mathcal{W}_{t,d}$ be the number of times that a document $d$ has appeared in the results set of a query containing term $t$. A document has a rank $r$ in a results set, indicating its relative relevance to the query. Let $\mathcal{W}_{d,r}$ be the number of times that document $d$ has ever been in rank level $r$ in a results set. For simplicity,

we take the significant levels to be top 5, top 10, top 25, top 50, and top 100 ($r_5$, $r_{10}$, $r_{25}$, $r_{50}$, $r_{100}$, respectively). Similarly, $\mathcal{W}_{t,d,r}$ is the number of times that document $d$ has occupied rank level $r$ in the results set of a query containing term $t$.

We calculate $IDR_{t,d}$, a document's inverse document relevance for term $t$, as $\mathcal{W}_{t,d}$ / $\mathcal{W}_d$. This gives us the relative frequency with which the document appears in a results set because of the appearance of term $t$ in the query. For instance, if $\mathcal{W}_d$, the total number of times $d$ has appeared in a results set, is 1000, and $\mathcal{W}_{t,d}$, the number of times $d$ appeared in a results set of a query containing term $t$, is 10, then $IDR_{t,d}$ is 0.01.

A Term-Document Relevance score is $TDR_{t,d,r} = IDR_{t,d} \times \mathcal{W}_{t,d,r}/\mathcal{W}_{d,r}$ where $r$ is the rank level of document $d$ in the results list. We then re-rank the documents in $R$ in the order of their term-document relevance scores. The synonym sets of the top 10 newly ranked documents are then merged according to word category and word sense. The synonym set selected to expand the query term is that of the most frequently occurring word category, word sense pair of the synonyms sets of the top 10 ranked documents.

For example, following a query $Q$ with the results set $R$, we retrieve the synonym sets $S$ for each $q_i, d_j$, where $q_i \in Q$, and $d_j \in R$, for the top-10 ranked documents in $R$. A synonym set is a tuple $category, sense, [syn_0, syn_1, ..., syn_n]$. The synonym set belonging to the most frequently occurring category and word sense combination are used to expand query term $q_i$.

## 5 Discussion

The ideas presented here are exploratory. Our approach is potentially more discriminating than the typical approach to query expansion, because we associate terms expressed by users with documents, and then we use the documents present in the result set following an initial query to select an appropriate synonym set for each term in the query. In limited trials, the same term occurring in different queries is frequently expanded with a different synonym set in each query. We intend to evaluate the performance using standard performance measures, and experiment with limited sharing of synonym sets within a community of users.

## References

1. H. Chen, T. D. Ng, J. Martinez, and B. R. Schatz. A concept space approach to addressing the vocabulary problem in scientific information retrieval: an experiment on the worm community system. *J. Am. Soc. Inf. Sci.*, 48(1):17–31, 1997.
2. G. W. Furnas. Experience with an adaptive indexing scheme. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 131–135. ACM Press, 1985.
3. G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Commun. ACM*, 30(11):964–971, 1987.
4. R. Mandala, T. Tokunaga, and H. Tanaka. Combining multiple evidence from different types of thesaurus for query expansion. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–197. ACM Press, 1999.
5. G. A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38(11):39–41, 1995.
6. M. Mitra, A. Singhal, and C. Buckley. Improving automatic query expansion. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 206–214. ACM Press, 1998.
7. Y. Qiu and H.-P. Frei. Improving the retrieval effectiveness by a similarity thesaurus. Technical Report 225, Zürich, Switzerland, 1995.
8. E. M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69. Springer-Verlag New York, Inc., 1994.

# Reading Between the Lines of Code: Visualising a Program's Lifetime

Nikolai Sultana

Department of Computer Science and AI,
University of Malta

**Abstract.** Visual representations of systems or processes are rife in all fields of science and engineering due to the concise yet effusive descriptions such representations convey. Humans' pervasive tendency to visualise has led to various methods being evolved through the years to represent different aspects of software. However visualising running software has been fraught with the challenges of providing a meaningful representation of a process which is stripped of meaningful cues and reduced to manipulating values and the field has consequently evolved very slowly. Visualising running software is particularly useful for analysing the behaviour of software (e.g. software written to make use of late binding) and to gain a better understanding of the ever-important assessment of how well the final product is fulfilling the initial request. This paper discusses the significance of gaining improved insight into a program's lifetime and demonstrates how attributing a geometric sense to the design of computer languages can serve to make it easier to visualise the execution of software by shifting the focus of semantics towards the spatial organisation of program parts.

## 1 Introduction

Humans find it easy and effective to communicate using drawings — both between themselves and increasingly with computers. Visualisation is heavily used when designing software however we lack the means of visualising a running program in a way that enables us to see the software working in a tangible manner. Such a feature of software would need to be added at an extra effort since code is typically written to work well and not to look good while working. A visual aspect to the program's workings would need to be explicitly created and would not be a natural part of the code hence potentially leading to a gap between the code and the visual representation of its execution.

In certain programmings languages circuit diagrams are used to convey a structure of how components connect and work together to form the program — this in itself lends itself to a notion of visualising what's going on in a program while its running (if the information about program components and interconnections is retained when the code is translated and executed, which is usually not the case).

The motivations of this research revolve around the desire to have richer tools for software developers, tools which are compatible with human cognitive processes and traits in order to improve qualities such as usability and (especially) sustainability in software development. It is widely held that the largest catalyst for the use of visual representations is that humans are quick to grasp the significance of images and reason about them. We tend to prefer to conceive parsimonious visual descriptions conveying elements of information which serve to enrich a representation by abating ambiguity. A lot of motivation for this paper was drawn from [1] in which, apart from the ubiquitously known argument against use of the go to statement, a remark was made about the correspondence between the running program and the code which controls it. This was followed

by another remark about the fairly primitive ways of visualising processes evolving in time available. After almost 40 years our methods and tools have still remained rather primitive, especially compared to the advancements made in the stages leading to the implementation of software. The difference of computing power available in 1968 and now certainly plays a role since to enrich one's experience of computing one requires more computing power.

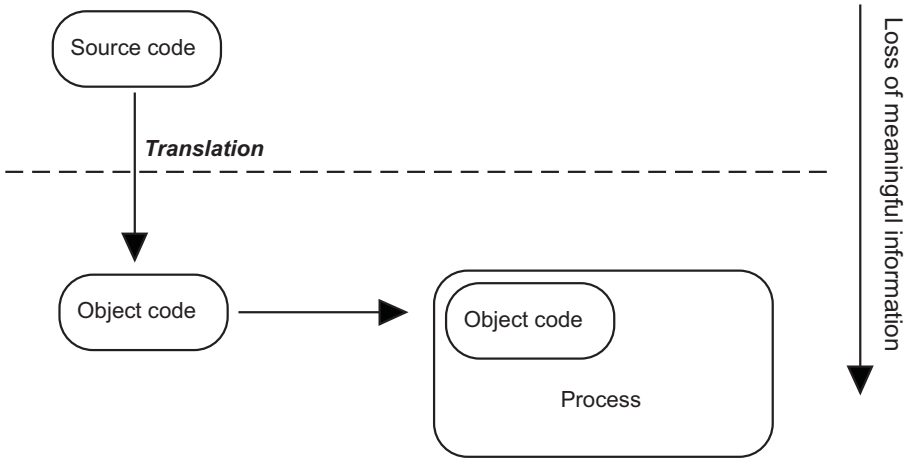
This paper attempts to address the concern of impoverished means of visualising running programs and Dijkstra's appeal to diminish the conceptual gap between the program and the process. By a program's lifetime we understand the time interval spanning a program's execution and not the path from inception leading to implementation and maintenance of a program since we believe that this gives the actual execution of a program the importance it deserves above other stages leading to or following its deployment.

It is generally (and overly generously) assumed that the implementation of software will follow its design. This gap between visualising the design and experiencing the implementation can be partly alleviated by having a means of reasoning about the system's execution in a visual fashion, allowing us (or another system for that matter) to follow the execution and observe discrepancies between our expectations and the program's behaviour. The importance of paying special attention to the quality of the implementation is not only due to it being an artefact which is the product in the software value-chain with which the client has most contact, but [2] presented results indicating that even though rigorous formal techniques are applied during the design of software it is not enough to eradicate problems in the implementation. The implementation of a system deserves more importance than being seen as one of a number of stages constituting the lifecycle of a piece of software, but rather be seen as standing out as the process which delicately hinges on to extending and abiding to other pieces of software which themselves might contain inconsistencies and faults and attempting to insulate as much as possible the end-user from shortcomings in software which talks to our software.

The complexity of software is increased primarily because of the increasing demands being made on it. This complexity tends to cloud designers' foresight — some results can only be predicted with a lot of difficulty due to the large number of interactions carried out by various systems. The dense amount of interactions renders it hard to ascertain what sequence of states could render a system or parts of it non-functional or have them return erroneous results.

Another motivation in this research is that of having an account of the lifetime of a piece of software (analogous to a detailed autobiography) which can be analysed by humans or other software in the event of any failure or to review a program's behaviour (e.g. to improve its performance).

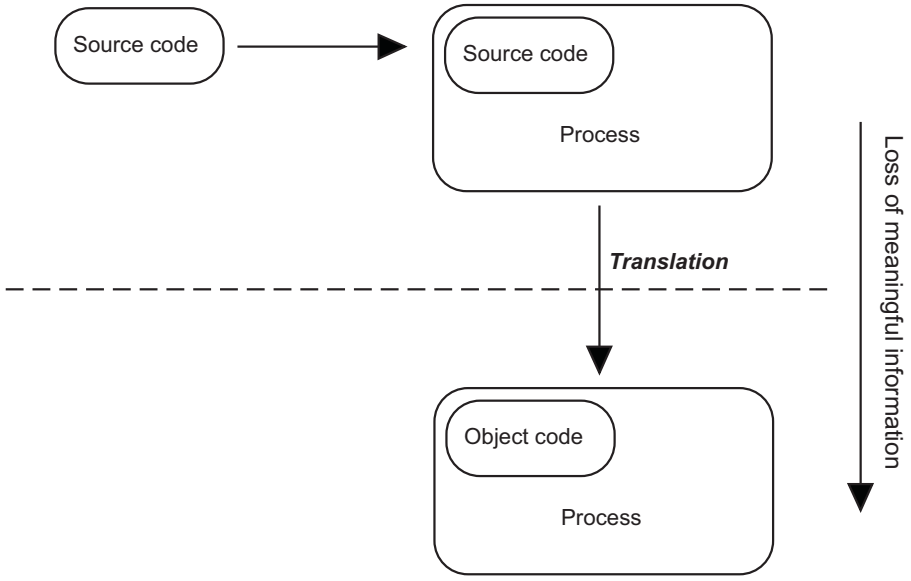
As in any communication channel, a software development process suffers form the arbitrary introduction of noise from shortcomings in the communication between the various stakeholders, making it harder for the implementation to be faithful to the design. Moreover the nature of executing software makes it more difficult to visualise due to it being stripped of symbolic cues about its overall function during translation to a lower level. When software is translated features of the text which render the code readable are removed from the object code. This is understandable since for the execution of the software such information is not required. Information lost in this manner is not retrievable if the translation process were to be reversed (and often such a reversal would return a slightly different of the code if any optimisation were applied by the compiler). Understanding the program's lifetime is more difficult due to the impoverished information we can be provided with and this challenges our ability to excogitate the details of the program's operation.




Certain ways of dealing with the translation of code change the organisation of the diagram shown above by leaving traces of information in intermediate positions along the path which leads to the production of object code. The primary example of this are some of the systems which generate intermediate code which is then executed by a virtual machine — this information can be used to reason about the program's execution in a more meaningful manner (e.g. represent objects and their relationships, methods invoked on them, etc). The intuitive wealth of such descriptions of the program's lifetime is still limited nonetheless and a number of ideas will be presented later on which attempt to push more of the program's execution towards the information-rich side of the representation rubicon.

The Bubble programming language will be used to demonstrate an approach towards computation by including spatial organisation of mobile program components and their relationships in the decisions taken to arrive to the program's behaviour. A different model for running software is used in Bubble's execution environment, which retains information about the program's lifetime and makes it very easy to visualise and reason about.

This paper proceeds to review the widespread notion of readability in programming, outline the state of affairs in runtime inspection of software and its interplay with visualising programs in execution in Section 2. In Section 3 a number of features are identified which would equip a programming language with basic properties to be used to express programs of which execution would be intertwined with their visualisation. Section 4 outlines the implementation of such a language and refers to a case study, moving it to the context of software visualisation since that is the aspect of interest in this paper. Section 5 briefly describes a tool written to generate a graphical representation of the description of a program's lifetime and Section 6 presents a short contrast of advantages and drawbacks of the ideas and approach put forward in this research, together with identifying areas which could benefit or benefit already from work in software visualisation.




## 2 Background

### 2.1 Program readability

A fair amount is emphasis is placed on the importance of producing readable code when teaching programming skills since this improves its maintainability, affects somewhat its re-usability as well as the total cost of ownership of the software — instilling such a sense at the very beginning of exposure to programming makes plenty of sense. Readability serves to assist the perception of the person reading the code (which could be the original author of the code) in interpreting what it does. It is commonly associated with indenting code pertaining to particular blocks, adding in-line comments describing in a natural language what certain bits of the code do and using meaningful strings of characters as identifiers. Readability is also influenced by the choice of operators used and the amount of space used for the expression of the solution to a problem (with obfuscated code usually being on the minimalist side of expression).

Due to the way software is translated it is rarely the case that much thought is given to a notion of process readability since a process (program in execution) is conventionally not for humans to control, but rather handled entirely by the machine during a program's lifetime as it manages its internal workings. Process readability is not addressed by traditional methods of eliciting information about running programs (runtime inspection) because of there not being a ubiquitous system of assigning execution of code with an appearance. In spite of that the appearance and its interpretation is of utmost importance when it comes to visualisation and having the execution of code being a function of an appearance it operates on would allow humans to create more analogies between pictures and operations.

The importance of producing readable code is supported by awareness about our capabilities of cognition — for example [3] had hypothesised that we can keep track of around seven things at the same time. This made it into programming as the Hrair limit and serves to curb the amount of subroutines called from the main program and fundamentally results in facilitating visualising the program when reading the code (since there are less things to track).

### 2.2 Eliciting information about program execution

A visualisation pipeline usually involves gathering data as the first step. Data is then processed and filtered, rendered and displayed. Conventionally the quality of information we can elicit about a running program is quite poor — it does show what the program is doing and how it is doing it but does so in an abstruse fashion.

We can gain insight about a running program (dynamic analysis) in a variety of ways. Initially one can look at a program's memory and processor-time usage. We can gain a lot more information by looking at its communication with the rest of the system — through files, sockets, and other channels (even UI). However, such information tends to be overwhelming and unwieldy since its contents can be arbitrary and were not intended for a visualisation of the running program to be produced — the intention behind them was just to get the program working. A variety of tools can be used to monitor the internal workings of a program, but very often the information accessed in this manner is too fine grained and not intuitive — thus it doesn't convey the big picture about what the program (or significantly large chunks of it) is doing. An often used technique is inserting diagnostic code (e.g. in the form of statements sending values to standard output) to obtain an indication of the values of certain variables during the program's lifetime. Software can be set to send information to a system-wide logging facility (e.g. SysLog) or have their own logging mechanism (e.g. this can be created with ease in AOP as an aspect weaved to an application).

As previously mentioned, a generous interpretation of visualisation of program run-time allows us to consider what we see on screen and what is dumped to file/communication channels as pertinent information about what the program is doing. This contains what we choose to show (e.g. to adhere to specification of software, protocols, etc) and typically offers poor quality of information in terms of what's going on inside due to the way software is/was written to carry things out rather than carry things out by showing it is doing so.

More pervasive means of inspecting and visualising program lifetime are required. This stems from the way software is written and/or expected to be written. An inherent geometric sense would lend itself well to human visualisation abilities by instilling a sense of spatial coherence in the writing of software that will manifest itself in its workings and will take a role in its execution. The following section elaborates on this point.

### 2.3 Software visualisation

Giving software an inherent appearance laden with information about its operation would consolidate it with its modelling aspect and empower stakeholders of software to take better decisions based on the operation of software. Traditionally a variety of graphs have been used to describe the lifetime of a program. Call graphs, control flow graphs and action diagrams are all depictions of the parts of a program which were called while it was running and helps us understand better the sequence of events which characterised the program's execution as well as offering the possibility of aggregating information into a profile. A profile is used to describe an aspect of a system (e.g. time spent executing in a part of a program or else representation of the distribution of statements in a program). Profiling therefore provides us with a summary of the program's execution but the information returned is often too general to deal with specific details.

Learning what's being done in running programs largely depends on the paradigm those programs were written in: it tends to be the case that the more sophisticated the paradigm (in terms of metaphors with the real world) the more meaningful the representations. For example, an execution trace can usually be produced for programs written in any language, but diagrams involving relationships between objects can be produced for programs written in OO languages.

A variety of tools exist which produce representations of program's lifetime for programs written in a variety of languages (e.g. [4–6]). Other tools depart from the source code and provide the user with support for software comprehension [7, 8]. Some tools are designed to augment existing tools and adding new features [9]. The link between software visualisation and cognition has been discussed in [10] together with an approach that attempts to support the latter to arrive at the former.

The way that software is written does make a difference to the representation — for example different approaches to a program's modularity will return different call graphs which would usually be of different complexity (in terms of clutter).

## 3 Programming Spaces

The primary concern in programming deals with getting a working program (which fulfils the requirements which caused it to be written). No thought is given to the program having any inherent visual significance and as previously mentioned information describing the program (albeit limited) is lost during translation and this further limits our ability to visualise running programs in a meaningful manner. Program components are usually designed in such a way that they have to be (at least) conceptually linked to be useful. The components themselves have no location and therefore the information conveyed lacks the relationship between the function and location of elements. If a more geometric view of programming is taken the program components can move and interact and would function according to their location. This pitches computation at a level where spatial organisation and behaviour are related, hence one can influence the other. This would encourage a mechanistic view of computation and allow transferring somewhat visual representations from a description of the problem to a program's implementation.

It follows that a program can therefore be thought of as a description of points (instances of certain types) moving in a space and the execution of the program involves the properties and proximity of such points — thus programming would consist of conveying the description of a system into a computer representation in a way that retains positional information. This information would also contribute to visualising the execution of the program in a meaningful manner.

The semantics of a programming language would need to cater for this information, but the benefit of this would be that the semantics would not only have a functional but also convey a visual concern.

## 4 The shape of a program

Bubble [11] is a language which was developed to model certain natural phenomena [12], transferring descriptions of aspects of our perceptions of such phenomena into a form which could then be used in a simulation. A framework handles the classification of roles and the ways of reproducing (an approximation of) certain dynamics as observed in nature. This language was also used to look at some ideas in Computer Science and a report [13] was prepared about different implementations of a Turing Machine using Bubble. This case will be used as an example of a simple example of a visualisation produced for a program lifetime and a statement about the suitability of this approach for different genres of programs will be made further down.

In Bubble a program is composed of entities having properties amongst which is a position in a space. The program's initiation is a process which involves producing and laying out such structures in the space and then a message can be passed to the execution environment instructing it

to start the program. The program's execution revolves around determining the truth value of certain triggers and then carrying out the appropriate transformation on the space. Triggers involve determining the satisfaction of conditions including property values and/or the composition of the neighbourhood of an object.

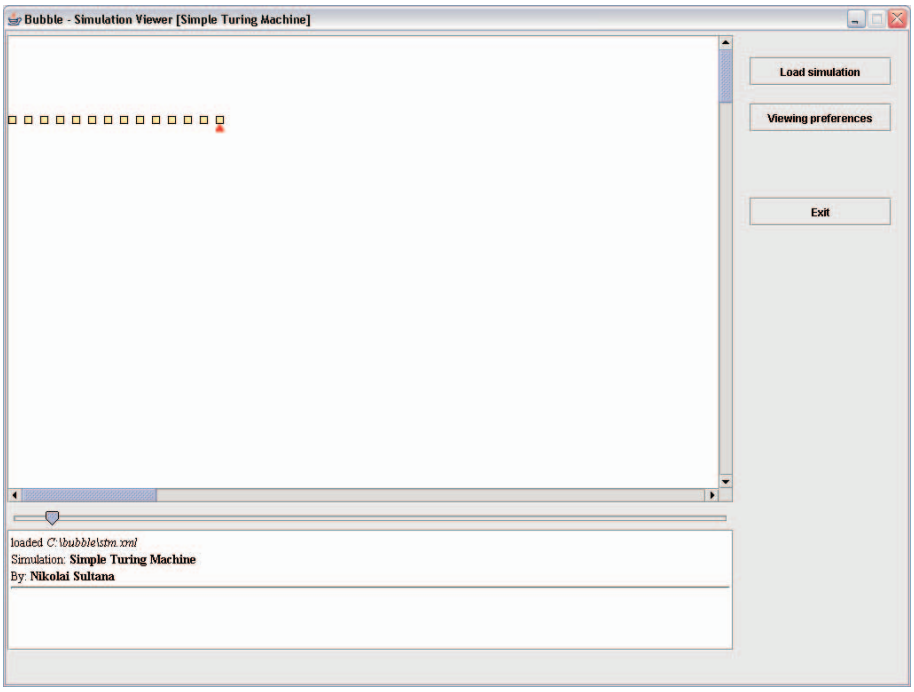
The end result of implementing a program in Bubble is that a program would consist of a distribution of points each with a potential for carrying out an execution. As the program executed the execution environment keeps track of the changes made to the space in which the program is situated and a description of its lifetime can be produced in XML. This can then be transformed using appropriate tools or else passed on to a specifically written tool for visualisation.

The preferred implementation of the Turing Machine in [13] treats two kinds of objects (bits of tape and the read-write head) and the initialisation sets up the tape and read-write head to be at the start position. The behaviour of the head follows a state-transition diagram by specifying the sensitivity of the head to the presence of the tape and its contents and these conditions trigger changes carried out inside the head (current state) or space (movement).

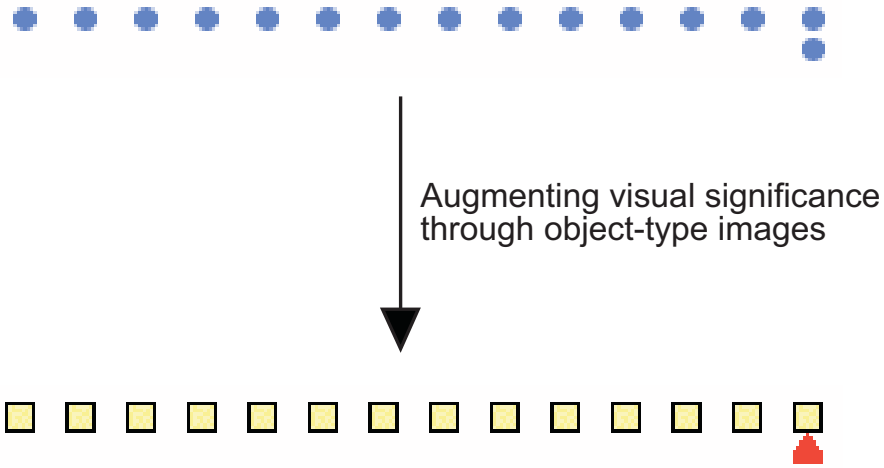
While the specific details of the implementation are provided in [13] the final result of writing the program in Bubble was having a program which worked by manipulating its spatial organisation. Information could be elicited about each stage of the program and the next section will describe a tool used to render this information into a more visually appealing form.

## 5 Visualising the Program's Autobiography

Once a description of a program's lifetime has been generated it is fed to a viewing tool written specifically to interpret it and present the user with a visual representation. A time-line is provided for the user to manipulate and move forwards and backwards in the lifetime of the program. The program's components are represented on the screen and their behaviour can be observed as the user moves across the time-line. Different types of components can be represented using different user-specified images, making it easier to understand what is occurring since the movement of the components is augmented with a pictorial representation pertinent to the nature of the component. Clicking on any component returns details about that component (its properties and their values).




This visualisation of the program's lifetime is only one of several possible processes which can be applied to that information — future work can seek ways of refining the information, filtering it, or possibly create an animation of the interactions between objects since the XML description contains certain information which the current visualisation software does not utilise to the fullest. The aim of this software is to show the possibilities of this approach to programming and attempt to shed some light on its suitability. Zooming into the system to observe its components of components will be required for complex systems and visual cues can be used to convey information to the user and enrich their perspective on the software's operation.




Bubble allows annotations to be added into the code and these will be automatically reproduced in appropriate locations in the description of the program's lifetime. These annotations are displayed according to the current point in the time-line the user is viewing in order to provide more information (as included by the programmer) about the context of that point in the program's lifetime.

## 6 Gains and Drawbacks

The advantage of having a representation of a program's lifetime adds value to the stakeholders of software since it empowers them with insight about the software. Writing software can be easier since it is more clear what is going on when the software is being executed — people can *see* it working. Another advantage of having a rich description of a program's behaviour is that tools can be developed to evaluate and assess the software's behaviour and performance in an automated fashion. In the case of a software failure the representation can be used to examine what led to the surfacing of the failure and take better decisions about fixing the fault.

Since a different (and not commonly used) programming language and execution environment where used in this research there is the disadvantage of the artefacts not being directly applicable to any program. It's also true that producing a visualisation involves a cost: executing and visualising software in this manner, although beneficial, adds a burden on its performance and slows it down. Furthermore, this approach to programming might not be ideal for certain genres of programming, though on the other hand it's always best to use languages inclined to a particular application to get the job done better (or having the qualities to render this more likely). Not having a silver bullet for software [14] also includes not having a single programming language which is apt for any problem: having a variety of languages is beneficial and allows formulating better solutions to more problems. Another criticism of this research might be that a Turing Machine is in itself highly mechanical and might not have served as the best example to use for software visualisation. However many algorithms tend to operate in a mechanical fashion, even if this is not immediately evident, or could be adapted to work that way (rendering them more intuitively comprehensible in the process).

Progress in software visualisation is inclined to produce benefits in diverse areas by allowing one to look at certain problems from new angles. Examples of applications of the information once a reasonable automated description of a program's lifetime has been generated are the following:

- Testing and debugging, preventive and corrective maintenance of software
- Simulations
- Understanding interaction between/within software
- Reflection Reverse engineering Teaching programming and computer literacy
- Refactoring

## 7 Conclusion

The duration of our engagement with the a formulation of a program usually lasts until we compile and run the software. Then testing is carried out to see how well our expectations are met. A dire need exists for more tools and techniques to extend our contact with the domain of processes in order to better understand what goes on during a program's lifetime, why specific failures occur, etc. While visual representations of a program's functioning are used frequently in the stages leading to its implementation very little visualisation is carried out on its execution.

In this research a lot of effort was made to avoid reinventing the wheel and instead approach the problem from identified shortcomings in the nature of software and moving towards software operating as a function of its appearance. Due to the way things evolved in the history of programming there wasn't sufficient opportunity of thinking things through (certain concepts *grew* that way) and it might be of benefit certain ideas are re-evaluated.

Dealing with with the heterogeneity in programming languages makes it difficult to create general purpose visualisation tools for program execution. The goals outlined in the beginning of the paper were met by the approach to programming outlined in the paper, but the cost involved is very high due to lack of alternatives for creating such programs. This makes it difficult to apply this approach to mainstream languages. However, this might be alleviated by focussing on the intermediate level and having the translation process involve creating geometric representations of programs in languages before going on to generate low-information representations of the code (i.e. first talking to a form of middleware, then the code will be translated to a lower level representation).

We believe that elegance in software design can be improved by converging execution of code with a meaningful spatial representation of the running program. This adds another dimension to programming since along with readable code and an intuitive interface the programmer must ensure that the program executes in such a way that the motion and interaction of its components make sense. The readability of code is somewhat extended by this concern since it also covers the dynamic aspect of executing code. Having a description of a program's lifetime and being able to translate that into pictures has several advantages and serve to increase the tangibility of software, which in turn offers plenty of appeal to human cognition. Having a representation of a running program which is compatible with the way we work should offer a good opportunity to understand such a program by giving us the means of looking at it through new eyes.

## References

1. Dijkstra, E. Go To Statement Considered Harmful. In: Communications of the ACM, Vol. 11 (March 1968) 147-148.
2. Pfleeger, S., Hatton, L.: Do formal methods really work? IEEE Computer (January 1997).
3. Miller, G.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. The Psychological Review (1956), Vol. 63, 81-97.
4. Hopfner, M., Seipel, D., Wolff von Gudenberg, J.: Comprehending and Visualizing Software Based on XML-Representations and Call Graphs. Proceedings of the 11th IEEE International Workshop on Program Comprehension (2003).
5. Bertuli, R., Ducasse, S., Lanza, M.: Run-Time Information Visualization for Understanding Object-Oriented Systems. 4th International Workshop on Object-Oriented Reengineering (2003) 10-19.
6. Systa, T.: Understanding the Behavior of Java Programs. Proceedings of the Seventh Working Conference on Reverse Engineering. IEEE Computer Society (2000).
7. Wu, J., Storey, M-A.: A Multi-Perspective Software Visualization Environment. Proceedings of the 2000 conference of the Centre for Advanced Studies on Collaborative research.
8. Storey, M-A., Muller, H., Wong, K.: Manipulating and Documenting Software Structures. Proceedings of the 1995 International Conference on Software Maintenance.
9. Zeller, A., Lutkehaus, D.: DDD-A Free Graphical Front-End for UNIX Debuggers.
10. Eduard Todureanu, M.: Designing effective program visualization tools for reducing user's cognitive effort. Proceedings of the 2003 ACM symposium on Software visualization.
11. Sultana, N.: Bubble Language Specification (2004).
12. Sultana, N.: Modelling and Simulating Systems using Molecule Interactions, Technical report, University of Kent (2004).
13. Sultana, N.: Bubble Case Study — a simple Turing Machine (2004).
14. Brooks, F.: No Silver Bullet: Essence and Accidents of Software Engineering. Computer Magazine, April 1987.

# SharpHDL: A Hardware Description Language Embedded in C#

Christine Vella[1]

Department of Computer Science and AI,
University of Malta

**Abstract.** Embedded domain specific languages have been shown to be useful in various domains. One particular domain in which this approach has been applied is hardware description languages. In this paper, we present such a language embedded in C#, enabling us to describe structurally large regular circuits in an intuitive way. These descriptions can then be automatically used in simulators and verification tools. We show the versatility of the approach by writing a hardware compiler for regular expressions in our language.

## 1 Introduction

Hardware Description Languages (HDLs) are programming languages used to describe a circuit behaviorally, structurally or both. They usually work hand-in-hand with other tools like simulators, which help a designer check that the description of a circuit works correctly. Although powerful HDLs exist, most of them lack to provide tools for verifying properties of circuits.

SharpHDL is an HDL that allows circuit verification by providing a means for model-checking *safety properties*. Safety properties can be defined in such a way that they state that some condition is always true no matter what the situation is [6]. Being an embedded language, SharpHDL is a meta language which allows a hardware designer to generate regular circuits. It is hosted in the C# language, an object-oriented programming language (OOL) which is easy to use and many documentation exist about it. The strong structure it is based upon makes SharpHDL easily extendible and helps the user produce neat circuit descriptions.

In a nutshell, SharpHDL is an elegant and simple structural HDL embedded in C# which incorporates various tools including the possibility of communicating with SMV[2] and Verilog[3] applications. This paper gives a brief introduction to SharpHDL syntax and highlights the various concepts combined to develop this HDL.

## 2 SharpHDL

SharpHDL is an HDL embedded in C# so one can describe a circuit by writing a C# program. Presently, it consists of three libraries:

[1] This work was presented in June 2004 to the Board of Studies of Information Technology at the University of Malta as a Final Year Project for the B.Sc.(Hons.)I.T. degree programme, supervised by Dr. Gordon Pace.

[2] A standard model checker.

[3] A standard HDL.

1. `SharpHDL`, which is the core library. It provides functions to control the structure of circuits as well as the necessary tools to build them, like wires and logic gates. Besides, it provides functions for a circuit designed in SharpHDL to be converted to two formats:
   - *Verilog* format which, given to a Verilog application, can be used to analyze and simulate a particular circuit, produce circuit layout etc.
   - *SMV* format which, given to the SMV application, checks a particular safety property for a wire in the circuit.
2. `LogicGates` library where composite logic gates, such as `Nand` and `Nor` amongst others are implemented;
3. `GenericCircuits` library, which caters for special structures that build a circuit in a regular format from a relatively simple gate.

### 2.1 Syntax Overview

As an introduction to basic SharpHDL syntax let us consider the construction of a half-adder circuit shown in figure 1.

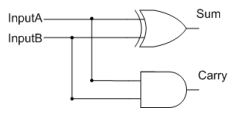



**Fig. 1.** A HALF-ADDER CIRCUIT

A half-adder consists of two input wires and it outputs two wires: `sum` and `carry`. The `sum` result is produced by applying a `Xor` operation over the two inputs; the `carry` result is produced by applying an `And` operation. Following is the code needed to build this circuit.

```
using System;
using SharpHDL;
using LogicGates;

namespace Arithmetic {

  //<1>
  public class HalfAdder : Logic
  {
    //<2>
    public static CellInterface[] cell_interface = {
                inPort("HALF_ADDER_A", 1),
                inPort("HALF_ADDER_B", 1),
                outPort("HALF_ADDER_SUM", 1),
                outPort("HALF_ADDER_CARRY", 1)
    };
    //<3>
    Xor xor;
    And and2;

    //<4>
    public HalfAdder()
    {
        xor = new Xor();
        and2 = new And();
    }

    //<5>
    public void gate_o(Wire inputA, Wire inputB,
                       Wire sum, Wire carry)
```

```
    {
        //<6>
        connect(this, "HALF_ADDER_A", inputA);
        connect(this, "HALF_ADDER_B", inputB);
        connect(this, "HALF_ADDER_SUM", sum);
        connect(this, "HALF_ADDER_CARRY", carry);

        //<7>
        xor.gate_o(inputA, inputB, sum);
        and2.gate_o(inputA, inputB, carry);
    }
  }
}
```

The `HalfAdder` class is created by subclassing the SharpHDL `Logic` class $< 1 >$. The interface to the circuit, which consist of input and output ports are declared using the `CellInterface` mechanism $< 2 >$. Input ports are declared using the `inPort` method, which accepts a string for the name of the port and an integer for its width. In this case, both inputs are one-bit so the width of each port is 1. The output ports are declared using the method `outPort` which works the same as `inPort`. Next, is the declaration of the two instances representing the gates that are needed to build the half-adder: a `Xor` instance and an `And` instance $< 3 >$. Like the majority of C# classes, the constructor of the `HalfAdder` is declared so that it can be invoked by the designer needing a half-adder in his circuit $< 4 >$. Here we create a `Xor` and `And` instance by calling their respective constructors.

Finally, we need to define the structure of the circuit we are designing. We do this in a function accepting four wires: two representing the input wires and the other two representing the output wires. We call this function `gate_o`[4] $< 5 >$. The first thing to do in this method is to connect the wires to the ports using method `connect` $< 6 >$ which accepts three parameters: the gate the port belongs to, the name of the port to which the wire is going to be connected to and the wire variable itself. Finally, we define the structure of the circuit by calling the respective methods that structurally define the `Xor` and `And` gate $< 7 >$. The first line invokes the method `gate_o` belonging to the `Xor` class. This accepts three wires: the first two being the input wires and the third being the output wire. The two input wires to the half adder and the wire `sum` are passed to this method. The same is done for the `And` instance, but this time the two input wires and the `carry` wire is passed to its `gate_o` method.

It is important to note the call to the `SharpHDL` and `LogicGates` library at the beginning. This is necessary to be able to use the classes implemented in them. It is also a good idea to define a `gate` method which although does the same operations as the `gate_o` method described, it accepts three `Wire` objects and creates a new `Wire` object, calls the `gate_o` method and returns the newly created `Wire` object. The method looks as follows:

```
public Wire gate(Wire inputA, Wire inputB, Wire sum)
{
    Wire carry = new LogicWire();
    this.gate_o(inputA, inputB, sum, carry);
    return carry;
}
```

[4] It is a convention of the SharpHDL libraries that every class representing a gate has a `gate_o` method which use the provided input and output wires to describe the particular circuit structurally. SharpHDL also provides the same method without the `_o` suffix. Such methods instantiate an output wire and returns it. Nevertheless they perform the same operation.

## 3 Generic Circuits

In this section we will discuss the various tools offered by the `GenericCircuits` library, which caters for *generic circuits*: circuits with a special regular format built from a relatively simple gate, and which we therefore refer to as *higher-order circuits* [6]. SharpHDL provides four different generic circuits:

1. **Map** — Sometimes, we want to perform a particular operation on each element of a bunch of signals. Instead of defining the operation for each signal, the generic circuit `Map` can be used. `Map` accepts a component having one input port and one output port, and a list of wires. It then constructs the circuit as shown in figure 2.

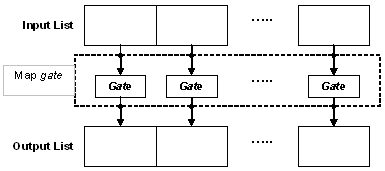



**Fig. 2.** MAP USING LOGIC COMPONENT GATE AND LIST OF WIRES INPUT.

   SharpHDL also implements two other variations of `Map`. One variation accepts a two-input port gate whilst another variation of `Map` accepts a three-input port gate.
2. **Sequence** — As its name suggests, a `Sequence` performs an operation sequentially over a set of input wires. Therefore, it accepts a component having an equal number of input and output ports, and a set of wires which can be accepted by such a component. Figure 3 illustrates the general structure of a `Sequence` generic circuit.

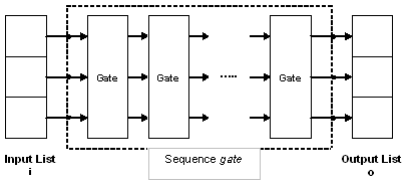



**Fig. 3.** A SEQUENCE USING COMPONENT GATE AND THE LIST OF WIRES I, REPEATED FOR $n$ TIMES. THE OUTPUT IS THE LIST OF WIRES O.

3. **Row** — The `Row` Generic Circuit resembles the combination of the `Map` and `Sequence` structures. It accepts a gate having two input ports and two output ports. Besides, it accepts a separate wire and a list of wires. The circuit formed is illustrated in figure 4.

   SharpHDL also implements two other variations of `Row`. One variation accepts a three-input port gate whilst another variation of `Row` accepts a four-input port gate.
4. **Tree** — There are circuits that can be built recursively, and an example of such is the `Tree` generic structure. The input to this structure is a logic component which takes two wires as inputs and outputs a single wire. It also accepts a list of wire inputs. Basically, the `Tree` structure builds a binary tree of the inputted component, as shown in Figure 5.

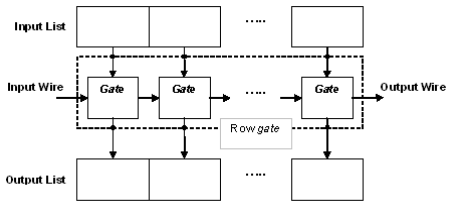



**Fig. 4.** A typical `Row` generic circuit structure

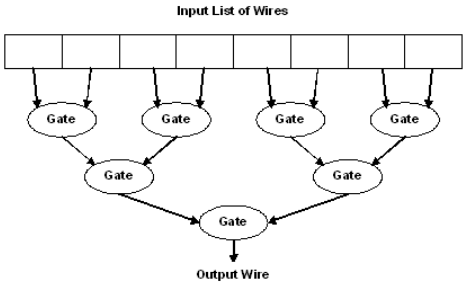



**Fig. 5.** A tree structure using 8 input wires.

Organizing a set of gates of the same type in a tree structure instead of leaving them in linear format, reduces the length of the circuit, such that applying an operator on $n$ wires using a `Tree` structure will have path of approximately $\log_2 n$. On the other hand, if a circuit is generated linearly, the length of the longest path is $n - 1$.

The major benefit of these type of circuits is that they reduce the number of lines of code needed to construct a complex circuit, thus helping the user to generate more elegant code, making it more understandable and in some cases building more efficient circuits as we have seen when using the `Tree` structure. Generic circuits also increase code usability since a particular pattern can be used for several components without having to rewrite any code.

## 4 Hardware Compilation

SharpHDL is a structural language and therefore it provides tools for specifying a circuit by specifying the components to use and the connections between them. Another way of describing hardware is by specifying the behavior of a circuit using *synthesisable behavioral description languages*. These type of languages can be used to write a program and then transform the program to a circuit with the same behavior. Such a process is better known as *Hardware Compilation*. SharpHDL was used to embed a behavioral language which allows a user to build circuits describing Regular Expressions[5].

### 4.1 Regular Expressions Compilation

The language consists of five regular expression constructs:

[5] This example is totally taken from [5].

**Empty String,** which is a string whose length is zero.
**Signal Input,** which is the input signal provided by the programmer of the regular expression. It can either be an output from another existing circuit or an extra parameter to the definition of a particular regular expression.
**Sequential Composition,** which provides for concatenation of two expressions.
**Iteration,** which provides for repetition of an expression for zero or more times. Repeating an expression for zero times produces `Empty String`.
**Non-Deterministic Choice,** which provides for alternation.

### 4.2 Regular Expression Circuits

The circuits that are generated for these constructs after compilation have one input signal `start` and two output signals, `match` and `prefix` as shown in figure 6. The circuit starts sampling the signals when `start` is set to `True`. When the sequence of signals are part of the language being represented by the expression, `match` outputs `True`. The output `prefix` indicates whether the circuit is still active and the parsing of the regular expression has not yet failed. The different constructs compile to a circuit with this structure which implements the necessary operation.

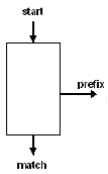



**Fig. 6.** The general structure of a Regular Expression Circuit

Implementationwise, each construct is represented by a class which implements a compilation method according to the circuit it has to produce. To build a regular expression circuit, we must call the classes representing each part of the expression and then call the `compile` method. So, for example, `(`$\varepsilon$ `+ C)`$^*$ `+ (A.B)` is expressed as:

```
RegularExpression example = new Choice(
    new Loop(
        new Choice(
            new EmptyString(),
            new SignalInput(new LogicWire("C"))
        )
    ),
    new SequentialComposition(
            new SignalInput(new LogicWire("A")),
            new SignalInput(new LogicWire("B"))
    )
);
\\Output wires: prefix and match
OutputSignals output = new OutputSignals();

example.compile(new LogicWire(), output);
```

## 5 Model Checking

SharpHDL can also be used in circuit verification by allowing the user to produce an SMV description of a SharpHDL circuit so that, given to an SMV system, he could verify *safety properties* of

the particular circuit. The safety property of a given circuit can be verified by using *synchronous observers*. These are circuits which take the inputs and outputs of the circuit which needs to be verified and decides whether at any time the stipulated property that the given circuit must hold is violated [7].

### 5.1 Case Study: Verifying Adder Implementations

There are many applications where safety properties can be used. One such application is to check that a particular implementation of a circuit is correctly built. This can be done if its output is compared to the output of a text-book example implementation of the circuit, given that both circuits are given the same input. An example is verifying that a *carry-select adder*, which is a special type of adder, is correctly implemented and this is done by comparing it to the classical *ripple-carry adder*. To verify that the former adder produces the correct result, the `sum` and `carry` wires of both adders are passed to an observer which compares the outputs and produces `True` if they are the same. The SMV definition of the whole circuit is given to an SMV application which verifies that for any input, the circuit outputs `True`, implying that the implementation is correct.

### 5.2 Another Case Study: Comparing Two Regular Expressions

In section 4 the Regular Expression Language was embedded in SharpHDL. A clear advantage of embedding languages is that the tools offered by the host language can be accessed by the embedded language without any difficulty. Therefore, circuits created using the Regular Expression Language can be verified. This can be illustrated by verifying that two expressions are equivalent.

An observing circuit *Comparer* can be constructed, which outputs `True` if the expressions are equal, given that both expressions are fed the same `start` signal. Figure 7 displays the framework of this circuit, which accepts two input expressions and outputs a signal wire. This circuit uses an `Equal` gate to check the `prefix`es of the two circuits, and another one to check their `match`es. The outputs from this gate are used as inputs to an `And` gate, since the equivalence property is followed iff both outputs are `True`.

Observing the output wire of this circuit will produce an SMV report, which can be fed to an SMV application for verification. So, given the described circuit, we can check the equivalence property of the two expressions `(ab+a)*a` and `a(ba+a)*`.

Feeding these expressions to the `Comparer` object, we can observe and produce the equivalent SMV for the whole circuit. When this code is verified by an SMV application, it confirms that the output of the circuit is `True` for any situation, meaning that the expressions are equivalent.

## 6 Producing Verilog code using SharpHDL

The two standard HDLs are Verilog and VHDL. Both provide efficient tools for hardware description and simulation [2, 4]. SharpHDL allows the user to generate a Verilog description of a circuit in design.

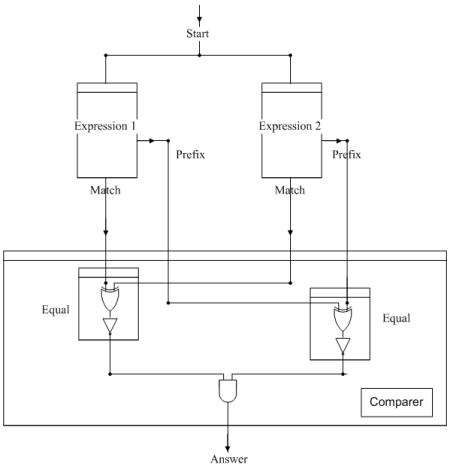



**Fig. 7.** Circuit for comparing two expressions

A user may use the Verilog code for various reasons including that of simulating the circuit in design using a Verilog simulator. SharpHDL does not include its own simulator since well-tested simulators already exist. The Verilog code generation tool provided by SharpHDL gives the user the chance to use the wide range of tools which exist for Verilog and therefore it is still offering the necessary tools for simulation and analysis of circuit.

## 7 Related Works — Comparing SharpHDL

Besides Verilog and VHDL other powerful HDLs exist, amongst which there is Lava and JHDL, both very similar to SharpHDL. This section will briefly describe these two languages and compare them to SharpHDL.

### 7.1 Lava & SharpHDL

Lava [6] consists of an HDL embedded in the functional programming language Haskell. Although, like SharpHDL, it is not possible to describe circuits in such a depth as the standard HDLs, the descriptions are short and sweet. It also offers the facilities of *connection patterns*, which basically are the generic circuits found in SharpHDL.

Lava is very elegant and simple and circuit descriptions are made using Haskell modules. Besides, it provides tools to analyze and test circuits, including simulation and verification. Unlike SharpHDL, Lava links and controls all the tools and therefore the user sees only one language.

### 7.2 JHDL & SharpHDL

JHDL [1] is an HDL embedded in the object-oriented programming language Java. It consists of a set of necessary tools needed to design a circuit in an efficient and user-friendly manner.

Besides providing a set of classes which help a user build an electronic circuit, it allows the user to simulate and test a circuit in design. It also provides a rich CAD suite. Such tools enhance the user-friendliness of the language, a lacking characteristic in SharpHDL. Nevertheless, one must point out that given the strong structure SharpHDL is based on, these can be easily implemented.

Another important difference is that unlike SharpHDL, JHDL does not provide the possibility for circuit verification. On the other hand, SharpHDL does not implement a simulator of its own. Nevertheless, one must point out that a user can still simulate and analyze a circuit by generating Verilog code for a particular SharpHDL circuit description and input it to a Verilog tool.

## 8 Further Work and Enhancements

In the preceding sections, we have illustrated how circuits can be described and used in SharpHDL. Future work on SharpHDL will mainly focus on the following points:

- ***Placement Extensions and Description of Other Non-functional Circuit Properties.*** SharpHDL allows the structural description of any circuit, but it lacks to provide the user the ability to specify placement information. Although automatic placement tools exist, study has shown that user-specified placement information often improves the performance of Field Programmable Gate Arrays (FPGAs). Therefore, it is important that an HDL should include placement information which guides design tools to produce efficient designs.
  Providing explicit coordinate information for every component in a large circuit, nevertheless, can become very tedious. It is therefore ideal to provide the user with the ability to use relative placement information, with high-level descriptions like `Beside` and `Below`. These descriptions allow blocks to be placed relative to each other without the user providing the actual coordinates [8].
  During the design phase, designers need to estimate other non-functional properties like area and timing. Since most of these properties are controlled by information found in wires, detailed information can be kept about wires. SharpHDL could be extended to accommodate such information [3].
- ***Developing a Framework for Hardware Compilers.*** In [5], a uniform framework was proposed within which behavioral languages can be developed, allowing them to be combined together for simulation, synthesis and verification. This is done by embedding the languages in Lava — a HDL embedded in Haskell.

  This concept can also be applied in SharpHDL. We saw how a behavioral language that allows the user to build circuits describing regular expressions was embedded in this language. To allow more behavioral languages to be embedded and combined in this language, a framework similar to the one proposed in [5] can be constructed. Such a framework will impose the functional and non-functional rules, syntax and other methods that hardware compilers should implement.

## 9 Conclusions

Throughout this document, we have described how we managed to merge various concepts and form a strong hardware description language. We have explored the idea of embedding an HDL in an object-oriented programming language by not only embedding SharpHDL in C#, but we

also used SharpHDL to embed another language which allows the user to build circuits describing regular expressions.

To develop the latter language we used standard hardware compilation techniques, where the idea was to write a program in a language and then transform the program to a circuit with the same behavior such that hardware is directly generated from a program.

SharpHDL provides other tools than just a set of classes to design simple circuits. It also includes a special library that constructs different complex circuits with a particular structure, called generic circuits. This library is useful especially to write elegant and short descriptions of large circuits.

An important tool provided by SharpHDL is that of generating input to a model checking application, which could allow verification of safety properties of a circuit. One useful application of this tool is to verify implementations of circuits. It was also used in conjunction with Regular Expression circuits and we were able to check the equivalence of two expressions.

Although SharpHDL does not implement simulation and analysis tools, it provides the user the tool to generate Verilog descriptions which allows the user to make use of the standard well-tested Verilog tools. At the end we managed to compare SharpHDL with Lava and JHDL and we saw that it compares well with such strong HDLs.

The main motivation was to explore an HDL in an OOL that is elegant and simple. By combining various concepts together we managed to develop a strongly-based HDL. We will be exploring the extension of the language for more advanced techniques in the future.

## References

1. Brad Hutchings, Peter Bellows, Joseph Hawkins, Scott Hemmert, Brent Nelson, Mike Rytting, *A CAD Suite for High-Performance FPGA Design (1999),* IEEE Symposium on FPGAs for Custom Computing Machines 1999.
2. Daniel C. Hyde, *CSCI 320 Computer Architecture Handbook on Verilog HDL,* Computer Science Department, Bucknell University, Lewisburg, PA 17837, August 23, 1997.
3. Emil Axelsson, Keon Claessen & Mary Sheeran, *Wired - A Language for Describing Non-Functional Properties of Digital Circuits,* Chalmers University of Technology. Designing Correct Circuits 2004, Barcelona. March 2004.
4. Gordon J. Pace, *Hardware Design Based on Verilog HDL,* Oxford University Computing Laboratory, Programming Research Group. Trinity Term 1998.
5. Koen Claessen & Gordon Pace, *An Embedded Language Framework for Hardware Compilation,* European Research Consortium in Informatics and Mathematics. Grenoble, France, 2002.
6. Koen Claessen & Mary Sheeran, *A Tutorial on Lava: A Hardware Description and Verification System,* August 15, 2000.
7. Nicolas Halbwachs, *Synchronous Programming of Reactive Systems — A tutorial and Commented Bibliography,* Verimag, Grenoble — France. Partially supported by ESPRIT-LTR project "SYRF". Computer Aided Verification 1998
8. Steve McKeever, Wayne Luk & Arran Derbyshire, *Towards Verifying Parameterised Hardware Libraries with Relative Placement Information,* Department of Computing, Imperial College, 180 Queen's Gate, London UK. Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03).

# Distributed Application Reliability on Unstable, Dynamic, P2P-based platforms

Wallace Wadge

Department of Computer Science and AI,
University of Malta

**Abstract.** Application developers are used to a homogeneous, reliable and easily manageable platform on which to deploy their applications. Increasingly however, the need for a highly scalable distributed environment is becoming prevalent. Just as TCP/IP offers reliability over unreliable links, we aim at providing a simple API to hide an underlying unstable, dynamic, P2P platform and present a consistent, reliable view to the applications requesting it. The API should provide robustness and features found in present Grid-systems, which have similar objectives but numerous inherent and logistical differences. Applications would be able to run in a distributed manner over such a platform.

## 1 Introduction

It is often desirable to abstract away the software components from the hardware requirements and allow the latter to be dispersed across a cluster of networks. For example, enterprises might wish to outsource a number of services to third-parties while retaining full control. To this end, significant progress has been made via the use and construction of grids [1], a collection of interconnected systems each providing a number of generic services [2].

The grid environment is designed for dedicated and fully co-operative systems integrated to each other via a consistent interface, relatively stable connections and dedicated links. By contrast, a peer-to-peer (P2P) environment is barely cooperative, node failures are common and the resources are very unbalanced and inconsistent. This is unfortunate since a P2P system implies that the users of the service are the ones supporting it's existence.

While we normally locate resources via a globally unique identifier (such as www.cs.um.edu.mt), in P2P systems we are more interested in locating a resource by a physical property or service; for example, find a Windows 2000 machine with 20MB of free RAM and 1.3 GB of disk space or a machine running the apache web server with an ability to execute cgi scripts. The problem is compounded by the fact that, unlike grids, the underlying platform is highly volatile; by the time the attribute search result is returned to the originator of the request, the target machine in question might have been replaced or removed.

Connectivity shifts might not be a problem for typical P2P applications in use today, namely, those dedicated to simple file sharing or task farming. For an application which requires consistency however, the maintenance required to handle all the routine housekeeping might very well result in overheads too large to perform any useful work. For example the overhead traffic grows exponentially in Gnutella [3], a cluster-based P2P application. It is not unusual for 63% of traffic devoted to platform overheads such as resource discovery and keep-alives [4].

Grid technologies rely on the fact that the deployment of grid systems are motivated by professional communities devoted to providing a degree of trust, accountability and sanctions for problem nodes.

In P2P systems by contrast, individuals have very little incentive for honesty, anonymity is sought and resource sharing is a hit-or-miss affair. Indeed, it is common to have a large percentage of users who use up valuable resources and offer nothing in return. Wilcox [5] showed that the typical P2P node remained connected for just 28% of the time and over 70% contributed nothing in return. It has also been shown that people also deliberately misreport their resources to minimise their chance of contributing to the grid. The end result is that a P2P platform is a hostile environment to work with and presents numerous challenges at a number of stages.

## 2 Proposal

We propose to create a middleware platform whereby applications can seamlessly offload their parallel-based computations across the network in a smooth manner.

Under this proposal, each P2P node advertises its services to its peers. Thereafter, a node can, via a defined protocol (possibly abstracted by the provided library), perform the following tasks:

- Accept a new task
- Accept a task and delegate it to subnodes staying completely out of the loop
- Accept a task but divide the workload
- Add new functionality to the P2P module
- Perform operations in a sandboxed environment.

For example, a P2P node client wishing to perform a complex, time-consuming task can opt to distribute the workload across a cluster of similar P2P nodes. Rather than utilising the P2P platform as a simple file-sharing application, it sends along a program which can be executed by one or more clients — ideally in a sandboxed environment. For instance, suppose we want to employ the quicksort algorithm on a huge set of data possibly scattered around the nodes itself via another module. Therefore from the point of view of the requester we would have:

1. A file pointer or equivalent. This file pointer would just be a way for the requester to indicate the data it is to operate on. This data may also not be physically present on the same machine, the underlying platform would handle this transparently.

2. Now the host would issue the recursive quicksort. Therefore, instead of starting to sort itself and just fetching in the data, it will distribute a sorting code fragment to its immediate peers to recursively sort their parts and instruct each node to return the result already pre-sorted, leaving the original requester to simply merge the parts.

3. Each one of its peers can opt to delegate the given task again.

4. Each peer can then either send the results back to the requester or else to its parent, who will in turn, forward the results. Again the original requester does not see the virtual cluster beneath it, it just sees a way to delegate tasks.

5. At each step the underlying nodes might suddenly fail or new nodes join in; it is up to the platform to hide these problems, for example, by issuing the request again to a new node.

## 3 Research Venues

Significant effort is currently being undertaken to make grids as flexible as P2P systems and, at the same time, considerable work is being undertaken to make P2P technologies as robust as Grids. Several issues still need to be addressed however; these include:

**Robustness** P2P systems are significantly unstable. The bottom layer must be sufficiently robust to try every possible avenue to make a connected network stay connected.

**Trust and anonymity** This is often perceived by users as an important attribute in any P2P-based platform. This is the main focus of Freenet [6].

**Standardization** At the moment there are few real P2P or grid standard protocols with the end result that developers tend to re-invent the wheel each time.

**Extendibility** The platform must not rely on too many assumptions and offer abstractions at each logical layer, for example, by letting the applications themselves define what a service or resource is.

## 4 Conclusion and future work

The project is still in its infancy and therefore many of the aforementioned features are not yet available. To date, the basic framework is in place, a pluggable interface has been implemented and simple tests are possible. Future papers will present further refinements to this model, highlight any problems encountered during development and present detailed performance results.

## References

1. Foster, I. The Grid: A New Infrastructure for 21st Century Science.*Physics Today, 55 (2). 42-47. 2002.*
2. The Physiology of the Grid An Open Grid Services Architecture for Distributed Systems Integration. *Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke.*
3. Adar, E. and Huberman, B.A. Free Riding on Gnutella. *First Monday, 5 (10). 2000.*
4. Early Measurements of a Cluster-based Architecture for P2P Systems. *Balachander Krishnamurthy, Jia Wang, Yinglian Xie*
5. B. Wilcox-O'Hearn. Experiences deploying a large-scale emergent network. *In 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), 2002.*
6. Clarke, I., Sandberg, O., Wiley, B. and Hong, T.W., Freenet: A Distributed Anonymous Information Storage and Retrieval System. *International Workshop on Designing Privacy Enhancing Technologies, Berkeley, CA, USA, 2000, Springer-Verlag.*

# Author Index

Join the club for even better rates

# 1c per SMS

For the first 100 SMS sent per month to your selected 3 **Ready to go club** numbers.*

Ready to go club

**Always the better value**
**www.go.com.mt**

go mobile

*To select 3 Ready to go club numbers of your choice either call go Care on 147 from a go mobile line, or send an SMS containing the word **club** to 4488, or visit www.go.com.mt/mygo