Abstract: This report contains the proceedings of the fifth Computer Science Annual Workshop (CSAW’07) — the research workshop held by the Department of Computer Science and AI of the University of Malta.
# Table of Contents

## Computer Science Annual Workshop 2007

<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Representation Does Matter.</td>
<td>1</td>
</tr>
<tr>
<td><em>John Abela</em></td>
<td></td>
</tr>
<tr>
<td>Automatic Clustering of News Report</td>
<td>11</td>
</tr>
<tr>
<td><em>Joel Azzopardi</em></td>
<td></td>
</tr>
<tr>
<td>Semantic eBroker</td>
<td>24</td>
</tr>
<tr>
<td><em>Bjorn Azzopardi and Matthew Montebello</em></td>
<td></td>
</tr>
<tr>
<td>An Online Educational Portal for Teachers and Students in a Subject</td>
<td>33</td>
</tr>
<tr>
<td><em>Carmel Azzopardi and Matthew Montebello</em></td>
<td></td>
</tr>
<tr>
<td>Language Technologies for an eLearning Scenario</td>
<td>42</td>
</tr>
<tr>
<td><em>Claudia Borg and Michael Rosner</em></td>
<td></td>
</tr>
<tr>
<td>Towards Automatic Extraction of Definitions</td>
<td>52</td>
</tr>
<tr>
<td><em>Claudia Borg, Michael Rosner and Gordon J. Pace</em></td>
<td></td>
</tr>
<tr>
<td>Mi-Learn: An evaluation of an m-learning management system</td>
<td>66</td>
</tr>
<tr>
<td><em>Ernest Cachia, Jonathan C. Borg, Christopher L. Spiteri and Mark Vella</em></td>
<td></td>
</tr>
<tr>
<td>Flexible Learning Systems — An Insight into Personalised Learning</td>
<td>82</td>
</tr>
<tr>
<td>Systems</td>
<td></td>
</tr>
<tr>
<td><em>Vanessa Camilleri and Matthew Montebello</em></td>
<td></td>
</tr>
<tr>
<td>Semantic Web Trust: The next step in web evolution</td>
<td>91</td>
</tr>
<tr>
<td><em>Steven Caruana and Matthew Montebello</em></td>
<td></td>
</tr>
<tr>
<td>Embedded Languages for Origami-Based Geometry</td>
<td>99</td>
</tr>
<tr>
<td><em>Gaetano Caruana and Gordon J. Pace</em></td>
<td></td>
</tr>
<tr>
<td>Transactional CSP Processes</td>
<td>110</td>
</tr>
<tr>
<td><em>Gail Cassar and Patrick Abela</em></td>
<td></td>
</tr>
<tr>
<td>High Performance Staged Event-Driven Middleware</td>
<td>119</td>
</tr>
<tr>
<td><em>Alan Cassar and Kevin Vella</em></td>
<td></td>
</tr>
<tr>
<td>KIKI — A Key to the Integration of Knowledge and Innovation</td>
<td>130</td>
</tr>
<tr>
<td><em>Abigail Cauchi, Jennifer Fenech, Karl Fenech and Luana Micallef</em></td>
<td></td>
</tr>
</tbody>
</table>
Kanban Scheduling System ........................................... 140

Christian Colombo and John Abela


Christian Colombo and Gordon J. Pace

Model Checking Concurrent Assembly Algorithms .................. 162

Joseph Cordina, Stephen Fenech and Gordon J. Pace

An Embedded Geometrical Language in Haskell: Construction, Visualisation, Proof ..................................................... 175

Maria Grima and Gordon J. Pace

An e-Commerce Framework for Wearable Devices ................... 187

Nicholas Micallef and Matthew Montebello

HeDLa: A Strongly Typed, Component-Based Embedded Hardware Description Language ............................................. 192

Gordon J. Pace

Embedding a Hardware Description Language in a Functional Meta-Programming Language ............................................. 203

Gordon Pace and Christian Tabone


Matthew Rizzo

Phrase Extraction for Machine Translation ................................ 226

Michael Rosner and Jo-Ann Bajada

Automatic Interface Generation for Compositional Verification ...... 234

Sandro Spina, Gordon J. Pace and Frédéric Lang

Comparing Title Only and Full Text Indexing to Classify Web Pages into Bookmark Categories .......................................... 248

Chris Staff and Charlie Abela

Author Index ...................................................................... 259
Representation Does Matter

John Abela
Department of Artificial Intelligence, University of Malta
jabel@cs.um.edu.mt

Abstract. In Machine Learning the main problem is that of learning a ‘description’ of a class (possibly an infinite set) from a finite number of positive and negative training examples. For real world problems, however, one must distinguish between the actual instance of the class to be learned and the numeric or symbolic encoding of the instances of the same class. The question here is whether different encodings (or representations) of the instances of a real-world class can actually affect the performance of the learning algorithm. In artificial neural networks (ANNs), for example, it is required that the classes are always encoded as vectors over some field (usually the set of reals). In this paper it is argued that the representation of the class instances plays a very important role in machine learning since it has bearing on two very important issues — the structural completeness of the training set and also the inductive bias of the learning algorithm.

1 The Learning Problem

A learning program is a program that improves its performance with experience. To explain how this is done let us first give an informal definition of the learning problem. Let \( O \) be a domain of discourse and let \( C \) be a, possibly infinite, set of related classes in \( O \). Let \( C \) be a class in \( C \) and let \( C^+ \) be a finite subset of \( C \) and \( C^- \) be a finite subset of \( O \) whose members do not belong to \( C \). We call \( C^+ \) the positive training set and \( C^- \) the negative training set. The learning problem is then to find, using \( C^+ \) and \( C^- \), a class description for \( C \). Of course, in practice, this might be, for all intents and purposes, impossible since if the number of classes in \( C \) is infinite, then \( C^+ \) may be a subset of infinitely many classes in \( C \). In other words, no finite subset, on its own, can characterize an infinite set. We therefore insist only on finding a class description for some class \( C' \in C \) such that \( C' \approximates \) \( C \). This depends, of course, on our having a satisfactory definition of what it means for a class to approximate another.

The above definition covers one of the most common learning tasks. This is the task of finding a function that maps objects that are considered ‘similar’ to the same class value. As Sammut explains in [Sam94], this is the so-called categorization problem referred to by Bruner, Goodnow, and Austin in their seminal 1956 work [BGA56]. To quote Sammut:
Learning experience may be in the form of examples from a trainer or the results of trial and error. In either case, the program must be able to represent its observations of the world, and it must also be able to represent the hypotheses about the patterns it may find in those observations. Thus, we shall often refer to the observation language and the hypothesis language. The observation language describes the inputs and outputs of the program and the hypothesis language describes the internal state of the learning program, which corresponds to its theory of the concepts and patterns that exist in the (training) data. The input to a learning program consists of descriptions of objects from the universe and, in the case of supervised learning, an output value associated with the example. The universe can be an abstract one, such as the set of all natural numbers, or the universe may be a subset of the real world. No matter which method of representation we choose, descriptions of objects in the real world must ultimately rely on measurements of some properties of those objects, for example the length of time a person has been employed for the purpose of approving a loan. The accuracy and reliability of a learned concept depends heavily on the accuracy and reliability of the measurements. A program is limited in the concepts that it can learn by the representational capabilities of both the observational and hypothesis languages.

The observation language is therefore used to encode instances of the universe of discourse while the hypothesis language is used to describe the concepts that are learned (from the data). Learning is therefore reduced to the task of searching through the space of all sentences in the hypothesis language for the sentence (concept) that is consistent with the training data.

2 What is Representation?

The issue of representation deals with the encoding, into some mathematical structure, of the domain of discourse of some learning problem. We cannot overemphasize the fundamental, and very important, distinction between the objects themselves in some domain of discourse and their representation in the observation language, i.e. the numeric or symbolic encoding of the objects (as strings, numbers, vectors, graphs, etc.). Consider, for example, the set of all humans. A human can be encoded, or represented, in a number of ways:

1. Bitmap encoding, i.e. a picture (binary image) of a human,
2. Genome, a string containing the DNA sequence of a human,
3. 2-D vector, e.g. (weight, height), and
4. Attribute n-tuple, e.g. race, complexion, colour of hair, etc.

Depending of what classes we want to consider in a given domain of discourse, an encoding may or may not be appropriate for the purpose of class description. It was shown in [Sch92], for instance, that a 2-D vector (weight, height) is,
in general, sufficient to distinguish the class of male humans from the class of female humans. In fact, when the vectors are plotted in the Cartesian plane it is easy to see that the two classes form two recognizable clusters. It is also conceivable that one can distinguish between males and females from bitmap representations. If, on the other hand, we want to consider the class of humans who have minor thalassaemia, a completely asymptomatic congenital condition of the blood, then such a 2-D vector will not do. Neither would a bitmap. The genome representation of the human would, in this case, be required. This is because people with minor thalassaemia are not externally distinguishable but their DNA contains the thalassaemia gene.

A representation may be better than another because it 'stores more information' relevant to the class in question. It has been argued that there is no form of representation that can 'store all the information' about a given object in the domain of discourse. The closest thing to such a perfect form of representation might be the hypothetical ‘transporter’ device in the Star trek TV series. The transporter device can, allegedly, ‘dissolve’ a human into the constituent atoms, encode the human is some form, transfer this information to a remote location, and then reconstruct the exact human, from different atoms, complete with identical DNA, thoughts, feelings, emotions, memory, experiences, etc.

Another argument why one particular representation may be better than another is that the regularities of a particular class might not be ‘visible’ under some representations [CT95]. We illustrate with an example.

<table>
<thead>
<tr>
<th>String</th>
<th>Gödel Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>ab</td>
<td>4</td>
</tr>
<tr>
<td>aabb</td>
<td>900</td>
</tr>
<tr>
<td>aaabbb</td>
<td>889,350</td>
</tr>
<tr>
<td>aaaaabbb</td>
<td>8,687,348,670</td>
</tr>
</tbody>
</table>

Table 1: Some strings from $a^n b^n$ and their Gödel Numbers.

Table 1 shows, in the first column, four strings from the language $a^n b^n$. It should be easy to anyone with a basic knowledge of formal languages to guess from which language the strings are drawn. The second column shows the Gödel number of the same strings. All we have done is to re-encode the strings as natural numbers. We have used the Gödel number encoding of the strings. This mapping is deterministic and computable in polynomial time. The regularity that was previously visible in the strings now ‘seems’ to have disappeared. It is clearly more difficult for humans to ‘guess’ the correct class if the encoding is not ‘right’. But why? Are learning algorithms also sensitive to the ‘appropriate’ choice of representation?

As Clark and Thornton explain in [CT95]:
Some regularities enjoy only an attenuated existence in a body of training data. These are regularities whose statistical visibility depends on some systematic re-coding of the data. The space of possible re-coding is, however, infinitely large — it is the space of applicable Turing machines.

Many learning algorithms, such as BP ANNs and C4.5, accept only one form of representation of the input data. This means that they allow only one observation language and the implication of this is that the practitioner must always use the same encoding (representation) of the domain of discourse. It is well known that artificial neural networks (ANNs) accept only vectors over the field of reals as input. The questions we ask is: *Does the choice of representation affect learning?* In other words, are learning algorithms sensitive to re-encodings of the domain of discourse and do these re-encodings effect the learning process in any way?

## 3 Does Representation Matter?

In the early days of A.I. (i.e. 1960s to 1980s) most researchers agreed that representation was an important issue in cognitive science. Recently this assumption has been questioned. As Thornton explains in [Tho03b],

*Now there is growing disagreement over the relevance of representation and a steadily deepening polarization of views with respect to the necessity of employing it in cognitive machinery.*

An argument very often used by those who do not see representation as a crucial issue is that if a class in a given domain of discourse can be learned in polynomial time under one encoding (observation language) then it should be learnable in polynomial time in another encoding as long as the mapping from one encoding is deterministic and is computable in polynomial time. This argument, *prima facie*, makes sense. Suppose we have a GI algorithm that learns the language $a^n b^n$ from a finite number of examples in time polynomial in the size of the training set. If we can re-encode the strings as natural numbers using, for instance, the Gödel encoding of the strings, then there must be a polynomial algorithm to learn the same class. Before we present our objections and reservations to this argument we must first clarify a number of points.

1. It is always assumed that, under any reasonable encoding, a class is always *computable*. If, for example, our domain of discourse is the set of all animals and we want to consider the class of cats, then if we encode the animals as strings in some observation language, the set of strings corresponding to cats must be a computable language. This assumption is fundamental since if a class is not computable (under some encoding) then it cannot have a finite description (a sentence in the hypothesis language).
2. If a class is computable and has a finite description under some encoding $E$ then it will still be computable and have a finite description under encoding $E'$ as long as there is a deterministic and computable mapping from $E$ to $E'$.
If the language $a^n b^n$ is a computable language and has a finite description then the set of natural numbers which are Gödel encodings of the strings of this language must also be a computable set with a finite description. The finite description of $a^n b^n$ is a formal grammar while the description of the associated set of Gödel numbers is a set of $\mu$-recursive functions. It is well known from the theory of computation that graphs can be encoded as strings and string as numbers. If fact, for every computable language, the set of Gödel number encodings of the strings in the language is itself a computable set [Sud97].

The argument of those who do not see representation as an issue is that any representation can be used as long as the class to be learned has a finite description in the hypothesis language. This argument is used mostly by those in the connectionist community. This is probably because artificial neural networks, ANNs, only accept vectors as input. Although the above is, in theory, true we believe that researchers in the connectionist and machine learning communities who claim that representation is not an important issue are missing some important points.

We are not claiming that classes become ‘unlearnable’ if the representation is changed. What we are claiming is that the learning problem changes if the representation changes. Let us consider again the language $a^n b^n$ and the set of strings drawn from this language shown in Table 1. It is conceivable that one can design a simple GI algorithm that learns languages of this type. Now what if one is given only the Gödel encodings of this language? Does learning become harder? We claim that these are two different learning problems. If one encodes the strings as natural numbers, the class then changes too. In other words, the specification of the class is tied to the form of representation. The set of natural numbers that are Gödel number encodings of the strings in $a^n b^n$ is no longer the class $a^n b^n$. This set of numbers is computable and does have a finite description (a set of $\mu$-recursive functions) but might require a completely different learning algorithm with a different inductive bias and a different search strategy. This may go some way towards explaining why neural networks perform so badly in certain applications. It has been claimed that a neural network (of the appropriate size) can approximate an computable function in the Euclidean space $\mathbb{R}^n$. In other words, given an ANN of the appropriate size, a set of weights exists that
will compute the function. Even if this is true the problems remain. When one chooses a particular architecture and size of a neural network one is putting an implicit bound on the size of the class description. Suppose, for example, that we have an ANN with \( n \) weights to learn a class \( C \). We are then implicitly assuming that \( C \) can be described (or parameterized) with \( n \) real numbers. This issue has recently received attention from the connectionist community [LGC96].

To demonstrate this we conducted a number of experiments with backpropagation (BP) ANNs. We tried to train a Backpropagation neural network with the complete data sets for 8, 10, 12, and 16-bit even parity mapping. By complete we mean that for each \( n \)-bit parity we labeled each of the \( 2^n \) strings with a \( T \) (True) or \( F \) (False). The reason we tried complete data sets was to find the optimal network size, i.e. number of weights, for each of the mappings. For the 8-bit parity data set we first tried an \( 8 \times 8 \times 2 \) network. This did not converge after running for many hours. We then tried changing the various learning parameters but to no avail. We increased the number of hidden neurons to 16 and then to 32 and got very much the same results. We finally tried a network with 64 hidden neurons and the network converged within a few seconds. This network had \( (8 \times 64) + (64 \times 2) = 640 \) weights. The ANN for 10-bit parity converged in just over 3 mins and required 256 hidden neurons while the ANN for 12-Bit parity required 1024 hidden neurons and converged in just under 12 minutes. We could not use Backpropagation to learn the 16-bit parity problem since the ANN software we used, Brainmaker\(^8\), was limited to 1024 hidden neurons. We then used a Cascade Correlation neural network to learn the 16-bit even parity mapping. The Cascade Correlation network changes its architecture during learning. We then tried to train the networks on incomplete data, sometimes with data sets that contained up to 85% of the \( 2^n \) possible strings. The networks never generalized correctly from incomplete data. This phenomenon has been observed many times [Tho03a]. We were not surprised at the neural network’s inability to learn from incomplete parity data. There is absolutely no reason why a learning algorithm should generalize correctly from incomplete parity data unless the algorithm has the right inductive bias. This point is, most unfortunately, not always well understood. For any given training set of incomplete parity data, there might be thousands of mappings that are consistent with the training set. Why should then the neural network converge to the parity mapping? It will only do so if it has the correct inductive preference bias. Our experiments with the complete parity data sets demonstrated that a Backpropagation neural network can, in fact, represent the parity mapping. Generalization from incomplete training sets is a totally different issue. Neural networks do not learn the parity mapping precisely because they do not have the correct bias. Moreover, the inductive bias of a neural network is fixed and can only be changed slightly by modifying some learning parameters. It is therefore unlikely that anyone can convince a Backpropagation ANN to generalize the parity problem correctly from incomplete training ex-
samples. Some researchers, including Thornton [Tho03a], have proposed various reasons as to why this happens.

Table 2 shows the complete list of mappings with two binary input values. Suppose we would like to teach a learning algorithm the AND function (function 2) and, to do this, we prepare the following training set $(0,0 \rightarrow 0)$ and $(0,1 \rightarrow 0)$. This is clearly an incomplete training set for the AND function. There is absolutely no reason why a learning algorithm should find the correct function from this training set. The reader should note that functions 1, 2, 3, and 4 all are consistent with the training examples. Why would a learning algorithm choose function 2 (AND) over the others unless it had exactly the right inductive preference bias?

It turns out that, for any given $n$, the set of binary strings that are odd or even parity form a kernel language [Abe02]. For example, 4-bit even parity can be described by the following evolving transformation system (ETS) [Gol90] description:

**Kernels:** $\varepsilon$, 11, and 1111.

**Transformations:** $0 \rightarrow \varepsilon$ (Weight 0.0) and $1 \rightarrow \varepsilon$ (Weight 1.0)

![Fig. 1: Enumeration of the search space according to the inductive bias of the learning algorithm.](image)

We ran an ETS grammatical inference (GI) algorithm on a number of incomplete 16-bit parity data sets — par01 had no noise and par02 had 2% misclassification noise. In each case the algorithm found the correct ETS description is less than 3 minutes on a training set of several thousand strings. The training examples were exactly the same that were used, unsuccessfully, to train the ANNs.

**par01** 16-bit parity-problem training set (even parity) used to train Backpropagation neural network. Binary alphabet, very large training set ($\approx 10,000$), multiple kernels, no noise, confluent.

Kernels: 11, 1111, 111111, 11111111, 1111111111, 111111111111,
par02 16-bit parity-problem training set (even parity) used to train Backpropagation neural network. Binary alphabet, large training set ($\approx 8,000$), multiple kernels, 2% misclassification noise, confluent.

Features: 0, and 1.

The ETS algorithm can learn the parity mapping even with very small, structurally complete, data sets of less than 20 strings. We must emphasize that this is because the ETS algorithms so happens to have the right inductive preference bias. When the algorithm learns, it first considers the ‘simple’ ETS descriptions, i.e. the ETS description with the least number of features. It so happens that the parity problem has a very simple ETS description and this was very close to the algorithm’s starting point in the search space. This is depicted in Figure 1. It might turn out that under some other representation, the target class description will be ‘far away’ from the starting point in the search space. The phenomenon was evident when the author was considering ETS learning of chain-code picture languages [Abe96]. A picture such as a rectangle could be represented by its string contour encoding. It this case, the language of all strings that are contour encodings of rectangles (or any other figure) would be a context-sensitive language — sometimes with many productions. A GI learning algorithm that used Occam’s bias would have a hard time learning these classes. This because the algorithm would first consider grammars with small number of productions before grammar with a large number of productions. The target grammar would therefore be ‘far away’ from the starting point in the search space. The same class of figures encoded as graphs could be represented by a very simple graph grammar. This would arguably be much closer to the starting point of the search if one used a learning algorithm with Occam’s bias to search the space of graph grammars. We also strongly feel that the ‘correct’ inductive bias for a given learning problem depends on the choice of representation. If the representation is changed then a new, completely different bias might be required. Most learning algorithms search through the space of class descriptions by first considering the simple class descriptions and progressing to more complex class descriptions. A DFA learning algorithm might first consider DFAs with 1 state, then DFAs with 2 states, and so on. In other words, the algorithm exploits an ordering of the space of class descriptions (i.e the search space). This bias is another example of Occam’s bias. In essence, a learning algorithm’s inductive preference bias specifies the method of enumerating the space of class descriptions. The algorithm stops when it finds a class description consistent with the training set. If one changes the method of enumerating the space, a different class description may be found. This is because, in general, the search space may contain several class descriptions consistent with the training set. In the case of parity problem, a neural network finds the first set of weights consistent with the set of incomplete parity data. It so happens that the mapping
found by the network is not the correct (i.e. parity) mapping. This means that
the network does not have the ‘right’ bias for the parity problem.
Wolpert [WM95] and many others have shown that no inductive bias can achieve
a higher generalization accuracy than any other bias when when considered over
all classes in a given domain. In spite of this, it has been documented that certain
bias do perform better than average on many real-world problems [Tho99]. This
strongly suggests that many real-world problems are homogenous in nature in
that they require very similar inductive biases. This explains why certain learning
algorithms such as ID3 do well on most applications. When learning algorithms
do badly it is very often a case of incorrect inductive bias. The answer, of course,
is to use a learning algorithm that has a variable inductive preference bias. With
these algorithms the user can, to a certain extent, change the algorithm’s bias
by modifying a number of parameters.

4 Conclusions

In this paper we have argued that the choice of representation (the observation
and hypothesis languages) is an important one. The choice of the observation lan-
guage is important since this determines which concepts can be learned [Sam94].
The choice of the hypothesis language is also of importance since, as Goldfarb
explains in [Gol90], the class description learned must be communicable. As
Sammut explains:

A representation that is opaque to the user may allow the program to
learn, but a representation that is transparent also allows the user to
learn

Learning algorithms that fix the observation and hypothesis languages, such
as BP ANNs, are limited in the concepts that they can learn. The choice of
representation influences the inductive bias of the learning algorithm and also
the structural completeness criteria of the training data. Given a real world
problem therefore, the choice of the observation and hypothesis languages must
be very carefully chosen.

References

[Abe02] Abela, John, ETS Learning of Kernel Languages. Ph.D. Thesis, University of
New Brunswick, Canada, 2002.
[ABKG95] Abela, J., Bhavsar, V., Kamat. V.N., and Goldfarb, L., Can a Vector
Space Based Learning Model Discover Inductive Class Generalisation in a Symbolic
[Abe96] Abela, John, Learning Picture Languages, Technical Report TR-CS-9605,
Department of Computer Science and Artificial Intelligence, University of Malta,
1996.
[BGA56] Bruner, J.S., Goodnow, J., and Austin, G.A., Study of Thinking John Wiley
and Sons, NY, 1956.


Automatic Clustering of News Reports

Joel Azzopardi

Department of Artificial Intelligence, University of Malta

Abstract. The automatic clustering of news reports from various web-based news sites into clusters according to the event they cover serves not only to facilitate browsing of news reports by users but may also serve as an initial stage in other complex systems such as Multi-Document Summarization systems or Document Fusion systems. In contrast to the usual scenarios of document clustering whereby the document collections are static or quasi-static, news sites are continuously updated with reports concerning new events. Here, we present a News Report Clustering system which is able to receive a stream of news reports which it clusters on the fly according to the event they cover. New clusters are automatically created as necessary for news reports which are covering ‘new’, previously unreported events. We compare the results of our system to the results produced by a standard K-Means clustering system, and we show that our system performs significantly better than the standard K-Means system even though the K-Means system was supplied with the correct number of clusters that should be produced. In fact, our clustering system obtained an average of 11.95% better recall, 28.68% better precision and 0.89% less fallout than the standard K-Means clustering system.

1 Introduction

Whenever an event occurs, numerous news reports appear on a great number of different news sites on the World Wide Web (WWW) within minutes of the occurrence of that event. Every news agency has its own reporters on the field of action and its own sources. Therefore, every news report may contain information that is unique — i.e. found only in that report.

A reader interested in a particular event will search for different reports on that event to learn as much information about that event as possible. However, it is time consuming for a user to search every news site for reports covering a particular event. The presence of an automatic document clustering system will make this task much easier each since such a system can cluster together those reports from different sources which are covering the same event.

The use of such a system is beneficial not only to facilitate the browsing of a news report, but can also be used as a component within a system with more complex goals such as multi-document summarization and document fusion.

The usual methods used for document clustering, such as Hierarchical Clustering and K-Means Clustering require the collection on which clustering is to be
performed to be static. Therefore, these methods are not feasible for the clustering of news reports since the collection of news reports should be continuously updated with new news reports, and every new news report needs to be clustered immediately for the system to work operationally. Moreover, Hierarchical Clustering and K-Means Clustering require the number of clusters to be known beforehand. When news reports are being received continuously, there can be no way of knowing beforehand the number of different events that are occurring.

To address these issues, we present a system which performs the automatic clustering of news reports. Our system produces a separate cluster for each event that is being covered within the news reports, and all news reports that are covering the same event are clustered together within one cluster. New news reports are read continuously from a number of different sources, and our system is able to detect when a new event is being reported and create new clusters accordingly.

We compared the clusters produced by our system to the clusters produced by a standard k-means system. We found that our system performed significantly better than the standard k-means system, even though the k-means systems had the number of clusters to produce for each corpus specified beforehand.

The remainder of this paper is divided as follows — in section 2 we give a brief overview of other document clustering systems. Section 3 contains a description of our approach to the document clustering, and the justification to this approach. Then in section 4 we describe the methodology of our system. The following section (section 5) contains a description of how we performed our evaluation and section 6 presents the results obtained. Finally, section 7 contains a brief discussion of the results obtained and the conclusions we drew from our results.

## 2 Related Work

The main purpose of document clustering is to generate hierarchies and facilitate browsing from a document collection [HP96]. Moreover, Document Clustering is also used to assist in Information Retrieval [HP96, SKK00, Sal72] — this is based on the proven fact that documents which are relevant to a particular query are found to be more similar to each other than to documents not relevant to that query [HP96, APR99].

The derivation of the appropriate set of categories into which a document collection is to be clustered is essential in Document Clustering so as to simplify the classification task [BB63]. This set of categories should be determined by the document collection itself, or by the system’s purpose. [LA99] stresses for the need to have a system which can discover and approximate topic hierarchies using unsupervised clustering methods.

The phases in Document Clustering are [LA99]:

- the extraction of features from the documents,
- the mapping of the documents to high-dimensional space, and
- the clustering of the points within the high-dimensional space.
The document features are usually represented by the set or a subset of the words they contain [Sal97, BB63, LA99, Sal72, VF95, APR99, TK05]. [BB63] advocates the use of pre-selected terms to represent each document. On the other hand, [Sal97, LA99, Sal72, VF95, TK05] extract all the terms from the documents to act as content-representatives — albeit using some filtering sometimes, such as using only the highest \( n \) terms. Besides the use of single terms as document labels, [Sal97] also suggests the use of term phrases.

[Sal97] claims that the importance of a term as a representative of the content within that directory is related to the occurrence frequency of that term within the document (or document excerpt) and the occurrence frequency of that term within the entire document collection. In view of this, the importance of a term may be calculated using the *Inverse Document Frequency* (IDF), which is the ratio of the term occurrence frequency within the document in question to the occurrence frequency of the term over the whole document collection. This IDF is also used for term weighting by [APR99, LA99, Gul05]. [Gul05] calculates the term weights by utilizing the TF.IDF measure that is centered on the DMOZ\(^1\) Categories. The advantage of this approach is that one does not need to have the entire document collection at hand to weight the terms which will be used to represent the documents.

The occurrence frequencies and *IDF* information of each term may be stored in an Inverted Index. An Inverted Index is a sorted list of terms, and along with each term other related information such as the occurrence frequencies of that term and its *IDF* information. Inverted Indexes are used in standard document retrieval and they also include a postings list for each term — i.e. a list of links to the occurrences of that terms within the document collection [Sal72].

Besides the IDF, [Sal97] also describes the term specificity in the context of the representation of documents in high-dimensional space. If broad terms are used to represent documents, they will lead to very small distances between the points representing the documents in the high-dimensional space. On the other hand, if the terms used to represent documents are too specific, the points in the high dimensional space will be too far apart from each other. In view of all this, [Sal97] defines the *Term Discrimination* measure which is the difference occurring in space density if a term which was previously considered to be representing a document, is not considered anymore.

An alternative method of document representation is by using *Lexical Chains* [SC01]. A Lexical chain is a set of semantically related words found within a text. To build lexical chains representation of a document, each term within that document is processed chronologically, and it is added to an existing chain or made the seed to a new chain. The criteria used for adding a term to an existing chain is by identifying a semantic relationship (using WordNet\(^2\)) between the term in question and the chain’s seed term, or by establishing a co-occurrence relationship within close proximity of that term with the chain’s terms. By analyzing

---

1 Open Directory Project — [http://www.dmoz.org](http://www.dmoz.org)
2 [http://wordnet.princeton.edu/](http://wordnet.princeton.edu/)
the lexical chains, one can identify those chains which have the most members as being representative of the more salient terms. After the documents representations are constructed, the next step would be that finding the similarities between the different documents. In [LA99,APR99], this is done using Cosine Similarity (refer to [Sal71]). Quite similarly, [BB63] performs correlation between matrices of the index terms’ occurrence frequencies. [SC01] finds document similarities by comparing the documents’ lexical chains together. According to [SKK00], there are 2 main clustering techniques — namely:

- **Hierarchical Clustering** — this technique produces *hierarchies* and is further split into:
  - *Agglomerative* — whereby we start with each point being in a separate cluster, and at each step, the most similar pair of clusters are merged together, and
  - *Divisive* — whereby we start with all the points being in a large single cluster, and at each step we split the cluster such as to maximize the intra-cluster similarity.

- **K-Means Clustering** — whereby we start with k points as the initial cluster centroids, and assign all the points to the nearest centroid. Then, a number of passes are made whereby the cluster centroid is recalculated and the cluster membership of each document is also recomputed. The application of this technique is also discussed in [LA99,Gul05,HP96,SC01,Sal72].

[Gul05] utilizes a variation of the K-Means Clustering approach whereby similarity thresholds are used to warrant cluster membership, and the number of clusters is not known beforehand. The use of similarity thresholds to warrant distances is also discussed in [LA99,SC01,HP96]. [HP96] also uses a junk cluster which will contain those documents that can not be clustered. In contrast to utilizing the similarity between each document and the cluster centroids to decide if that document warrants membership within that cluster, [APR99] describes two other membership policies:

- **Single Link** — whereby a document is considered to be part of a cluster if it is related to at least one document within that cluster,
- **Average Link** — whereby a document is considered to part of a cluster if it is related to at least the average number of documents within that cluster.

In contrast to the Clustering Methods described above, [Gul05] discusses a news classification system which uses training but in a dynamic manner. The problem of [Gul05] is to classify news reports as they are received from the multiple sources. Now some news reports have classification information defined explicitly contained within themselves — for example an article may be marked to be part of “Sports News” or “U.S. News” or “British News”. When such classification information is identified, the system switches to training mode. Then, when news reports are received which have no classification information contained in them, their classification is decided based on the training the system has incurred with the other news reports.
In the usual scenario in which Document Clustering is analyzed, there is a static document collection which is to be clustered. However, in the cases where the document collection is dynamic — i.e. documents are being added, and/or others are being retired (removed from the document area) continuously — things get more complicated. A case in point is when we have a system which is working on news reports which are continually being received from streams such as from RSS feeds, such as the News Search Engine system described in [Gul05].

As more documents are added to clusters, eventually a complete reorganization of clusters will be needed [Sal72,LA99]. Furthermore, [VF95] shows that when using the normal weighting schemes (using the vector space IR model), term weights updates are expensive — in fact, adding a single document can affect a large part of the Inverted Index.

The suggested solutions to this problem include:

- The use of pre-computed term weights without any update to these weights ([VF95]). This approach is also utilized by [Gul05] which uses term weights calculated by parsing the DMOZ. [Sal72] also discusses this option, whereby new documents can be added to clusters without changing the cluster representations.
- Updating only the weights of existing terms in the cluster profiles but without introducing new terms to the cluster profiles [Sal72].
- Keeping the Inverse Document Frequency (IDF) information separate from the document term weights, and IDF weighting is only applied to query terms, thus avoiding the recalculation of document term weights when a new document is introduced to the collection [VF95].
- Updating the Term weights and the terms' IDF information only intermittently [VF95].

3 Our Approach

As we already mentioned in section 1, we are presenting here a system which performs the automatic clustering of news reports. Within our system, news reports are being continuously downloaded from a number of different news sites. Our clustering system processes each report and clusters it with those other reports which are covering the same event, or otherwise clusters it into a new cluster on its own if the event covered by that report has not been covered any other report yet.

The main issues within our system are that:

- the document collection is dynamic since new news reports are being downloaded continuously and presented for clustering,
- since new news reports are being downloaded continuously, the news reports must be categorized on the fly since at no point will the downloading of news reports stop to allow for the clustering to proceed, and
- the number of clusters can not be known beforehand — when a news report describing a ‘new’ event appears, a new cluster must be created automatically for this report.
The clustering approach utilized by our system is a variation of the *K-Means Clustering* approach, and uses a similarity threshold between the individual document and the cluster centroid to warrant cluster membership, as described in [Gul05,LA99,SC01,Gul05,HP96]. In our opinion, *Hierarchical Clustering* is not appropriate for a dynamic system such as ours since hierarchical clustering will require the entire document collection to be present before it can start clustering the documents, and the knowledge before hand of the level where one needs to stop the clustering procedure. The K-Means approach has been adapted to work on a dynamic collection — new clusters are created and necessary, and old clusters are not considered for processing anymore if they have not been modified after a certain period of time.

In this way, our system can be adapted to perform new topic detection and tracking as well. If a major event occurs in the news such as a terrorist attack on a city in the United States, all reports covering this event will be clustered together in a single cluster. Moreover, news reports which are issued later on in time but contain more information that has been uncovered are also clustered within this same cluster. On the other hand whenever a report appears which is covering a new event (hence a new topic), a new cluster is created for that report.

The size of each produced cluster depends on the amount of reports issued that are related to the event being covered by the reports within that cluster. The number of reports issued is dependent on various factors such as the significance of the event in question, and also certain events such as the uncovering of a new piece of information on that event also triggers new interest in that event. The temporal effects of news reports are also discussed in [yGGLL01a] and [yGGLL01b].

To represent each news report in high-dimensional space, the use of pre-selected terms requires additional effort and is not feasible for us since our system processes news reports as they are read from the feeds. Therefore, to simplify matters, each news report is represented using all the terms contained within it, and those terms are weighted using the TF.IDF measure. The use of knowledge bases, as suggested in [SC01], is avoided to adhere to surface-based methods.

In our opinion, the usage of the *Single Link* policy will tend to produce clusters that form ‘chains’ rather than actual clusters. A news report may contain information relevant to more than one topic. Therefore, by using the *Single Link* clustering policy, one can end up with unrelated documents within the same cluster. In our system, the similarity between a news report and a cluster is quantified by the cosine similarity between the term index of that news report and the cluster centroid index. The cluster centroid index is representative of the ‘average’ index of the documents within that cluster.

The use of an external document collection to calculate the term weights, as suggested in [Gul05] will reduce the dependence of the Inverted Index on the current state of the document collection. Since in our case, the collection of news reports is changing continuously, such an effect would be desirable. We apply a fairly similar approach — each document is represented using term weights which are
calculated based on the current state of the document (news report) collection. A global index is maintained which keeps track of the occurrence of each term over the entire collection, and this global inverted index is updated with the processing of each document. Meanwhile, a document index is constructed for each document which stores the occurrence frequency of each term within that document. Whenever similarity between documents needs to be calculated, the term weights are calculated on the fly — thus no re-calculation is performed for terms which do not appear in either document.

4 Methodology

In this section, we describe how we implement the approaches which we discussed in the previous section. Within our system the documents (news reports) are processed — i.e. indexed and categorized — one by one, and once a document has been categorized, the system moves on to the next document. It does not perform any cluster re-organization. The justification behind this is that the incoming stream of news reports (documents) for clustering never stops.

Within this phase, the documents are first tokenized and each term is then stemmed using the Porter Stemming Algorithm [Por97]. The stemmed versions of these terms, with the exception of stop words, are placed into an individual index for each document.

The terms within each document’s index are weighted using the $TF.IDF$ measure, whereby the $TF$ is taken to be the Term Occurrence frequency of the term within that document, and the $IDF$ is taken to be the Inverse Document Frequency of that term over the entire document collection in its current state. The Occurrence Frequencies of all the terms over the entire document collection are stored in a global index, and this global index is dynamic — whenever a new document has been presented for processing and has been indexed, the global index is updated with the terms from that document.

For the case where the system has just started processing its first documents, a special procedure is performed for the weighting of those documents’ terms. Before starting the categorization process, the system waits for the first 70 documents to be available for processing, and initializes a global index with the occurrence frequencies of these documents’ terms within this initial collection of 70 documents. Then whilst processing these 70 documents, the global index is not re-updated.

We chose to make the system wait for the first 70 documents to initialize the global index since 70 is approximately 40% of the entire set of document downloaded within the first 24 hours of the start of the system. We found out that when the system is started (i.e. there are no downloaded reports), the system downloads approximately 180 different news reports in the first day when using 4 different sources. In our opinion, 40% of the entire set of documents downloaded in the 24 hours from 4 different sources provide ample indication of which terms are important, and which terms are common throughout the entire collection.
After the document being processed has been indexed, it is clustered. The clustering is performed by calculating the cosine similarity between the document in question and the centroids of each existing cluster. Once a cluster is found to have a similarity higher than a pre-defined threshold with that document, the document is placed within that cluster. If no cluster is found to have a similarity which exceeds the similarity threshold, a new cluster is created with that document as its first member.

The cluster centroid is represented by an index of the stemmed versions of all the terms (excluding stop-words) which appear in those documents which are members of that cluster. The weight of each term within the cluster index is set to be the average weight of that term within the documents in that cluster. More specifically:

$$w_{t,c} = \frac{\sum_{d \in D}(w_{t,d})}{|D|}$$

where $w_{t,c}$ refers to the weight of term $t$ within cluster $c$, $D$ refers to the set of documents within the cluster, and $w_{t,d}$ refers to the weight of term $t$ within document $d$.

Since news reports are being continuously received and clustered, the amount of clusters is constantly growing. Since prior to classifying a document within a cluster on its own, it must be compared to all the existing clusters, as more news reports are clustered and new clusters are created, the clustering procedure will start to take longer. Therefore, a system where the clusters are continuously being created, but never removed, would not be scalable.

To resolve this issue, we utilized the concept of “freezing” old clusters. This means that clusters which have not had new members since a period of time are “frozen”, and incoming documents are not compared to them at all. They are assumed to be describing events whose “influence” has now passed and are not any more of “interest” within the world of news broadcasting. The identification of “frozen” clusters is performed by the system, which traverses the list of active (unfrozen) clusters every period of time.

## 5 Evaluation

For the evaluation of the Document Clustering System, we perform clustering on a set of corpora of news reports and then compare our results with how Google News\(^3\) clustered these same reports. In our opinion the clustering produced by Google News may be seen as the Gold Standard for news report clustering. From personal experience, the news reports clustered together in Google News are always related to each other — i.e. are always covering the same event. Therefore, we assume that the closer the clusters produced by our Document Clustering system are to the clusters produced by Google News, the more effective our Document Clustering system may be considered to be.

\(^3\) [http://news.google.com](http://news.google.com)
A single corpus is built by downloading the reports that appear within Google News in each of the news sections that appear on the Google News front page — namely World News, U.S. News, Business News, Science & Technology News, Sport News, Entertainment News and Health News. This is done by downloading the RSS feeds for each of the afore-mentioned section. Each record within these RSS feeds refers to a cluster of news reports and contains links to 4 or 5 reports, as well as a link to the Google News page which displays all the related news reports for that cluster. We downloaded those reports which are referred to directly within the RSS record.

Each corpus represents a ‘snap-shot’ of the Google News Clusters at a particular time. We built the multiple corpora by downloading the RSS feeds every hour and building a new corpus for each time the download is performed. We built 49 corpora in this way.

Since each downloaded news report is in HTML format (as it was displayed in its original web-site), we filter each report to remove the HTML code and surrounding text which are not part of the actual report text.

When attempting to download a report which was referenced in the Google News RSS feeds, there is no guarantee that the report is available for downloading. Reasons for such cases may be that the report is not available anymore, or that the news site providing that report requires a subscription to enable readers to access its reports. Therefore, when we download the reports, inevitably we download documents which instead of containing the actual news reports contain messages such as “This report is no longer available.”. To remove such documents from our corpora, we implemented a filter which calculates the average of the Inverse Document Frequencies of all the terms in the each document (excluding the stop words). Those documents whose average Inverse Document Frequency score falls below a particular threshold (in our case, this was set to 1.7), were removed from the corpora. We set this threshold to 1.7 after trial and error to see which value generates the best result — i.e. it removes as much “junk” as possible without removing good reports.

The evaluation of our Document Clustering procedure was performed separately for each of the news reports corpora. Each Google News cluster was compared to the most similar cluster produced by our Document Clustering system. Then, for each pair of such clusters the Recall, Precision and Fallout values were calculated.

To obtain a baseline measure, we implemented a clustering system which uses the standard K-Means method to cluster the reports. In this baseline system, the documents (news reports) within each corpus are indexed. The index terms are stemmed (using Porter’s stemming algorithm) and weighted using the TF.IDF measure. The IDF of each term is calculated relative to the term occurrence frequency within the entire corpus. The cluster centroids are represented by an ‘average’ inverted index, and the cluster centroids are updated only at the end of each pass. The clustering stops either when convergence has been reached (i.e. a pass has been performed which did not modify any cluster), or otherwise when 10000 passes have been made. The clusters produced by this baseline clustering system are evaluated in the same way we evaluated our clustering system.
The results obtained both by our clustering system and by the baseline system are presented in section 6.

6 Results

This section presents the results obtained for the evaluation described in the previous section (section 5. We calculated the Recall, Precision and Fallout values for each data corpus for our clustering system as well as for the baseline system. Figures 1, 2 and 3 show the recall, precision and fallout results obtained.

Fig. 1: Recall values

Our system obtained 89.21% average recall whilst the baseline system obtained 77.26%. This means that our system obtained 11.95% better recall. As regards precision, our system obtained 90.17% average precision whilst the baseline system obtained 61.49% — an improvement of 28.68% of precision from our system’s side. Our system obtained 1.02% fallout rate whilst the baseline system obtained 1.91% — a difference of 0.89%.

7 Conclusion

The results presented in section 6 show that our clustering system performs significantly better than the standard K-Means clustering system. With a couple of exceptions, our system obtained better recall and precision in all corpora, and had lower fallout as well. One has to bear in mind that the baseline K-Means clustering system was provided with the number of news clusters that should be created beforehand whereas our system did not possess and use this information.
Fig. 2: Precision values

Fig. 3: Fallout values
Nevertheless, our system produced better results. This shows that our system is able to detect the cluster set very well. Moreover, our clustering system is also more efficient than the standard K-Means system since our system does not perform any cluster re-organization, and each report is only processed once.

Besides producing better results than the baseline clustering system, our Document Clustering system also compares pretty well to the Google News clustering system which we consider to be the gold standard of news clustering. There is an instance — corpus 35 — where the recall is 1.00 for this entire corpus of data. This means that for this data corpus all the reports which should have been clustered together were in fact clustered together. There are also various cases where precision is 1.00 — corpora 22, 23, 36, 38, 43, 44 and 46. This means that for these corpora, our system produces clusters which are equivalent to, or are sub-sets of the clusters produced by Google News.

The results also show instances where our system did not perform so well — for example corpus 39 has 0.688 recall, corpus 1 has 0.725 precision. The main reason behind such instances is the presence of documents which do not contain an actual news report (due to erroneous download, or the report not being available anymore). The report filter (described in section 5) does not manage to remove all such reports. When processing such documents, our Document Clustering system places such document into clusters of their own. Obviously, Google News does not have the equivalent of such clusters.

Another reason for the occurrence of some low recall and precision values is that some news report documents contain more than 1 news item in them. For example, in the case of breaking news, a single document may contain 5 different news items where each item is covering an event totally different from the events covered by the other news items within that same document. In Google News, each news item is considered separately, and the same news document may be forming part of different clusters. Our Document Clustering system assumes that each document contains only one news item. Therefore it performs poorly when it encounters such documents.

The results obtained in our evaluation show that our News Report Clustering system produces news report clusters which are very similar to the clusters produced by Google News, and that it performs significantly better than a standard K-Means clustering system. When one considers that our system is able to work on a dynamic collection — i.e. it reads reports from a news stream and clusters them on the fly — it shows that our News Report Clustering system performs a satisfactory job, and it can be used as a reliable component in a News Web Portal similar to Google News, or as a part of a more complex system.

References

Automatic Clustering of News Reports


Abstract. The world of e-commerce presents ample opportunity to fully utilise the capability of intelligent Agents. The highly dynamic, fast-moving and information-rich environment can often be overwhelming for the human participant. Agents can intelligently assist users by mimicking human behaviour and adapting themselves to their client’s specification. This thesis presents an e-commerce framework that would introduce negotiation techniques which allows sellers and buyers to trade using Case Base Reasoning techniques as well as being proactive in remembering users’ requests and autonomously monitoring vendor sites for new items that might match the users’ needs and preferences. It observes the users whilst shopping and learns their preferences with respect to various features that characterise shopping items.

1 Background

This paper introduces a thesis with a vision of creating an e-commerce system which would facilitate the buying process in today’s competitive world. There currently exist a lot of price comparison sites which compare several products but however these are subject to the website having predefined vendors available. There has been substantial research in the area of shopping agents. BargainFinder was presented by Andersen Consulting as part of the SMART STORE initiative. It allowed users to compare prices of music CDs from several stores. Ringo was another example which later evolved to FireFly which was based on collaborative filtering. It suggested music basing its suggestions on what people with similar interest chose. The ShopBot was an agent that could learn how to submit queries to e-commerce sites and interpret the resulting items to identify lowest priced items.

The eBroker continues to evolve on these ideas with a vision of being more helpful. Using new technologies it is possible to build a framework for e-commerce comparison sites.

Semantic web gives the possibility for machines to read data within the site. Developing vendor sites using this markup would benefit the readability of the data it contains. This would allow agents to browse through the website without the need to build vendor specific modules.

Software agents assist users and will act on their behalf. An agent is often based on fixed pre-programmed rules and multiple agents can coexist together to achieve several goals.
Learning through experience is one of the main drivers to human know how. Case base reasoning is a proven area in artificial intelligence which uses past cases to devise a strategy for the future. This can be useful in being able to choose a negotiation strategy.

Inductive machine learning methods extract rules and patterns out of massive data sets. The system will learn user behaviour by observing user preferences whilst requiring no or minimal user feedback. The system will be give the ability to adapt to the users preferences and suggest the products the user is most likely to buy.

2 Introduction

The architecture of the system would be that of having multiple agents with diverse goals and capabilities to solve specific problems. Using an effective platform for coordination and cooperation amongst themselves would allow them to achieve one specific goal which would be to offer a client a service. The system would be made up of two external entities namely the client and the vendor. For each client, there are a number of lists for different items which are ordered according to the user preferences. The lists are gathered according to the interests the user has had in the past whilst also being able to suggest the most likely product the client would like to purchase. A buyer can specify the type of product, the number of lots, the maximum purchasing price, expiry date and time, etc. The vendor, through his website or application can specify the product, the number of lots, the minimum selling price, expiry date and time, etc. The seller group of agents residing on the vendor’s server, communicate product information, negotiation details, etc to the E-Broker group of agents residing on the E-Broker server.

The system maintains a database of outstanding requests from prospective buyers towards products various vendors are offering. The system would then enrol in a negotiation process between the buyer and the seller trying to arrive to a price that suits both parties. An agent (the buyer negotiator agent) makes an offer to another agent (the seller negotiator agent). If the seller accepts the offer the negotiation process ends, and the buyer can buy the product. If the seller does not accept the offer then the buyer can make a counter offer. The bidding proceeds in this fashion until an offer is accepted or a maximum number of offers have been rejected.

In such a brokerage system, AI techniques are applied to the short-listing and the negotiating stages and the amount of computation grows rapidly with a large number of clients and requests. The focus of research is therefore on efficient strategies and algorithms so that the system can respond to clients' requests within a short time. An e-commerce trading system is expected to complete a great number of transactions every day and the system has to respond in a matter of seconds. Thus the demand on response time is very critical.
3 System Architecture

Figure 1 illustrates the Privacy and the eBroker server. The privacy agent allows the user to take on a shopping persona and hides all identifying information. A shopping persona is unique identity for a particular user and its external details are independent to the owner’s identity, therefore the privacy of the user is kept intact. Each user can have many personae for different purposes, even outside the scope of this system. This will allow the eBroker system to learn a different profile for each persona.

![Diagram of Privacy Server, eBroker Server and Negotiation Server]

Fig. 1: Privacy Server, eBroker Server and Negotiation Server

The remaining module in figure 1 is the eBroker which is hosted on a separate server and is used for the user to interact with. The buyer agent takes user requests, displays results for searches and communicates with vendors for products. The learning agent takes user requests through the buyer agent and saves
them on a local database. It also monitors users’ actions and adjusts the profile accordingly. When a user submits a search through the Buyer agent the learning agent parses this request and returns results according to the profile learnt in the past.

The monitor agent periodically queries vendors for outstanding user requests. The negotiator agent acts as a client to the negotiation server where all the negotiations between the buyer and server will occur.

Figure 2 illustrates the Vendor Server which contains a semantic enabled website containing all the products the vendor is selling. The seller agent communicates with prospective clients and accepts requests for various products. Using the Browsing agent a list of products matching to the criteria are retrieved and returned to the buyer. The browsing agent is an agent that can traverse semantic websites searching for specific queries, which in this case are various products.

![Vendor Server and Negotiation Server](image)

The seller agent can also accept a negotiation request which is passed on to the negotiator agent which in turn communicates with the negotiation server. The negotiation will commence on the negotiation server.

The negotiation server is a module where all negotiations between the respective vendors and buyers will occur. The negotiation server holds a case repository which is used for case based reasoning. This consists of a repository of successful and unsuccessful transactions attached to a user profile. The case base repository
is then used to extract similar negotiations in the past and use that past knowledge to come to an accord. The user profile is saved in order for the case base reasoning filter to extract only cases that have occurred to that specific user. This creates different negotiation skills to different sellers and buyers, allowing vendors to be better than others, etc.

3.1 Learning Agent

One of the main features of the eBroker system is that the system can adapt to user preferences. The system gathers information whilst requesting minimal feedback from the user, therefore it studies user actions. For example if a user shows interest in expensive items then the system learns that this persona likes expensive items and starts giving more importance to expensive items. If the user ignores or removes an item from a list then the persona has shown that it is not interested in this type of item therefore showing that they dislike the features of this product. In time the system will learn the likes and dislikes of the user.

Inductive machine learning requires the extraction of features from the items in question. In this case the features which have been chosen in the current model are the price of the item, the number of bids that have been made towards this item, the time remaining for the auction and the similarity of the user query with the description of the item.

For each feature, e.g. price, a range of distribution temperatures are maintained across the range of values, e.g. low, medium and high. The temperature of a feature and value pair should indicate the users past desirable features.

The actions the user conducts maintain the temperatures for a feature value pair. The four possible reactions are to buy, browse, ignore and remove. Each reaction contains a predefined weight. For example buy contains a change in temperature weight of +0.5 since the reaction is considered to be a strong and positive one. When an item is removed a change in temperature weight of −0.5 is applied since it is considered a strong negative reaction.

\[
T(t + 1) = \alpha_1 T(t) + \alpha_2 \Delta T \quad \text{where } \alpha_1 = \alpha_2 = 1
\]

(1)

Items are then listed according to a simple sum of their temperature. User interactions then cause an update in temperatures and a re-ranking of the items based on updated temperatures.

3.2 Monitor Agent

The monitoring agent is a background agent which continuously queries vendors for any requests the user has which have not yet expired. The queries are done at intervals specified in the profile, and update the list of available items for the user. Updates could also include change in prices or product descriptions, whilst also adding new items that might have been added by the vendor. The next time the user will log on an amended list will appear.
This means that the system is autonomous by browsing the several vendors available for any products that the user might want then giving a result within a predefined timeframe.

### 3.3 Buyer Agent

The buyer agent acts in the name of the actual user. Its main tasks are that of coordinating all surrounding agents allowing them to work together and interface with any external systems which in this case would be the vendor and the privacy agent.

### 3.4 Negotiator Agent

The negotiator agent is in charge of communication with the negotiation module. This agent exists on both the vendor module and the eBroker module. When a user decides that a product can be bought then the negotiation agent is notified to start the negotiation process with the seller. In the meantime the buyer agent also notifies the seller agent that a negotiation process can commence which instructs the negotiation agent to commence negotiations through the negotiation server.

All negotiations between buyer and seller will occur on the negotiation server and when an accord on price has been reached the respective negotiation agents are notified together with any relevant information.

### 3.5 Privacy Agent

The privacy agent is used to hide the real identity of the user. It creates a persona which is used by the seller agent to communicate with other components. Using this agent none of the external components would know the real identity of the user.

### 3.6 Browsing Agent

As will be discussed further in this document the main aim behind semantic websites is that machines can read the information they contain. This agent would be used to browse semantic websites and search for specific queries the buyer might require. This would allow the development of one agent that could be used for all vendors having semantic websites rather then developing vendor specific modules.

### 3.7 Seller Agent

The seller agent acts in the name of the actual vendor and coordinates with the all surrounding agents. It is also used as an interface for the vendor module and is used for all communications with the seller.
3.8 Semantic Website

Humans are able to browse the web and extract useful information however an agent cannot achieve this due to the fact that web pages are built for humans to read. The semantic web provides a common standard called the Resource Description Framework (RDF) to publish relevant information in a machine readable form. RDF is a markup language used for describing information and resources on the web.

Using the semantic web as a requirement for our vendor sites will give agents the ability to browse through the website and extract data which on this case are products that will interest the buyer. This will remove the need to develop vendor specific modules to extract information from various websites.

3.9 Negotiation Module

A good negotiation skill in humans seems to come from experience, which comes to the reason why Case Base Reasoning was chosen. Past negotiation situations are used as strategies for future similar situations.

Figure 3 shows all the negotiation modules needed to develop the negotiation server.
- The External Interface would be the interface that the negotiator agents would be using to communicate with the negotiation server.
- The Database Interface would be the module with which the server can communicate with the negotiation case base repository database server.
- The Case Base Negotiator helps the clients to negotiate between each other. Each user will have its own set of cases or can be set to use all past cases. This module will also save any new negotiations to its repository.
- The Negotiation Case Browser allows user to browse past cases.
- The Descriptive Statistics gives useful statistics on the case base repository.
- The Case Maintenance allows negotiation experts to amend the case base repository and fine tune it.

Negotiation is an iterative process of offers and counter offers where a seller and a vendor would both have pre-defined goals and with this process their goals would converge to an agreement. Given an offer by a buyer, the negotiation engine will evaluate it to decide whether the offer is acceptable. If the seller decides the offer is not acceptable then the buyer has to decide a strategy for a counter offer. Case Base Reasoning techniques are used to extract strategies from previous similar negotiations to propose a new strategy. Similarity/matching filters are used to filter out the best matching past cases. Any new strategies are saved to increase the knowledge of the case base repository.

4 Conclusion

This paper introduced the eBroker system which would give shoppers the ability to buy products with as little human intervention as possible. The system would also learn human user preferences without any feedback from the user and also allowing the system to negotiate a desirable price whilst learning how to negotiate as time goes by.

The current market shows that e-commerce is rapidly growing and there is a need for such a system. The e-commerce market is so vast that products are available all over the place and there is no way that a user would have seen all products before coming to a decision. Currently there exist comparison websites which have predefined vendors programmed within, however using this framework, will give the user the ability to view all the products available within the query.

References


An Online Educational Portal for Teachers and Students in a Subject Department

Carmel Azzopardi and Matthew Montebello

Department of Artificial Intelligence, University of Malta

Abstract. Teachers are against duplication of efforts and dislike the sideline clerical duties that use up their resources for doing the proper job — enriching their knowledge and giving pastoral care for their students. The duty of marking attendance, entering marks and subsequently producing a global grade is an important link in the chain of school administration and hence a necessary evil. Today we are in an era enriched by useful technologies and tools that can help in the learning environment. In this paper, an already-existing open-source e-learning software platform is used and manipulated to accommodate the exigencies and specific needs of running a subject department at a post-secondary level in Malta.

1 Introduction

At the Giovanni Curmi Higher Secondary Naxxar, the Physics department caters for students who prepare for the MATSEC (Matriculation) certificate covered in two years and others who need to revise and re-sit for the SEC (Secondary) certificate in order to recover from a previous failure in the subject. The MATSEC certificate offers two levels in the subject, namely intermediate and advanced. Each intermediate and ordinary level Physics group is taught by one teacher while the advanced level group is taught by two teachers who run two different topics in parallel. While each teacher independently runs his/her course, the subject coordinator requires compiling the records given by both teachers per group to assign a final grade every term. Obtaining the relevant records from both teachers in time and using the traditional copy-and-write method is time-consuming, irritating, annoying and probably unwillingly having to chase the persons (who are equally professional) to adhere with stipulated deadlines dictated from the upper level — the administration. The responsible teachers restricted by school-time and their professional duties have to carry this not-so-considered high priority task using an irritating procedure of copying down what they have already done throughout the term.

Teaching as a profession should focus more on the student’s needs rather than losing time in transferring records manually from one spreadsheet to another. The continuous physical presence of the teacher in the school is not necessary as professional duties can also be carried out from any location if Web technologies and tools are used. Such tools allow the subject coordinator to retrieve the
required important information automatically without repeated intervention of these teachers. Students can communicate and access learning materials at any time and from any place as well as helping hindered students to make courage and engage in communication behind the barrier of the computer screen and the Internet. The system thus brings efficient collaboration between staff and friendly relationship between teachers and students providing a healthy and prospective learning environment.

Many e-learning software platforms exist which can partially satisfy the above-mentioned criteria as these are designed for university courses. In this environment each lecturer works independently in his/her course structure and thus the lecturer can build up the course, deposit the material, assess assignments and communicate with his/her students. As a result these software platforms allow courses to be designed independently and in isolation from one another. Some of these platforms are open-source and give the opportunity for the user to manipulate and shape the functionality of the system according to specific requirements. Our problem requires the merging of two independent courses run by two teachers in order to retrieve all the marks scored per student from both teachers and subsequently providing a final grade for the term. The project also includes some additional student information that is necessary but missing in the chosen e-learning platform software.

2 Physics department

Students taking advanced level course in physics are assigned in groups and will be having five lessons in theory plus two lessons in practical sessions every week. The five lessons are split into 3-lesson and 2-lesson blocks in both first and second year of their two-year course. In first year, three lessons for Mechanics and two lessons for Waves while in the second year there is Electricity and Heat for the same blocks respectively. Each block is delivered by a teacher who works independently of the other but will have to merge their records at the end of each term.

3 Moodle (Modular Object-oriented Dynamic Learning Environment)

In this study, Moodle is chosen as the learning environment framework to accomplish the requirements of the problem. Moodle is an open-source software framework founded by Martin Dougiamas [Dou07]. The first beta version was released in August 5, 2002 while the latest available version is Moodle 1.8.2 released on July 7, 2007. It runs on many platforms that support PHP (PHP: Hypertext Preprocessor) and stores data in an SQL database. The availability of its source code makes it possible to change, modify and add features into the existing framework in order to customise the product for the satisfaction of the user. Moodle is backed up by an informative website with active forums that help
the user to find solutions to the problems encountered during development and
an increasing number of plug-ins being developed by enthusiasts from around
the globe. Moodle offers a course management learning environment for online
classes offering

- organisation of independent courses,
- different types of assessment activity (offline, upload and inline assignments),
- grading facility for teachers,
- displaying records for students and teachers to consult,
- resources (like wikis, blogs, chats, forums etc) for an active participation and
  sharing of information between the participants and
- quizzes such as multiple choice, true/false, short answer questions and more.

4 Customising Moodle for our Institution

Students are registered to the course using an official identification number (I.D.)
and are assigned a group number. The department sorts students by groups and
similar names are identified by the I.D. number. Moodle provides an I.D. number
field but is only accessed from profiles menu. We require the student to enter
all relevant information as he/she creates a new account. The fields for I.D.
or registration number and group number are added. The page is also set to check for a unique I.D. number in which case the student is referred to the administrator or to re-enter the number correctly.

Teachers are assigned a course by the administrator and each has rights to master their part independently from one another. Lesson material, assignments and news are easily uploaded to the site for the student to watch and download. The coursework and tests are corrected and graded online. The students can see their individual grades while the teachers have the full-view for each group. In our situation the students must enroll to two courses that belong to two different teachers — First-year students for Mechanics and Waves while second-year students for Electricity and Heat. Each course is protected by an enrolment key supplied by the respective teacher. Subject coordinator is able to retrieve all the marks gained by each student in one spreadsheet file. Moodle exports the results in three file types — ODS, excel and text. The excel type is chosen for our project because one can easily produce statistics and charts in its application.

The combination of records from two separate courses is not available in Moodle but can only display record from one single course. The modification in the program is carried out in such a way as to leave the existing framework for the normal operation of individual courses but will be able to combine results and subsequently using the same existing channels to produce the output. Such operation will be restricted to the subject coordinator with administration rights and will be protected by an enrolment key. This operation of combining results is carried out from a course addition named ‘FIRST YEAR’ and another ‘SECOND YEAR’ constrained by the short names PHY01 and PHY02 respectively. Such codes are very important to be recognised by the conditional statements in the PHP code. Similarly the physics courses are also constrained by the short names
MECH and WAVES for the first-year courses and ELEC and HEAT for the second-year courses. The results can be exported to Excel format with the most relevant data entered by the student and his/her teachers as shown below. The learning environment may be expanded to cover other departments. For single courses, moodle requires no changes or alterations but for the combination of courses, the administrator will require the modification of the conditional statements already in place for the physics department and will have to be careful to keep to already-established short names.

5 Discussion and Related Work

In this age of advanced technology and sophisticated machines, the need for continuous education is accepted by the large majority. Education is not only attributed to the young age but is also attracting adults who are committed to their jobs and businesses but seeking to further their studies. This is the philosophy of lifelong education which is the key to survive and evolve in one’s own career. Cross [Cro06] argues that knowledge becomes obsolete in few years time since the pace of knowledge change is accelerating. In the past, one learned the method of doing something, thus required knowledge and skills while in the future one will have to learn what changed since last night and therefore requires innovation and ingenuity. In the not-so-distant past, distance learning used postage system to reach the committed adults to continue their studies at their own pace, in their own free time and in the luxury of their own homes. Nowadays the explosive growth of the Internet and the various existing and evolving technologies influenced the learning methods. Choy [Cho07] argues that e-learning (electronic learning) changed the medium of communication between the course organiser and the learner bringing the benefits of fastness, face-to-face participation and a huge repository of easily-accessed knowledge. He continues by quoting other authors that e-learning comprises the integration of three elements — content, technology and services. The technology should be limited to help learners while the teaching approach should be very important. Alsultanny [Als06] points out the benefits of e-learning in corporate environments mainly saving on costs in travelling and venues and gaining in more production time as the employees
do not have to leave their office. Attwell [Att07] stresses the usefulness of PLE (Personal Learning Environment) to future learning which helps the learner in developing an environment that suits his/her learning style. The concept utilises the use of social software (using computer network to visit services for accessing content and reaching other people).

Tweddell Levinsen [TL07] reminds us of the presence of two groups of people in ICT (Information and communication Technology) competencies. One group has acquired the skills in their lives while the other found the technology already in place before they were born. The author points out that both organisations and teachers are not ready for the change and adaptation of ICT. He insists that both sides can win this challenge through collaboration and knowledge-sharing. Tweddell Levinsen [TL07] states that although the use of ICT has grown among teachers, it is still mostly used for course organisation, administration and sharing of material. A change in culture is required for the new way of learning as a result of a looming information society. Learning must not remain a process of grasping and understanding content but a process showing the characteristics of adaptability and readiness to new problems by mastering the content independently.

Berners-Lee et al. [BLHL01] suggests an evolved type of Web (Semantic Web) having the capacity of processing information on behalf of the user making searches more relevant and useful. Alsultanny [Als06] lists several advantages of the Semantic Web to the e-learning context.

- Intelligent searching for content through the use of ontologies.
- Student has a personal agent cooperating with other agents for fast retrieval of material.
- Querying and navigation related to goals.
- Dynamic learning environment.
- Learning content created by the interaction of learners and educators.
- Individualised and personal learning.

Robberecht [Rob07] argues that computer-based learning materials offer linear (or sequential) approach where the learner is exposed to a sequential pedagogical experience with content displayed as one page after another. This type of model gives the same learning path for all learners except for the learning pace. The author recommends the non-linear design in which the learner determines his/her learning path on a personal and individual basis. This design incorporates interactive material containing context-sensitive and active elements to accommodate various levels and styles of learning. The difficulty of such design is the problem of teachers lacking programming skills while the technical people do not have the pedagogical expertise. The non-linear model accommodates any type of learner with a path and pace solely determined by the learner himself/herself. Information must be rationally linked to other information packets in such a way to provide a coherent learning experience that can be implemented by authoring tools (an application composed of linked objects like text, images or multimedia). Such tools help the educator to design active (based on doing and speaking) and passive (based on reading and seeing) learning.
Holger [Hol04] argues that information is doubling every two years and ironically its over-abundance is causing discomfort as we sift through the information to retrieve what we need. In the past, when information was stored in books and other printed material, a cataloguing system was used to lead to a shelf number and then the topic in the index of the book. The electronic information may be organised in a similar way by topic maps that provide the link between the domain of knowledge and information management. Topic maps are explicitly modelled ontologies capable of driving intelligent search engines and content management systems. Commercial publishers are interested in the technology to add value to their content, web portal providers will be able to organise their web sites with clear navigation patterns and in call centres they give support to the operator or directions to the client for the relevant answer to his/her query.

Olsevicova [Ols06] visualises the need of a virtual study environment based on Semantic Web and using topic maps as the solution to channel to all information and knowledge resources of the educational institution. The virtual study environment will apply topic maps technology in order to help students understand the structures of the course, teachers can recommend a study flow and all relevant parts of the course are unified together thus reducing the number of clicks required.

Evans and Bellett [EB06] discuss the outcome of a research carried out on two projects aimed at getting primary teachers to collaborate online for the development of teaching skills in two different subject areas — teaching maths to mixed-aged classes and developing teaching material for religious education. The comparison showed fruitful results for the project coordinated by an advisor with regular group sessions but lethargy proliferated among the participants of the other project which was conducted via a web-based discussion and detailed leaflets. The authors claim four necessary ingredients for successful establishment of e-learning communities namely face-to-face meetings; high-quality IT support; participants’ involvement and sharing of experiences; appropriate funding.

The invention of machines and robotics brought job cuts in the industry sector because the machines can do manual job quickly, reliably and for many hours without breaks. Will history repeat itself in the education environment? In a survey conducted by Wang et al. [WFSG03] found that e-learning did not replace traditional learning. They were studying employees in organisations who embarked on e-learning programs for their career advancement. The main reasons for quitting e-learning programmes were lack of motivation, learning style mismatch, same remaining workload and time dedication. Schmidt and Werner [SW07] claim that the main culprit is the human interaction factor, lack of discipline and motivation. As a result they propose blended learning which uses the advantages of face-to-face interaction and online instruction. Barrett et al. [BRM07] report a study on the effect of students’ attendance to classes in a blended learning environment that supplied course material for self-learning. The study showed no correlation between attendance and assessment gain. The authors concluded that the lack of attendance is an always-present phenomenon and commitments and external pressures were the true reasons for such be-
haviour. Although the Managed Learning Environment (MLE) compensated for skipped classes, both teachers and students appreciated the importance of the traditional way of learning.

Many educational institutions are experimenting with online e-learning environments to reach potential learners who are committed to their jobs but ready to further their studies and providing additional support to regular students who attend classes. The e-learning environments require the support of Web technology, communication facility, multimedia, assessment support and database tools. Various software products, proprietary and open-source, are available using popular Web browsers for the user front-end, server-hosting application and database tools. All the software platforms are intended to master the organisation of independent courses administered by one educator and provide collaboration between the teacher and his/her students. There is no facility for the administration of split courses running in parallel by different teachers to be finally merged together for the issuing of a global record at the end of the term.

6 Conclusion

In my sixteen years of teaching experience, Web technology has not yet infiltrated into the teaching profession. The younger generation of teachers use their personal computer for the organisation of content material, enriching their knowledge and keeping students records. This attitude is rather a substitute to the pen-and-paper method and not an appropriate exploitation of the technology. The time consumed and duplication of work for the administration of records is not reduced. Web technology gives the power to the user to access material from any location at any convenient time. Why do teachers have to remain immobile at their place of work? How much quality time do teachers really give in a noisy and distracted staffroom environment deprived of Web resources? Teachers need the premises for delivering lessons, meeting together for staff development and establish contacts with their students if enquired. The rest of the profession can be done very effectively from any place with the help of a virtual learning environment.

7 Future Work

We shall put the system online (using the department’s computer as a server) and test it for the coming scholastic year at the Giovanni Curmi Higher Secondary Naxxar for the physics department. In future, the benefits of the system may be sufficiently attractive to urge other subject departments in the same school to participate.

Each subject department in this school is expected to hand over to the school administration a formal assessment report carrying the following criteria: 50% for test/s, 40% for course work and 10% for attendance. There will be three reports for the first-year students and two reports for the second-year students. The second report for the first-year groups should not include the marks scored
in the first report while the final report will carry the average of the three formal assessment marks. The final report for the second-year groups is concentrated on the end-of-year test and formal assessment grade obtained in their first year and first term of their last year. We shall focus on the possibility of publishing this formal assessment in an Excel file according to the criteria.

This project may take a direction in exploring the possibilities of using further technology to implement the attendance record system. A Personal Digital Assistant (PDA) may be used by teachers in their class for recording the attendance and then conveniently sent to central administration for further processing. The attendance records will also be used for the formal assessments without further manual inputting. The PDA may also serve the teacher for inputting marks while correcting scripts and then transmitted to the department’s course management system.

References


Abstract. One of the problems with eLearning platforms when collating together documents from different resources is the retrieval of documents and their accessibility. By providing documents with additional metadata using Language Technologies one enables users to access information more effectively. In this paper we present an overview of the objectives and results achieved for the LT4eL Project, which aims at providing Language Technologies to eLearning platforms and to integrate semantic knowledge to facilitate the management, distribution and retrieval of the learning material.

1 Introduction

eLearning is the process of acquiring knowledge, information or skill through electronic means. One of the most popular gateways to eLearning is online via the Internet, often through Learning Management Systems (LMS). LMS allow tutors to manage collections of learning materials and monitor students’ progress, whilst providing students with a structured way to access data. However, given the huge amount of static and dynamic learning content created for eLearning tasks, it becomes necessary to improve the effectiveness of retrieval and the accessibility of such documents through the LMS.

Language Technology can support the evolution of eLearning, especially when used to enhance LMS. From a content perspective, it would be ideal if Learning Objects (LOs) would contain additional information to facilitate the retrieval of such documents. Content creators would want to emphasis their efforts on the learning task, rather than manually selecting and entering metadata. In the project Language Technologies for eLearning (LT4eL) we assist content creators by providing tools, such as a keyword extractor and a glossary candidate detector, to produce useful metadata which can be included within the LO.

Standard retrieval systems tend to offer keyword-based searching, matching words present only in the query term. Some more advanced search techniques might take synonyms into account, yet most techniques do not take into account systematic relationships between concepts denoted in the query term and other related concepts which might be relevant for the user. Ontologies are instrumental in expressing such relations and could result in better and more sophisticated ways to navigate through the learning objects. LT4eL has developed an ICT domain ontology that also allows multilingual retrieval of LOs.
The functionalities developed by LT4eL can be integrated in any open source LMS. For the purpose of validation within the LT4eL Project, ILIAS Learning Management System has been adopted.

The contribution of the project consists thus in the introduction of new functionalities which will enhance the adaptability and the personalization of the learning process through the software which mediates it. In particular, the system enables the construction of user-specific courses, by semantic querying for topics of interest through the ontology. Furthermore, the metadata and the ontology are the link between user needs and characteristics of the learning material; content can thus be personalized. In addition, the functionalities allow for retrieval of both static (introduced by the educator) and dynamic (learner contribution) content within the LMS and across different LMSs allowing for decentralization and for an effective co-operative content management.

LT4eL is a 6th Framework Programme Project, with the aim to facilitate the retrieval of learning objects through the use of Language Technologies and semantic knowledge. The consortium is made up of 12 European Universities representing 9 languages including English and Maltese for which the University of Malta is responsible. This paper presents an overview of the work that has been carried out so far, problems encountered and the results achieved for both English and Maltese.

2 Setting the Scene

The initial proposal submitted for LT4eL was to create new functionalities that will improve the eLearning process. Thus, our goals included putting together a corpus of learning material which could be used for the project. We also needed linguistic tools that would provide us with the required information to enable us to use Language Technology tools within the project. The main tools identified as being most important were a part-of-speech tagger and a morpho-syntactic analyser. Each language had to provide its own tools, and optionally could also make use of a noun phrase chunker and other linguistic tools.

The document collection for English proved to be a relatively easy task, with many IPR-free (Intellectual Property Rights) documents available for download from the Internet. The target for the corpus was of 200,000 words for each language in the areas of ICT and eLearning, with the final English corpus consisting of over 1.2 million words. The situation for Maltese was quite the opposite, where no ICT documents and a negligible amount of eLearning documents are available in Maltese. One can assume that the reason behind this is that the education and examination of many subjects taught in Malta is mainly held in English. Thus it stands to reason that the content is created in English rather than in Maltese. With no corpus available for Maltese, the following sections focused on the work carried out by the University of Malta on the English content.

The functionalities proposed by LT4el are described in Sections 3 and 4. The integration of these functionalities for the purpose of validation is described in

1 www.ilias.de
Section 5. Section 6 presents the possible ways for the Maltese language to be included in such a project, and what is being proposed to reach this end. Finally we present our conclusions in Section 7 with a view to possible future work outside of the project.

3 Language Technologies for Metadata Generation

The learning objects within the corpus tend to be written in a proprietary format, which does not allow easy manipulation and addition of metadata. In order to standardise the formats across the corpus all LOs were initially converted into HTML\textsuperscript{2} to retain layout information. Additionally we included linguistic information to enable the application of Language Technologies. A part-of-speech tagger, a lemmatizer and a morpho-syntactic analyser were used to provide documents with the necessary additional linguistic metadata. The linguistic metadata together with the HTML files were combined into an XML\textsuperscript{3}-based format conforming to the XCES\textsuperscript{4} DTD, a specification for linguistically annotated corpora [IS02].

Once the corpus has all the linguistic metadata available, we manually identified and annotated a set of 1000 keywords and 450 definitions within our corpora to assist in the creation and evaluation of the tools described in Sections 3.1 and 3.2. Through this schema, all information within our corpus becomes easily extractable and machine readable. Below is a sample of the final annotation, including the marking of keywords and definitions.

\begin{verbatim}
<\text{s id="s1501"> 
  \text{<definingText id="dt46" def="m281"> 
    \text{<markedTerm id="m281" dt="y"> 
      \text{<tok id="t20908" rend="b" ctag="NNP" base="datum" msd="N,SG,proper,vrbl"> 
        Metadata</tok> 
      \text{</markedTerm> 
      \text{<tok id="t20909" ctag="VBZ" base="be" msd="AUX,PRES,S,finite">is</tok> 
      \text{<tok id="t20910" ctag="VBN" base="define" msd="V,PAST,ED,finite"> 
        defined</tok> 
    \text{</definingText> 
  \text{\ldots}} 
</text>s>

3.1 Keyword Extraction

A keyword extract (described in [LD06]) was created as part of the deliverables of the project. The first task was to identify the characteristics of keywords, by analysing the manually annotated keywords for their linguistic features.

\textsuperscript{2} Hypertext Markup Language  
\textsuperscript{3} eXtensible Markup Language  
\textsuperscript{4} Corpus Encoding Standard using XML
Generally keyword extraction techniques consider only word count, with more sophisticated techniques taking into consideration the frequency of a word not only in the document, but also in the whole corpus. The keyword extractor was implemented with three different statistical techniques (Inverse Document Frequency and Residual Inverse Document Frequency described by [CG95], and Term Burstiness [Kat96, SGD05]). The keyword extractor was then further improved by taking into consideration the Part-Of-Speech (POS) tags. A language model was created to reflect what type of linguistic classes would fall under single-word keywords (such as nouns) or multi-word keywords at any positions (such as verbs). Additional weighting is also given to those words which have particular layout information, such as bold or italic, which was retained from the original file.

### 3.2 Definition Extraction

In the case of definition extraction, the approach taken by the project was for each language partner to identify possible linguistic patterns that could extract definitions. An XML transducer, `lxtransduce` [Tob05], was used to match the defined patterns and a rewrite rule is then applied to the matched cases. In our case, definitions are left intact, surrounded with `definingText` tags. The following is an example of a grammar rule which looks for a determiner at the beginning of a sentence followed by a noun:

```xml
<rule name="det_S_noun_phrase">
<seq>
  <query match="s/ *[1][name()=’tok’][@ctag=’DT’]"/>
  <ref name="noun_group" mult="*"/>
</seq>
</rule>
```

The set of 450 manually annotated definitions was split into three sets: (i) a training set; (ii) a testing set; and finally (iii) an evaluation set, each consisting of 150 definitions. The training set was used to extract possible patterns that can be commonly found in definitions. The rules were created through the manual observation of these sentence definitions, representing mainly the POS sequences noticed.

We observed that this task was a tedious one, and it was easy to overlook certain cases. A divide-and-conquer approach was adapted, and the definitions were split into categories. This reduced the complexity of the search space, whereby at each grammar identification attempt it is possible to focus only on one type of definition. The types of definitions observed in our texts have been classified as follows:

1. Definitions containing the verb “to be” as a connector.
   E.g.: ‘A joystick is a small lever (as in a car transmission gearshift) used mostly in computer games.’
2. Definitions containing other verbs as connectors such as “means”, “is defined”, “is called”.
   E.g.: ‘the ability to copy any text fragment and to move it as a solid object anywhere within a text, or to another text, usually referred to as cut-and-paste.’ In this case the term being defined is at the end of the sentence, and it is classified so by the use of ‘refer to’

3. Definitions containing punctuation features, usually separating the term being defined and the definition itself.
   E.g.: ‘hardware (the term applied to computers and all the connecting devices like scanners, modems, telephones, and satellites that are tools for information processing and communicating across the globe).’ where the definition is contained within brackets

4. Definitions containing particular layout style, similar to the punctuation feature, but separated through the use of a table (similar to the punctuation definition, however the term and definition would be placed in separate cells) or the defining term is a heading and the definition is the sentence below it.

5. Definitions containing a pronoun, usually referring to the defining term which would be placed outside the definitory context. This is common in cases where the definition is over more than one sentence, and the second sentence would refer to the defining term using a pronoun.
   E.g.: ‘This (Technology emulation) involves developing techniques for imitating obsolete systems on future generations of computers.’

6. Other definitions to capture those which do not fall in the above categories.
   E.g.: ‘information skills, i.e. their ability to collect and process the appropriate information properly in order to reach a preset goal.’ where the defining term and the definition are separated by ‘i.e.’

The above classification allows for the generalisation of rules to identify definitions in categories 1–5. However, the sixth categorisation does not facilitate the task of identifying a grammar for this category since it contains exceptional cases, and thus cannot be generalised.

**Machine Learning for Definition Extraction**

Through the categorisation of definitions, we were able to improve results for certain categories, such as the is-a category. However, having achieved a high recall, precision was considerably low. This meant that whilst good definitions were being captured, a high number of incorrect definitions were also being included in the result set.

An other problem was that there was no ranking of the results as the extraction method used was a simple yes-no classification. This meant that definitions were presented to the user in no particular order. Since the system was intended to suggest definitions to a content creator for approval, having a few incorrect definitions was not deemed as a problem. However, it was desirable that the definitions are presented in a ranked order, so that those definitions with a higher confidence value are presented at the top of the results. We also observed that incorrectly classified definitions could be filtered out using post-processing filtering after the initial grammar was applied.
To tackle these tasks, a Machine Learning (ML) group within the project was created to research on possible ML techniques and to improve these results. The University of Malta is part of this group and its approach to this task is further reported in [Bor07,BRP07].

4 Enhancing Learning Objects with Semantic Knowledge and Multilinguality

Semantic knowledge provides additional useful information that can be utilised for enhancing of document retrieval. There are two types of users that we consider: (i) educators and content creators compiling a course from existing resources, and (ii) learners searching for content to suit their current needs. LT4eL aims to improve the retrieval of LOs with the use of ontologies, which will be integrated within the LMS to structure, query and navigate the LOs.

An ontology constitutes a formal representation of concepts (classes), and the relations (properties) between them. There are different approaches to ontology design. In our case we looked at a layered design, where generic concepts are represented in an upper-level ontology and more specific concepts are represented in a domain ontology. Through this approach we are able to re-use existing upper-level ontologies, and concentrate more on creating a domain ontology (described in [MLS07]). An analysis of upper-level domain ontologies was carried out to identify which would suit our requirements best. We concluded that DOLCE [MBG02] (Descriptive Ontology for Linguistic and Cognitive Engineering) suited best our requirements. DOLCE was built using formal ontological analysis and formal semantics, approached using ontological engineering practices. It is also modular and has an open license. These factors influenced our bias towards selecting DOLCE as the upper domain ontology.

In designing a domain ontology, we followed the strategy as specified in [Gua00]:

1. lexicon (vocabulary with natural language definitions);
2. taxonomy;
3. thesaurus (taxonomy plus related terms);
4. relational model (unconstrained use of arbitrary relations);
5. fully axiomatized theory.

We started by constructing a terminological dictionary, which contains the term in English, a short definition describing the term, and the translations of the term in the represented languages within the project. Then we formalised these terms by including basic ontological relations (is–a, part–of, used–for) which are inferred from the upper ontology. These will be translated in OWL–DL. Within the scope of the project, we aim to achieve a full relational model of the domain ontology. By connecting the ontology domain to the upper-level ontology, we will ensure inheritance of the axiomatization of the upper ontology to the concepts in the domain ontology.

The annotation of LOs with semantic knowledge can be based either on a concept or on several concepts and their relations. The latter is referred to as an ontology
chunk. We envisage this to be more relevant to our task as it will allow (i) a more detailed search without consulting the ontology, (ii) represents the relevant information in the context of the LO, and (iii) it facilitates generic ontology searches by allowing navigation over the ontology.

The availability of a multilingual lexicon allows for retrieval of documents in several languages. For instance, a user can search using a Polish term and request to view both Polish and English documents where that concept is present. This is particularly useful in a LMS were learners would want to see all relevant information possible. In future, we intend to provide the facility for a content creator to annotate the content of an LO with ontology chunks. Thus the user will be able to employ a mechanism to select a chunk on the basis of the concepts and relations between them, and to include that chunk as part of a LO’s metadata to facilitate the navigation over the ontology with respect to the content of the LO.

5 Integrating and Validating eLearning Scenarios

The functionalities described above had to be integrated to an open source LMS. For purposes of validation, ILIAS was chosen to develop an interface between the functionalities and the LMS. ILIAS offers typical LMS features such as creating, editing and publishing of materials, collaboration and communication tools, course management, test and assessment tools and user administration. It also includes basic LOM\textsuperscript{5} support, but lacks, as is the case for other LMSs, advanced techniques for more efficient metadata handling and learning object retrieval. In order to make the functionalities available for any LMS, a web service based architecture was used to integrate these functionalities.

The basis for the integration of the functionalities within the LMS is constituted by the use cases. They show how the behaviour of the LMS changes through the use of the developed functionalities, especially how existing features of the system are improved and how new features have been made possible through their use. Examples of relevant use cases are:

- author annotates semi-automatically learning objects with keywords;
- author generates semi-automatically glossaries for learning objects;
- learner searches for learning objects.

The use cases also provide a starting point for the definition of validation scenarios in the validation phase of the project. eLearning applications are very much an emerging field, and there are no standard, general methodologies that can be used to validate the effectiveness of the learning process in our specific context. A suitable validation methodology is being developed which will be applied to the validation of the new functionalities as well as to their integrated set into ILIAS.

\textsuperscript{5} Learning Object Metadata
Our validation process will be centred on the development of a number of User Scenarios, which focus on the role of teachers and learners. User Scenarios are defined as ‘a story focused on a user or group of users, which provides information on the nature of the users, the goals they wish to achieve and the context in which the activities will take place’ [ET05]. They are written in ordinary language, and are therefore understandable to various stakeholders, including users. They may also contain different degrees of detail. In the context of the LT4eL project, scenarios being developed currently focus on course and content creators, teachers and students. The scenarios will be constructed to take the following four dimensions into account in order to evaluate the success of the project:

- the usability of the platform itself, and in what way it is affected by the integration of the new functionalities;
- the pedagogical impact of integrating the functionalities;
- the consequences of incorporating multilinguality;
- the social impact on virtual learner communities — and crucially, how this is affected by multilinguality.

The scenarios are still very much in their infancy and it is expected that they will be considerably enriched as the development of the functionalities progresses. The resulting dialog between evaluators and developers will help to establish the possibilities for future use and subsequent scenario development and may also influence the development process.

6 Maltese in Context of LT4eL

The inclusion of Maltese within LT4eL was challenging from its inception. We were aware of the lack of resources in the domain of ICT. Yet running in parallel with this project, the Maltese Lexicon Resource Server [RFAG06] aimed to create a corpus for Maltese, and to set out the framework for language tools to be created. Still the domain of ICT remained short of documents, and thus we were forced to limit the participation of Maltese in LT4eL.

We proposed alternative solutions which had to be discarded as they were deemed either unpractical or unrealistic for the scope of the project. Amongst these:

- to translate existing documents in ICT from English to Maltese — this was discarded because it would have been expensive, and the end result would be a translation — something that we wanted to avoid within the project.
- to use an ICT related corpus in Maltese that is not aimed for eLearning — here we suggested to use documents from the European Parliament website[^6] which discussed ICT in general. This idea was also discarded as the corpus would cover to varied a domain from the one LT4eL had achieved (e.g. LOs on Word Processors).

Finally, we opted for the translation of the lexical dictionary described in Section 4. This translation task still remains a challenging one, since many ICT terms are not yet defined in Maltese. Once completed these terms will be integrated into the ontology. This will enable search for LOs in Maltese, and then retrieving English LOs. Of course, this is far from the ideal scenario. However, this discussion is beyond the scope of this paper.

7 Conclusion

The project LT4eL brings together existing research knowledge and tools in different areas and puts their use and application in an innovative way. Keyword-based retrieval of documents is used extensively in many search applications. However, apart from taking the normal statistic approach, we are also including linguistic knowledge to improve the results. We also propose both quantitative and qualitative approaches to the validation of the keyword extractor. In definition extraction, we not only look at grammar patterns that could form definitions, but we will also apply machine learning to this area to see the improvement of extraction. Semantic knowledge will give us increased meta-data that will enable improved document retrieval and navigation through the ontology. All these functionalities will be applied to a LMS to improve the eLearning experience. We believe that the most innovative result of LT4eL will be the crosslingual retrieval of learning objects with the help of language independent ontology and language specification lexicons.

References


Towards Automatic Extraction of Definitions

Claudia Borg¹, Michael Rosner¹, and Gordon J. Pace²

¹ Department of Artificial Intelligence, University of Malta
² Department of Computer Science, University of Malta
{claudia.borg|mike.rosner|gordon.pace}@um.edu.mt

Abstract. Definition extraction can be useful for the creation of glossaries and in question answering systems. It is a tedious task to extract such sentences manually, and thus an automatic system is desirable. In this work we review various attempts at rule-based approaches reported in the literature and discuss their results. We also propose a novel experiment involving the use of genetic programming and genetic algorithms, aimed at assisting the discovery of grammar rules which can be used for the task of definition extraction.

1 Introduction

A definition is a term together with a description of its meaning or the concept it refers to. Definitions are helpful because they facilitate the understanding of new terms. The extraction of definitions from text can be useful in various scenarios, including the automatic creation of glossaries for the building of dictionaries and in question answering systems. In this work, we will focus on the use of definition extraction in eLearning, where definitions can help learners conceptualise new terms and help towards the understanding of new concepts encountered in learning material.

eLearning is the process of acquiring knowledge through electronic aids, by providing access to materials that will enable them to learn a particular task. A tutor can direct the learning process through a Learning Management System (LMS), where learning material is presented to the student according to the tutor’s direction. The tutor can also use the LMS to add new content, create courses, structure layout and presentation of courses and monitor student performance.

Learning material, packaged into units known as learning objects (LOs), normally contain implicit information in natural language, which would require a lot of work for the tutor to extract manually. An LMS can be enhanced by introducing tools to extract such information automatically. One such piece of information is the presence of definitions in texts. We propose a tool that will attempt to extract definitions from LOs, which an LMS could use to extract definitions (for instance, to be used to create a glossary). The tutor will then be able to refine this information rather than create it from scratch.

The task of definition extraction is a challenging one. We are trying to identify sentences that contain knowledge which could then be used by applications such
as those mentioned above. What more, we are attempting to identify sentences which define a term, rather than simply describe it vaguely, or compare it to other terms.

Since definitions are made up of natural language texts, we propose to use linguistic knowledge such as part-of-speech tagging and morphological analysis to support the definition extraction. Furthermore, it was noticed that there exist different syntactic forms of definitions. Hence, rather than trying to identify arbitrary definitions, we propose to look at the different definition categories separately.

We propose an experiment in Section 2 which combines genetic algorithms (GA) and genetic programming (GP) to try and discover grammar rules that could identify definitions present in LOs. Section 3 will review the work related to our task, divided into two parts. First we look at different approaches in definition extraction using rule-based techniques. Then we will look at the area of grammatical inference and the application of GAs and GPs to learn grammars. The outcome of this work is to evaluate the use of machine learning techniques (GAs and GPs) and their results in learning restricted grammars. The grammars developed through these experiments can then be applied by rule-based techniques to extract definitions. The results of the GP and the GA will be used to discover features which identify certain definitions with a high rate of accuracy, but also other features to classify less clearcut definitions using features in a combined manner.

This work is done in collaboration with an EU-funded FP6 project LT4eL. The project is described in more detail in [BR07] and [MLS07], and aims at enhancing LMSs by using language technologies and semantic knowledge.

2 Proposed Approach

In eLearning, LOs are generally created by tutors in different formats such as HTML, PDF and other text formats. A corpus of LOs, gathered within the LT4eL project, has been converted to one standard XML format with added linguistic information. Work carried out in the project has also manually identified and annotated a set of 450 definitions from this corpus.

Given that a corpus contains both a set of definitions and a (usually larger) set of non-definitions, an attempt to learn the importance of features present in definitions is possible. A feature can be seen as a description of characteristics that can help us identify a definition. To simplify the identification process, definitions have been split into six different categories (as described in [BR07]). By learning to identify the categories separately, we reduce the size and complexity of the search space.

---

3 Language Technologies for Learning www.lt4el.eu
2.1 Experiment One: Genetic Algorithm

A genetic algorithm (GA) [Hol75,Gol89] is a possible technique that can be used to learn the importance of the features that can recognise definitions. This can be done by assigning weights to each feature and allowing the algorithm to adjust the weights according to the performance. It also makes it ideal to run the GA on the separate categories of definitions identified (as described in [BR07]), so that the results can be directed to one given situation at a time.

A feature is a function which given a sentence (which includes linguistic information) returns a numeric score. An example of a feature would be that of a part-of-speech sequence that may capture a definition — returning a numeric value indicating how close the given sentence matches that particular sequence (e.g. DT→NN→VBZ→DT→NN→IN→NNS). An other example of a feature is a test whether the sentence contains the verb ‘to be’ — with the only possible values of the score now being 1 or 0, indicating the presence or otherwise of the verb. A feature is said to be an effective classifier of definitions if it gives higher scores to sentences which define a term than to other sentences.

Given a set of \( n \) basic features, \( f_1 \) to \( f_n \), and \( n \) numeric constants, \( \alpha_1 \) to \( \alpha_n \), one can produce a new feature combining these basic features in a linear fashion:

\[
F(s) = \sum_{i=1}^{n} \alpha_i \times f_i(s)
\]

The problem is now: given a fixed set of features, how can we calculate a good set of weights so as to maximise the effectiveness of the combined features as a definition classifier? We propose to use a GA to learn these weights. The values learnt would thus correspond to the relative effectiveness of the individual features as classifiers of definitions. Before starting the experiment, a predefined set of features will be adopted, which will remain static throughout the experiment. A gene will be a list of length equal to the number of predefined features of numbers. Thus, the \( i \)th gene (\( 1 \leq i \leq \text{populationsize} \)) will have the following structure:

\[
g_i = (\alpha_{i,1}, \alpha_{i,2} \ldots \alpha_{i,n})
\]

Note that \( n \) corresponds to the number of predefined features. The interpretation of the gene is thus a list of weights which will be applied to the output of the predefined features. For instance, \( \alpha_{i,1} \) is the weight that the gene \( g_i \) would assign to the first feature. Such a gene will therefore return a score to a given sentence \( s \) as follows:

\[
score_i(s) = \sum_{j=1}^{n} f_j(s) \times \alpha_{i,j}
\]

The initial population will consist of genes that contain random weights assigned to each feature. The interpretation of the gene is a function that when applied
to a sentence gives the summation of the feature-function scores multiplied by their weights.
The fitness function will take an individual and produce a score according to its performance. The score will be calculated by applying the gene to both positive and negative examples and will be judged according to how the gene is able to separate the two sets of data from each other.
Crossover and mutation will be carried out in the traditional way of GA. Crossover will take two individuals, split them at a random position and create two new children by interchanging the parts of the parents. Mutation will take a random position in the gene and change its value. If the children perform better than the parents, they will replace them.
Once the population converges, the weights of the best individual would give the clearest separating threshold between definitions and non-definitions. This will also allow us to identify which of the features in the predefined feature set are most important.

2.2 Experiment Two: Genetic Programming

Genetic Programming (GP) [Koz92] is a technique that uses GA principles to evolve simple programs. The main difference between GAs and GPs is the representation of the population and how the operations of crossover and mutation are carried out. The members of the population are parse trees, usually of computer programs, whose fitness is determined by execution. Crossover and mutation are carried out on subtrees, ensuring that the resulting tree would still be of the correct structure.

Whereas the scope of the previous experiment was to learn which are the best performing features for the task of definition extraction from a set of predefined ones, this experiment aims at identifying new features. The choice of what type of structure we are trying to learn determines the complexity of the search space. In our application, two possible options could be regular languages (in the form of regular expressions) or context-free languages (in the form of context-free grammars), the latter having a larger search space than the former. Through observations and current work with lextransduce [Tob05], regular expressions (extended with a number of constructs) should be sufficient to produce expressions that would, in most cases, correctly identify definitions.

Both basic and combined features used in the GA can serve to inject an initial population into the GP. The selection can be made based on the weights learned by the GA and translating those features into extended regular expressions. The extensions that are being considered are conjunction (possibly only at the top level due to complexity issues), negation and operators such as contains sub-expression. Note that some of these can already be expressed as regular expressions, however, introducing them as a new single operator helps the genetic program learn faster.

The population will evolve with the support a fitness function in order to select those individuals for mating. The fitness function can apply the extended regular expressions on the given training set and then use measurements such as precision
and recall over the captured definitions. Such measurements can indicate the performance of the individuals and will allow us to fine-tune the GP according to the focus of the experiment (where one could emphasis on a high percentage for one measurement at a time, or take an average for both). This flexibility will also allow us to have different results in the various runs of the experiment, where, for instance, in one we could try to learn over-approximations whereas in another we can learn an under-approximation.

Crossover takes two trees and creates two new children by exchanging nodes with a similar structure. If an offspring is able to parse correctly one definition, it survives into the next generation, otherwise it is discarded. Parents would normally also be retained in the population, since we would not want to lose the good individuals (it is not obvious that their offspring would have the same capability of correctly identifying definitions).

Mutation takes an individual and selects at random one node. If that node is a leaf, it is randomly replaced by a terminal symbol. If it is an internal node, it is randomly replaced by one of the allowed operators. Once again, the new tree is allowed to survive to the next generation only if it is able to capture at least one definition.

Once the GP converges, we expect to have new expressions that would capture some aspects of a definition. The application of this program will allow us to extend our current set of grammar rules by deriving new rules from the above operations. Although we do not expect the GP to learn rules, it will help towards the discovery of new rules which might have been overlooked, and thus helping towards a more complete grammar for definition extraction.

The GP will also allow the flexibility of running this experiment separately for each of the categories of the definitions as identified in section 2.2. This means that the new features being learned will be restricted to one category at a time.

### 2.3 Combining the Two Experiments

The role of this work is to develop techniques to extract definitions. The two experiments are independent of each other. The GA takes a set of features and assigns a weight to each feature, whereas the GP learns new features through the evolution of the population of extended regular expressions. We can combine the two experiments by migrating the new features learned by the GP to augment the feature set which is used in the GA.

In the final definition extractor one can start by checking whether a given sentence can be confidently classified as a definition by using the features learnt by the GP, possibly giving a preference to over- or under-approximations. One would then run the weighted sum and threshold as learned by the GA based on the features we manually identified and others that the GP may have learned. Clearly the training of the GA would have to be done on a subset of the training set, removing the confidently classified non/definitions. We believe that this approach will improve the quality of the definition identifier.
3 Literature Review

We split this review into two main parts, starting with an overview of published results using rule-based definition extraction, followed by work in grammatical inference which applies GAs and GPs. A final discussion comparing the proposed work to the work reviewed, then follows.

3.1 Rule-Based Definition Extraction

Work carried out on automatic creation of glossaries usually tends to be rule-based, taking into consideration mainly part-of-speech as the main linguistic feature. Park et al. [PBB02] propose a system whereby glossary candidates are presented to an expert in the relevant domain to be approved and made available through an API. In their work, they concentrate on detecting the terms and their glosses rather than full definitions. Glosses present a summary of the meaning, usually giving the full form of an abbreviation or a variant of the term. They also deal only with technical texts, where glosses are normally well structured. They propose a pipeline architecture using several tools, including POS tagging and morphological analysis, with each tool providing additional annotations. The glossary extraction algorithm first looks at the possible linguistic structure of glossary items found technical texts. They identify POS structures for noun phrases or verbs, which are used to identify possible glosses. These are described as a cascade of finite-state transducers, which are easy to extend and re-use even across different languages. An observation they make is that is difficult to identify glosses by simply looking at POS sequences, since many other non-gloss items would have the same sequence. To overcome this problem, rules are applied to discard certain forms from the candidate set. These include person and place names, special tokens such as URLs, words containing symbols (except for hyphens and dashes) and candidates having more than six words. Variants are identified, grouped and one is set at the canonical form, others listed as variants (including misspelt items and abbreviations). Finally all glossary candidates are ranked and presented to the expert. In the evaluation of their work, three human experts accepted 228 (76%), 217 (72%), and 203 (68%) out of the top 300 as valid glossary items. The evaluation does not consider missed definition which ranked lower. Inter-annotater agreement is not discussed in this evaluation.

Klavans and Muresan [KM00] propose a system, Definder, to extract definitions from technical, medical texts. The corpora used comprise of consumer-oriented texts, were a medical term is explained in general language words in order to provide a paraphrase. The aim for their system is to be able to extract definitions that can then be fed into a dictionary. Their approach uses NLP techniques to identify definitions and synonyms (which are also considered as definitions in this context). They point out that the structure of definitions might not follow
the genus et differentia model and that the different styles of writing can be a challenge for the automatic identification of definitions. Definder first identifies candidate phrases by looking for cue-phrases such as “is called a”, “is the term for”, “is defined as”, or a set of punctuation marks which are deemed important for this task (namely :, (, ), , ). A finite state grammar is then applied to extract the definitions. The system uses part-of-speech and noun phrase chunking to help with the identification process. In order to improve results, the Definder uses statistical information from a grammar analysis module. The authors claim that doing so takes into account the styles for writing of definitions (apposition, relative clauses, anaphora). In this work we see that the automatic identification of definitions is mainly based on the primary identification of certain phrases, and then further filtered through certain rules that reinforce a sentence being a definition (such as its POS structure).

Klavans et al. [KPP03] look at the Internet as a corpus, focusing mainly on large government websites, trying to identify definitions by genus to extract conceptual relations for ontology building. In this task, several problems are identified, including the format of the definitions and the content in which they are present. Definitions on the Internet can be ambiguous, uncertain or incomplete. They are also being derived from heterogeneous document sources. Another problem encountered is that the Internet is a dynamic corpus, and websites could change their information over time. An interesting discussion is presented in how to evaluate a definition extractor, proposing a Gold Standard for such a type of evaluation, based on qualitative evaluation apart from the standard quantitative metrics.

Liu et al. [LCN03] are interested in definition extraction of concepts for learning purposes. Their strategy to assist learning is to present learners with definitions of concepts, and the sub-topics or salient concepts of the original topic. Their system queries search engines with a concept, and the top 100 ranked results are retrieved. In order to discover salient and sub-topics, they look at layout information presented in the html tags for features such as headings, bold and italic. A rule-based approach is then applied to filter out items which are generally also highlighted in webpages (such as company names, URLs, lengthy descriptions). Further filtering is applied through stopword removal, taking frequency into consideration and ranking the proposed salient or sub-topics. Definition extraction is then attempted for the concepts and sub-concepts identified in the first phase of their work. The identification is carried out through rule-based patterns, (e.g. {concept} {refers to | satisfies} . . . ). Webpages containing definitions are attached to the concept, and presented to the user in ranked order. The ranking is based on how many concepts/definitions are present in a webpage (the more being available, the higher the ranking as it is considered more informative).

Liu et al. also propose a way of dealing with ambiguity, where the term being learned is too generic and may appear in different context (e.g. classification may

---

4 A genus et differentia definition first describes the term explain the broader concept, the genus, and then distinguishes from other items in the category by differentia
be used in library classification, product classification, data mining classification, etc.). Again, to differentiate between different contexts, the term is allocated a parent topic, and through the use of document layout structure a hierarchical structure of topics is built accordingly. Mutual reinforcement is also used to provide further evidence of the hierarchy built, by further searching for sub-topics under a particular concept. This generally would result in finding information about other salient topics under that same concept, which thus continues to re-ensure that the sub-topics are related and belong to the parent concept.

It is interesting to note that the definition extraction phase is preceded by a phase of concept identification, simplifying the task of definition extraction. However, the search for definitions is carried out on particular concepts, and not all definitions contained in a given text. This might result in definitions of other concepts present in parsed documents being lost. The work also does not make any use of linguistic information, since at their level of processing simple pattern matching on particular keywords is sufficient. The web is a very large corpus and thus can provide many documents containing definitions. However, this can also be a disadvantage affecting the quality of definitions found, similar to the problems encountered by [KPP03] described above (i.e. ambiguous, uncertain or incomplete definitions). This is partially surpassed by providing several resulting definitions, and documents containing more definitions (which might be more authoritative) are presented first. However, quality of definitions is important in any learning phase, and can determine the learner’s understanding of a concept.

Storrer and Wellinghoff [SW06] report work on definition classification based on 19 primary verbs, specified in valency frames. These frames indicate what arguments a verb takes, such as object, subject, position and prepositions. This frame can be used to match the structure of a sentence containing one of the specified 19 verbs. Thus definitions are extracted by using the valency frames specified for the defining verbs. The approach presented in the paper is a rule-based expert driven, with all information being provided by human experts (valency frame, definition categorisation). This is possible because they are looking at technical texts, where definitions are well-structured, frequently matching more crisp rules.

Fahmi and Bouma [Fah06] tackle the problem of definition extraction using an incremental approach, starting with individual words, then adding syntactic features etc. They look at the potential definition sentences that fall into our first category (containing the verb to be) from a Dutch corpus of medical articles extracted from Wikipedia. These sentences are manually annotated as definitions, non-definitions and undecided, and this corpus of sentences is used as their training and evaluation data for the experiments carried out. They identify several attributes that could be of importance to the experiments, namely text properties, sentence position, syntactic properties and named entity classes. Learning-based methods are then used to identify which of these features, or combination of, would provide the best results. These feature combinations can also be considered for the experiments described above.
3.2 Grammatical Inference Using GAs and GPs

Identifying grammars for definition extraction is closely related to grammatical inference — the use of machine learning techniques to learn grammars from data. GAs, and less frequently GPs, are two such techniques which have been applied to grammatical inference.

Genetic Algorithms

Lankhorst [Lan94] describes a genetic algorithm used to infer context-free grammars from positive and negative examples of a language. The schema theorem states that schemas occurring in the higher scoring individuals will tend to occur more frequently in following generations. This feature, also referred to as the building blocks hypothesis, is the motivation for applying GAs to grammatical inference. New, possibly better performing, grammar rules may be discovered by combining parts of different grammar rules.

A discussion is presented on the choice of gene representation, between a binary representation and a high-level representation. It is argued that a bit string can represent many more schemata than higher level representations, yielding to a better coverage of the search space. A bit representation is chosen, with the lower order bits encoding grammar rules on the right hand side of the rule, whereas higher order bits are encoding the left-hand side symbol.

Selection is based on a stochastic universal sampling algorithm, that helps to prevent premature convergence by ‘holding back’ super individuals from taking over the population within a few generations. The best individual is also always allowed to survive to the next generation. Mutation allows for a bit in a chromosome to be mutated. However, this operation is given a low probability so as not to change the population randomly. Reproduction is effected by the schema chosen. The crossover point is influenced by how the representation of the rule is expressed. Lankhorst choses a two-point crossover system, thus allowing right-hand side rules to crossover more easily.

The grammars that Lankhorst is learning with the GA are aimed to classify positive and negative examples over a language correctly. Thus the fitness function that is considered takes into consideration the correct classification of positive and negative examples. For various sets of languages, such as matching bracketing and ‘0*1*0*1*’, the fitness function allows the population to converge into a solution within reasonable results. However, for a micro–NL language, the fitness function did not result in a correct grammar. A further adjustment to the fitness function also allows correct classifications of substrings to influence the fitness score. This modified fitness function allows the GA for a micro–NL grammar to converge correctly.

This work provides an interesting insight to different techniques of how the fitness of an individual should be calculated. Different fitness functions will output different results and it is important to explore alternatives. In our case we would like to take into consideration which measures should be given more importance to (e.g. recall, precision or F-measure). The representation of an individual is
also important, Lankhorst having selected a bit–representation over a higher-
level representation such as trees. Tree representations can be easily applied
with Genetic Programming, and yet, exactly the same principles/discussions
apply with respect to the fitness function.

Losee [Los96] applies a GA to learn the syntactic rules and tags with a direct
application in document retrieval. The system parses just over 100 abstracts, all
about the same general topic, subdivided into 5 different subtopics. The task of
the system is to retrieve the documents in the ideal ranking order according to the
search term used. The GA is applied to learn syntactic rules and tags, and thus
provides linguistic meaning to both documents and terms. As a fitness function,
the GA uses a weighted function of the resulting ranking of the document and
the average maximum parse length.

Belz and Eskikaya [BE98] attempt grammatical induction from positive data
sets in the field of phonotactics. The grammar is represented with finite state
automata, and looks at the use of GA for this task. In this paper two results
are produced, one for German syllables and the other for Russian bisyllabic
words. The GA is described in detail, including the type of methods selected, and
chromosome representation. The GA used is a fine-grained one, where individuals
are on a 2–D grid of fixed size and are only allowed to mate with one of their
neighbours (this implementation is referred to as a torus).

They argue that an important issue is the representation used for individuals.
They present two alternatives: (1) production rules of the form $s_1 \rightarrow as_2$ and
$s_1 \rightarrow a$ (where $s$ is a non–terminal symbol and $a$ is a terminal symbol), or (2)
a state transition matrix. They argue that production rules produce more fine
grained genotype representations since the terminals and non–terminals can be
represented individually. On the other hand, state transition matrices can be
only seen as a whole, each represented by a single cell. The final representation
chosen for this work is that of transition matrices. Each individual represents a
possible transition matrix which in turn represents the grammar being induced.
In order for genetic algorithms to be used, the matrix is ‘flattened’ into one
string (one row after the other).

The chosen representation has direct implications on the rest of the GA oper-
ations. Crossover and mutation cannot be carried out in the traditional sense
of GAs, and certain knowledge must be present in the GA so as to maintain a
sound structure of this flat matrix. Belz and Eskikaya seem not to have consid-
ered GPs and tree representation for their problem. Such consideration could
have provided an interesting alternative in their work.

Keller and Lutz [KL97] attempt learning Context Free Grammars through the
use of GAs, by learning probabilities to all possible grammar rules. The initial
population of the GA is made of all possible combinations of terminals and non–
terminals of the form $A \rightarrow BC$ and $A \rightarrow a$, where $A,B$ and $C$ are non–terminals
and $a$ is a terminal. This guarantees that, although the grammar is a large one,
it is finite. There is also no loss of generality, as all possible rules are present in
the initial grammar (including those that will not be part of the final solution).
The type of GA chosen for this work also uses a 2-D grid representation in torus formation, where mating occurs only with immediate neighbours. Each individual is encoded as a set of weights, each weight relevant to one parameter. The weight is represented as an n-bit block, and an individual can be viewed as consisting of $M$ blocks of n-bits, where $M$ is the number of all possible rules. Since the grammar is considerably larger than the final solution, Keller and Lutz try to give more importance to 0-probability assignment. Thus, the initial bit/s of the n-bit block is seen as a “binary-switch” as to whether the rest of the bits should be taken into consideration or not.

As for crossover, Keller and Lutz achieved better performance by using a novel genetic operator which they call and–or crossover than by the classical crossover operation. The and–or crossover looks at the parents bit by bit, with one child taking the bits produced by the and operator (conservative), and the other child takes the bits produced by the or operator (liberal). In our proposed work, this idea of the bit-by-bit crossover can be adapted slightly to use functions such as minimum, average and maximum to develop new children.

Spasić et al. [SNA04] aim at classifying biomedical terms into specific classes, which represent concepts from an ontology in the biomedicine domain. In order to derive the possible class of a term, they look at the surrounding context of that term. This context is learned through data mining, extracting the contextual patterns surrounding the terms. The patterns contain morph-syntactic and terminological information and are represented as generalised regular expressions. Each contextual pattern is then given a value indicating its statistical relevance. They use this to remove the top and bottom ranked features since they are considered too general or too rare to play a role in term classification. Class selection of a term is then learned using a Genetic Algorithm. For a particular class, the GA tries to learn which of the contextual patterns are relevant. Each individual in the GA is a subset of contextual patterns and its fitness corresponds to the precision and recall of using these patterns on the training data. Eventually the GA learns a good subset of features which can be used to identify terms in that class.

**Genetic Programming**

Smith and Witten [SW95] propose a GP that adapts a population of hypothesis grammars towards a more effective model of language structure. They discuss grammatical inference using statistical methods, and the problems encountered in their work. They point out that probabilistic n-gram models allow frequent, well-formed expressions to statistically overwhelm infrequent ungrammatical expressions. There is also the problem with allowing probability for unseen data. The ‘zero-frequency problem’ entails assigning a small probability to all unseen data, resulting in both ungrammatical n-grams becoming as probable as unseen grammatical ones.

The population is represented as LISP AND–OR S–expressions. Initial experiments showed that certain constraints were required in order for the GP to evolve. These constraints included a maximum depth for nesting and a grammar-
Towards Automatic Extraction of Definitions

generator to allow the GP to evolve towards more suitable grammars. With these constraints in place, the GP evolves simple grammars even within 2 generations, forming simple sentences such as ‘the dog saw a cat’. However, the GP is left to run over more generations to achieve a broader exploration of the search space and hopefully result in a more efficient grammar. Although the results seem positive, there is no comparison to other statistical techniques mentioned that attempt grammatical inference.

3.3 Discussion

Work carried out in definition extraction shows that although it is possible to achieve a good basis for a grammar through manual observation, this task requires specialized linguistic understanding of grammatical features present in definitions. Ideally, a filtering or ranking mechanism is used to over and above these techniques to further improve the results. Work by Fahmi and Bouma [Fah06] moves towards this direction. In our proposed experiments we introduce the concept of ranking through the use of the GA, learning weights assigned to grammars indicating the certainty of whether the captured sentences are actually definitions.

The GP, and in particular the gene representation, is a crucial point which comes out clearly in the work reviewed. There is a lack of knowledge in the area of GPs, and this is visible in the various attempts of describing a grammar as a linear structure to work with GAs rather than taking advantage of a GP’s capability to handle tree representation. Smith and Witten [SW95] overcome representation issues by using grammars as LISP S–expressions. In our work, the resulting grammar that will be produced will be used within lxtransduce [Tob05]. Thus any type of representation chosen will be translated to the XML format accepted by lxtransduce.

It is also clear that the fitness function will determine the success of the experiments. Since we have to our availability manually annotated definitions, precision and recall could be used as part of the fitness function. However, different tests could be carried out to determine what should comprise the fitness evaluation of the population.

4 Conclusion

Attempts at definition extraction have focused mainly on rule-based approaches, with some later work improving results by introducing statistical analysis as a filtering step. Our proposal introduces an element of grammar learning for the set of definitional sentences, influenced by the work carried out in grammatical inference. We will also use the weights produced by the GA as part of the filtering and ranking process, as evidence to what should be classified as a definition. The proposal takes a novel approach in combining GAs and GPs for natural language processing. A quantitative evaluation of these techniques will compare
the results achieved to the work carried out so far in definition extraction. The results should be of interest not only to the natural language task of extracting definitions, but also to the machine learning task of combining GAs with GPs.

References


Mi-Learn: An evaluation of an m-learning management system

Ernest Cachia¹, Jonathan C. Borg², Christopher L. Spiteri², and Mark Vella¹

¹ Department of Computer Science, University of Malta
{ernest.cachia|mvel0022}@um.edu.mt
² Department of Manufacturing Engineering, University of Malta
jonathan.borg@um.edu.mt, chspit@eng.um.edu.mt

Abstract. M-Learning is a novel concept concerned with delivering learning content over mobile devices, which today is being seen as a way to support for conventional and collaborative learning as well as for extending the scope of e-learning. This paper presents the work carried out on a research project named Mi-Learn, whose goal is that of gathering background knowledge within the field of m-learning, identification of related research problems, and creating an environment where solutions to these problems can be identified and evaluated. Pocket SCORM initiatives such as [ADL04] that are working towards a set of standards for m-learning have identified the restricted user interface and the requirement for offline learning sessions as the two major challenges in this area. The work presented in this paper forms part of the first phase of an m-learning research project and concentrates on the first of these challenges. By means of a pilot m-learning management system, an evaluation exercise was carried out in order to assess the impact of a restricted user interface on the learning experience. The evaluation carried out takes both the pedagogical and user interface aspects as the evaluation criteria. Evaluation results show that currently there seems to be a value for m-learning, but more as an extension for exiting e-learning programmes rather than a complete learning management system on its own. The results also helped in clarifying the research area and setting a direction for further research work.

1 Background

The Mi-Learn project is a research project [BVP03] funded by the University of Malta with an overall objective of building a mobile learning infrastructure for the departments of Computer Science and Manufacturing Engineering at the University of Malta. This would allow students to use their mobile devices (mobile phones, personal digital assistants (PDAs), etc.) to follow courses (or part thereof) online. The project’s goal is that of gathering background knowledge within the field of m-learning, identification of related research problems, and creating an environment where solutions to these problems can be identified and evaluated. This paper presents the results from the first phase of the project. In this phase a pilot m-learning environment was developed, with the
aim of analysing its potential value as well as the limitations that such a learning environment may provide.

1.1 M-Learning

A literature review was conducted on the following research areas: Mobile Devices, Mobile Learning Systems, EU Mobile learning projects and the adaptation of the Shareable Content Object Reference Model (SCORM) in m–learning. The main points emerging from this study are the following:

- **User interface adaptation** — Mobile device screens such as the ones found in PDAs to the ones found in smart phones are relatively much smaller compared to screens available in bulkier devices such as personal computers (PCs) and laptops. This reduced screen size in mobile devices creates limitations in creating mobile device user interfaces. Mobile presentation technologies such as the Wireless Markup Language (WML)\(^3\) and later the Compact Hypertext Markup Language (cHTML)\(^4\) address this limitation. User Interface restriction is one of the two main challenges presented in Pocket SCORM proposals such as [ADL04]. Pocket SCORM is a project attempting to adapt the SCORM standard to mobile devices.

- **Memory** — Whilst mobile devices are commonly associated with memory restrictions, the latest generations of mobile devices such as phone enabled PDAs are changing the situation mainly via SD chip technology.

- **Bandwidth** — Whilst bandwidth available for mobile devices has been known to be low and costly (e.g. over GPRS [BMV04]), extended WI-FI coverage [BVP03] [OdM03] and the increasing popularity of PDA devices [DMAM04] are improving the situation.

- **Asynchronous communication** — Bandwidth limitations are favouring asynchronous communication within the field of mobile devices, where due to lack of WI-FI coverage or GPRS costs, mobile device user sessions are carried out off-line with the device being required to be on-line only at the stages where application server synchronization is required. Asynchronous communication is the second main challenge identified by Pocket SCORM within the field of mobile learning.

- **M-Learning as a conventional learning support** — Mobile devices are being used as a support for conventional learning. An example of this is the slideshow exposure system proposed in [WFP05], where the integration of a presentation software, such as Microsoft\(^\circledast\) PowerPoint, with handheld devices for use in a teaching environment that allows users to selectively download and annotate parts of the slide show using mobile devices. A web application has been developed for use in a lecture theater or classroom environment that allows students to access content from a slide show during a presentation. The website content is extracted into a markup language,

\(^{3}\) [http://www.wapforum.org](http://www.wapforum.org)

and an HTML version of a PowerPoint slide show is made available via a wireless access point (Figure 1). Another example of mobile devices assisting conventional learning can be found in the iSign [MFA04] project. The iSign project started as a web-based laboratory setting for students of electrical engineering. It has expanded into a heterogeneous learning environment offering learning material, adaptive user settings and access to a simulation tool that can be accessed via the web and also by wireless clients, such as PCs, PDAs and mobile phones (Figure 2).

![Fig. 1: Slideshow](image)

**M-Learning as a collaborative learning support** — Mobile devices are also being used as a means of enhancing collaborative learning. One such example is mCLT [AMT04]. The mCLT system is a Javatm Mobile Information Device Profile (MIDP) client for mobile telephones. It introduces an innovative mobile platform for computer-supported collaborative learning, based on 3rd Generation mobile telephones. Students can collect and share live data immediately, anywhere and anytime. This enables them to play an active role in the knowledge-building process.

**M-Learning as an adaptation of e-learning systems for mobile devices** — The most common m-learning applications could be found identified in this category. Learning Management Systems (LMSs) found in this category are primarily e-learning systems that are either adapted or extended to leverage the use of mobile phone devices. An application in this category is the Intelligent Web Teacher
Fig. 2: iSign Modules [MFA04]

(IWT). This project is an extension of a state-of-the-art e-learning system —
the Intelligent Web Teacher (IWT) — to support multimodal mobile access
[CGMP04]. This work offers a complete set of learning experiences, services and models that are well suited for the complex mobile world. The extended platform offers customized e-learning experiences depending on the type and capabilities of the users’ mobile device. Subsets of IWT functionalities can be accessed from HTML-enabled mobile devices, where the content and layout are automatically adapted by the IWT engine that recognizes the type of device used.

1.2 Research Direction

In the light of the literature review’s main points, a research decision was taken in the direction of evaluating an m-learning experience vis--vis the user interface restrictions posed by mobile devices, as the primary task. This decision was made since this aspect could be considered as a fundamental one affecting all others. In fact, from the results derived from the evaluation of an m-learning experience, it would be possible to identify those learning tasks that could be feasible to carry out via mobile devices in order to extend existing e-learning systems, and those which would not be feasible. In addition, positive results from such an evaluation would give value to research carried out on the area of asynchronous communication in relation to m-learning. With this research direction taken, the deliverables of the first stage of the project were set as follows:

- An m-learning LMS allowing the evaluation of an m-learning experience in respect of a restricted user interface.
- Two sample m-learning courses on which the evaluation is based.
- The results of the evaluation, with a focus on which aspects of a learning experience are well suited for m-learning, and which do not.
2 Mi-Learn LMS — A mobile Learning Management System (LMS)

The Mi-Learn LMS is a pilot application built with the focus of enabling the evaluation of an m-learning experience and as such sophisticated features such as asynchronous communication or the automated adaptation of learning content for use with mobile device profiling was not included in this first phase of research. The LMS can be accessed by any mobile device having a cHTML client and supporting a GPRS or WI-FI connection. The Mi-Learn URL at the time of writing of this paper is http://milearn.cs.um.edu.mt. Figures 3, 4a and 5a display a sample set of screen shots from the mi-learn m-learning portal. Figure 3 shows the main page, figure 4a shows the course list screen, whilst figure 5a shows a sample course content screen. Figure 4b and 5b show the equivalent of the latter screens, in case no mobile device screen adaptation is performed. The difference in the screen layout where no adaptation is carried out can be clearly noted, whilst the other difference that cannot be shown here is the difference in the download time required.

Fig. 3: Mi-Learn portal
2.1 Architecture

Figure 6 shows the architecture of the mi-learn portal in UML Deployment Diagram notation [OMG04]. The main part of the architecture can be found within the Mi-Learn node. The components deployed on this node are:

- a web server (httpd);
- a PHP application server;
- an SQL database (MySQL); and
- the Mi-Learn LMS that compromises of a customized Moodle (http://moodle.org) component.

A repository of SCORM compliant course contents and course images are also deployed on the Mi-Learn node.

The customizations carried out on the Moodle LMS were mainly concerned with compacting the user interface for restricted screen size presentation purposes. Although the cHTML versions of the pages are immediately adapted for mobile device viewing, some Moodle pages were considered to be too cluttered even after conversion to cHTML. The customization activities are as follows:

- Removal of the breadcrumb menu from header.html residing in the chosen theme directory;
- Addition of the Mi-Learn logo in the header.html in the chosen theme directory;
- Addition of the Google\textregistered logo in the footer.html in the chosen theme directory;
- Update of the /course/player.php file in order to display the table of contents at the bottom of the page rather than at the top. Keeping the contents at the top can give rise to cumbersome user navigation due to cHTML page segmentation;
- Update of the /course/player.php file in order to add a further SCORM navigation bar at the bottom of content screens in order to enhance ease of navigation;
- Update of the /lib/weblib.php file in order to remove recurring login message;
- Update of the /login/index_form.html file in order to remove the recurring “Returning to website” and “New user?” header messages;
- Update of the /lang/en_utf8/moodle.php file by changing “Available courses” string to “Mobile courses” string;
- Removal of the recurring Moodle image from the print\_footer() function in weblib.php;
- Removal of the recurring Moodle documents link from footer.html found in the chosen theme directory;
- Removal of the top horizontal break from header.html found in the chosen theme directory.

2.2 Learning Content

The learning content used for the evaluation sessions was adapted from two courses originally created for deployment on an e-learning LMS. The two courses are: Computer Integrated Manufacturing (CIM) and Linux Operating System Principles. Both courses feature the use of diverse media (text, images, sound, and video) and course components (content, assignments, assessment sessions, etc.). The process of adapting these courses for delivery to mobile devices itself resulted in the identification of the necessary adaptation requirements needed to convert an e-learning course content to an m-learning one. The m-learning
experience evaluation could highlight further adaptation requirements. The idea here is that a way for automating the course content adaptation to m-learning environments is researched.

Fig. 4: (a) Mi-Learn course list, and (b) Course list (with no UI adaptation)

This way, course content would need only to be created once, and applied both in e-learning as well as m-learning environments. The main result of the adaptation process was found to be concerning with non-text media. Images and video could be too large in size to transfer, too cumbersome to view, or have detail lost, when viewed on small screens. Sound media could not be adequate in case when a mobile device is used in a quiet environment and no headphones are available. Flash formatted presentations, a popular format for multimedia presentations are not supported as yet by mobile phone devices.

Some devices though do support the Flash-Lite format, which is a sub-set of the full flash format. Still, a flash presentation cannot be converted to Flash-Lite, but rather created from scratch specifically for the lite version. In all of these cases the solution was to have a text equivalent of all the non-text media utilized, in order not to deprive m-learning students from being able to follow certain course contents. A further necessary adaptation, purely from a technical point of view was that images embedded within a SCORM content package are no longer visible once the cHTML version is created by the Google proxy. This technical limitation requires that images forming part of a course content package are uploaded separately in an appropriate web server location. This in turn requires that SCORM HTML files no longer refer to these images as local course resources, but reference the appropriate images repository URL.

http://www.adobe.com/products/flash
Fig. 5: (a) Mi-Learn course content, and (b) Course content (with no UI adaptation)

Fig. 6: Mi-Learn portal architecture in UML Deployment Diagram notation
2.3 Evaluation Procedure

The evaluation procedure consists of three parts:

– Virtual Learning Environment (VLE) evaluation;
– User Interface evaluation;
– Critical evaluation.

The VLE evaluation is concerned with evaluating the pedagogical aspects of the Mi-Learn portal and is based on the framework for Pedagogical Evaluation of eLearning Environments by Britain and Liber [BL04]. The User Interface evaluation is concerned specifically with the m-learning portal user interface in regards of the limited screen sized and is based on scientific research carried out by Jeffries et al. [JMWU91]. The critical evaluation exercise is concerned with the research process itself — critically appraising the research process so far and setting a research direction for the future.

3 Virtual Learning Environment (VLE) Evaluation

The VLE evaluation is concerned with evaluating the pedagogical aspects of the Mi-Learn portal and is based on the framework for Pedagogical Evaluation of eLearning Environments by Britain and Liber [BL04]. This evaluation framework provides a means by which the pedagogical process underlying VLEs can be reasoned about, and then how a particular VLE under test can be evaluated based on how the VLE encourages or less these pedagogical aspects. Although originally built with e-learning systems in mind, the framework can be utilized with any form of a Virtual Learning Environment due to the fact that it abstracts from the particular user interface being adopted. The framework is built upon two fundamental principles. The first principle is that of evaluating the incorporation of effective teaching and learning practice into a VLE. The second principle assesses the organizational aspect of the VLE, that influences whether the system will facilitate or less the ease with which a pedagogical can be used within that system.

3.1 VLE Evaluation Results

The VLE evaluation exercise was carried out by the research and development team, since an in-depth knowledge of the virtual learning system is required in order to carry out such an evaluation. Being mainly a customisation of Moodle, the starting point for evaluating Mi-Learn’s virtual learning environment was the evaluation of Moodle carried out in [BL04]. The main differences between Moodle and Mi-Learn were identified and the evaluation carried out accordingly.
Mi-Learn: An evaluation of an m-learning management system

The Module Level — Overall results (Table 1)

Presentation and re-presentation of key concepts and ideas — Mi-Learn allows several tools for tutors and student to express teaching ideas and supplying feedback. These tools, as provided by the underlying Moodle infrastructure are: Resources (content), forums, journals, quizzes, assignments, surveys, chat and workshops. All of these tools are usable both by students and tutors except for resources, which can be used only by tutors. The VLE experience as seen by tutors is exactly the same as the one experience in Moodle, since tutors access Mi-Learn from the same identical Moodle portal. Students access Mi-Learn via the mobile device portal, still having access to the listed tools, but through the restricted mobile device portal.

Coordination of people, resources and activities — The model of teaching and learning interactions was retained as originally provided by Moodle. The system encourages modules to be laid out in a sequential order. A module outline is required. Learners can be organised in a whole group, separate subgroups, visible subgroups or individuals (group of one). Types of learning activities include: Connected discussions with optional peer evaluation, Reflective journals, Reading, Glossary/Encyclopaedia writing (Students can build up a glossary and any of those entries automatically link from any text throughout the system), Chatting, Peer-evaluated assignments and Quizzing.

Resource negotiation and agreement — The ‘rules of the module’ are expressed and made evident to the student in the same way as it is carried out in Moodle. Essentially this is left to the teacher to express, using the tools provided (e.g. setting an introductory activity containing the instructions to be followed throughout the particular course).

Monitoring and Learning — The facilities required to monitor the learning progress within the context of a module are available for tutors (activity reports and grade books) but not available directly for students. An integrated SMS gateway that sends assessment results to students having phone enabled mobile devices, would be a welcome addition.

Self organization amongst learners — Outside the purview of the teacher, learners are able to both upload files as well as to locate other people, as per Moodle. Student file upload is possible via forums and glossaries, but of course restricted by the capabilities of the mobile device in use. Locating people is possible via a people search allowing the localisation of any visible Mi-Learn user.

Adaptability of module and system — Like in Moodle, module structure can be adapted and assigned to particular groups of students.

The Student Level — Overall Results (Table 2)

Coordination of people and activities — The concept of programme-level progression is not directly supported by Mi-Learn.

Resource negotiation and agreement — As per Moodle, Mi-Learn does not allow for the specification of programme rules for delivering a module nor does it permit or provide a space for negotiation between programme managers and module tutors on resource questions.
Table 1: Module level evaluation

<table>
<thead>
<tr>
<th>Evaluation Point</th>
<th>Evaluation Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Presentation and re-presentation of key concepts and ideas</td>
<td>*****</td>
</tr>
<tr>
<td>Coordination of people, resources and activities</td>
<td>*****</td>
</tr>
<tr>
<td>Resource negotiation and agreement</td>
<td>*****</td>
</tr>
<tr>
<td>Monitoring and Learning</td>
<td>****</td>
</tr>
<tr>
<td>Self organization amongst learners</td>
<td>*****</td>
</tr>
<tr>
<td>Adaptability of module and system</td>
<td>*****</td>
</tr>
</tbody>
</table>

Monitoring of modules — The performance of a module can be monitored by the programme manager by means of logging as an administrator, an invisible teacher or an invisible guest. QA examination or peer observation of module activities is not included.

Self organization of teachers — As in Moodle, self-organization of teacher is possible by making a “course” for teachers (who attend as students), where they can coordinate and assist each other.

Adaptability of programme — At a content level, apart from the ability to import SCORM content, courses can be designed and developed. Courses can also be hidden from student allowing development to take place before being “made live”. No direct support for validation is present.

Table 2: Student level evaluation results

<table>
<thead>
<tr>
<th>Evaluation Point</th>
<th>Evaluation Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learner-centeredness</td>
<td>*****</td>
</tr>
<tr>
<td>Time management / planning</td>
<td>*****</td>
</tr>
<tr>
<td>Monitoring own learning</td>
<td>*****</td>
</tr>
<tr>
<td>Adaptation / reflection</td>
<td>*****</td>
</tr>
</tbody>
</table>

The Programme Level — Overall Results (Table 3)

Coordination of people and activities — The concept of programme-level progression is not directly supported by Mi-Learn.

Resource negotiation and agreement — As per Moodle, Mi-Learn does not allow for the specification of programme rules for delivering a module nor does it permit or provide a space for negotiation between programme managers and module tutors on resource questions.
Monitoring of modules — The performance of a module can be monitored by the programme manager by means of logging as an administrator, an invisible teacher or an invisible guest. QA examination or peer observation of module activities is not included.

Self organization of teachers — As in Moodle, self-organization of teacher is possible by making a “course” for teachers (who attend as students), where they can coordinate and assist each other.

Adaptability of programme — At a content level, apart from the ability to import SCORM content, courses can be designed and developed. Courses can also be hidden from student allowing development to take place before being “made live”. No direct support for validation is present.

<table>
<thead>
<tr>
<th>Evaluation Point</th>
<th>Evaluation Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coordination of people and activities</td>
<td>*****</td>
</tr>
<tr>
<td>Resource negotiation and agreement</td>
<td>*****</td>
</tr>
<tr>
<td>Monitoring of modules</td>
<td>*****</td>
</tr>
<tr>
<td>Self organization of teachers</td>
<td>*****</td>
</tr>
<tr>
<td>Adaptability of programme</td>
<td>*****</td>
</tr>
</tbody>
</table>

Table 3: Programme level evaluation results

The VLE evaluation of Mi-Learn shows that in terms of a virtual learning environment, Mi-Learn is strong on the module level, but lacks both on the student and programme levels. The strong evaluation results on the module level are mainly due to the amount of learning activities than can be deployed and adapted in the learning environment, as well as the way in which learners enrolled in a module can be grouped. Yet, self-monitoring and organization were found to be very limited in this respect. Although the individuality of students is given prominence throughout the m-learning system, Mi-Learn was found to be weakest in respect of the student level. This result is mainly given due to the lack of Personal Development Planning (PDP), time management tools, and the inability of monitoring one’s progress. The evaluation results for the programme level aspects of Mi-Learn are only average. Whilst showing a strong implementation of course design, tutor co-ordination and overall module progress monitoring, the concept of a programme of courses is missing.

4 User Interface (UI) Evaluation

The User Interface (UI) evaluation is concerned specifically with the m-learning portal user interface in regards of the limited screen sized and is based on scientific research carried out by Jeffries et al. [JMWU91]. This first phase of the research concentrated on the point that was considered to be the most crucial in
moving from an e-learning to an m-learning environment — the restricted user interface. In [JMWU91] four user interface evaluation techniques are presented and compared, these being: Heuristic Evaluation, Usability Testing, Guidelines and Cognitive Walkthroughs. In Heuristic Evaluation, UI specialists study the interface in depth and look for properties that they know, from experience, will lead to usability problems. In Usability Testing, the interface is studied under real-world or controlled conditions, with evaluators gathering data on problems that arise during its use. The Guidelines technique involves in the publishing of user interface qualities to be observed by developers during development of interfaces. In Cognitive Walkthrough the developers of an interface walk through the interface in the context of core tasks a typical user would need to accomplish, with the actions and feedback of the interface being compare with the user's goals, with any discrepancies arising being noted. According to the comparison study carried out in [JMWU91], the Heuristic Evaluation and Usability Testing guarantee the best results mainly due to the best rate identifies in identifying serious and recurring problems. In this case the choice was made for the Usability Testing approach for the reason of being able to get first hand feedback from potential users of such a system — in this case the students themselves.

4.1 UI Evaluation Results

<table>
<thead>
<tr>
<th>User Interface Issue</th>
<th>Reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Textual presentation focus</td>
<td>48%</td>
</tr>
<tr>
<td>Continuous navigation required</td>
<td>30%</td>
</tr>
<tr>
<td>Low quality technical images</td>
<td>13%</td>
</tr>
<tr>
<td>Cumbersome collaboration tools</td>
<td>9%</td>
</tr>
</tbody>
</table>

The usability testing exercise for the Mi-Learn LMS was carried out by a sample of potential end users at the Computer Science and Manufacturing Engineering Departments. Two full courses were setup on Mi-Learn, and students asked to complete one of these, whilst in the meantime pointing out all those issues that in their view undermined the overall learning experience. Table 4 shows the overall UI evaluation results. All the reported issues could be grouped in one of the following four m-learning related UI issue categories:

- Textual presentation focus (48%)
- Continuous navigation required (30%);
- Low quality technical images (13%);
- Cumbersome collaboration tools (9%).
The problems within the ‘textual presentation focus’ category are concerned with the fact that there is too much textual content in courses, all of which is unjustified. Also related to the issue is the lack of animation in the learning content. Long sections of text make the learning content less interesting, and also difficult to follow on a small screen. Due to cHTML limitations, text justification is not as yet possible, as well the use of varied fonts is limited, making it even harder on the learner to read the text. The lack of animations make course content less interesting. The reported problems within the ‘continuous navigation’ category are concerned with the facts that learning on a small screen is strenuous and hinders learning experience and the continuous clicking and scrolling through learning content hinder learning experience. Essentially a full course is too bulky to follow on a mobile device screen and it becomes tiresome after a while. Being able to see only a small portion of the screen makes learners feel lost at times and hinders learning. The low quality of technical images stems from the fact that when images are resized to fit mobile device screens and to make them more lightweight for client download, these lose the quality that certain technical images require. The ‘cumbersome collaboration tools’ is concerned with the use of forums and email to collaborate with tutors and fellow students. Essentially, in several instances the text written in these messages is rather lengthy and detailed. Carrying this out using a mobile device was found to be quite cumbersome. Analysing the overall UI evaluation results, the main deterrent for a learning experience over mobile devices seems to be that of low quality of the presented medium concerning the emphasis on textual presentation and the continuous navigation required due to restricted screen size. These account to nearly 80% of all reported issues.

5 Conclusions and Future Work

Taking the Mi-Learn LMS as a pilot evaluation environment for course delivery within an m-learning environment, one may conclude the following points, both from a pedagogical and user interface aspects. From a pedagogical point of view m-learning seems to be strong in content delivery but weak in creating the supporting environment in which this content is delivered. As shown by the evaluation, this supporting infrastructure consists of tools for helping student achieve overall academic objectives, and tools for helping a teaching institution to organize and monitor a programme of courses. This result hints that m-learning could be a strong addition to an existing e-learning environment, in delivering the additional comfort provided by the use of mobile devices, but difficult to offer a VLE of complete pedagogical quality, on its own. The UI evaluation results back the argument that m-learning would be best suited to complement e-learning (e.g. follow parts of a course whilst on the move, carrying out revisions and quick self-assessment tests etc.) Although from the pedagogical evaluation it resulted that m-learning is strong in content delivery, UI evaluation results show that even though the content is there, this is of lower quality than expected and strenuous to follow for a long periods of time. Content quality in
turn, heavily relies on the technology advances in the mobile device field, both on the transport field (increase in bandwidth) and the content level (the availability of lightweight graphics and animation formats).

Being the first phase of this research project, the main achievement was that of subject immersion and the setting a direction for the rest of the research project. The following is a list of candidate tasks to be carried out during the next phase.

- Inclusion of asynchronous communication between the LMS and the mobile clients in order to evaluate the additional value this may give to an M-Learning LMS, as per Pocket SCORM proposition;
- Analysing the possibilities of proposing an e-learning to m-learning content converter, that automates the conversion of existing e-learning content and adapting it for display on a cHTML client;
- Enhancement of the Mi-Learn LMS with an integrated SMS gateway that enables the delivery of progress reports and assessment results to phone enabled mobile devices;
- Refine both VLE and UI evaluations in order to make them sensitive to the type of mobile devices being used and investigating whether certain mobile devices are more adequate than others for following an m-learning course;
- An investigation of what are the m-learning specific strong points in respect to e-learning, and as a consequence analysing the best way that m-learning may complement e-learning.

References


Flexible Learning Systems:
An Insight into Personalised Learning Systems

Vanessa Camilleri and Matthew Montebello
Department of Artificial Intelligence, University of Malta
{vanessa.camilleri|matthew.montebello}@um.edu.mt

Abstract. Web services are defined as accessible software programs exposed through an Internet interface description which enhances client to server requests and are not only easily invoked and consumed but they provide interoperability for applications through Service-Oriented Architectures. The Semantic Web, Web services and Web technologies, have so far been mostly utilised in business models and processes throughout industry. This research paper proposes to show how these emergent technologies are also being exploited for E-learning environments. Such a service applies in fact not only to businesses and the work-place but also to academic settings. The ability to make a provision for flexible, personalised and adaptable services is heavily dependent on Web technologies which need to be moulded into rich, dynamic and active environments based on individual user needs and requirements. The paper aims to highlight ongoing projects in this area offering a brief description of their findings and achievements as well as identify future trends in the areas of flexible learning systems.

1 Introduction

E-learning has developed and evolved over the years from learning management systems, dealing with course management, design and delivery through systems which involve the learners in a more active rather than a passive way. Hart, [Har07] mentions how the approach to e-learning is now being moulded according to a methodology which stimulates collaboration and sharing in peer to peer environment. This is at the core of the socio-constructivist approach as is the knowledge sharing in an environment which stimulates harvesting and retrieving information for the sole purpose of learning, in what is also termed as an informal learning process. This new evolution of the virtual learning settings renders the online environment more learner-centred which as Downes [Dow05] refers to “is the placing of the control of learning itself into the hands of the learner”. This new evolution stems from the new vision of the web, the semantic Web or Web 2.0 “in which information is broken up into microcontent units that can be distributed over dozens of domains. The Web of documents has morphed into a Web of data... now we’re looking to a new set of tools to aggregate and remix microcontent in new and useful ways” [MP05]. Commercially this transition is seen in sites which have migrated from simply displaying static content to
deploying services in the Web context, which provide interfaces to content which is shareable. Such a provision of information as applied to the e-learning context makes the whole learning experience more dynamic and flexible around the user’s educational needs. Downes refers to this approach to learning as being different with the content which becomes distributed in more heterogeneous format in a manner which syndicates information such as a blog post or podcast to be collated and accumulated by the learners. This idea renders a wider perspective to the terms ubiquitous computing as applied to e-learning, where the concept of education which is context free, platform free, independent of time and location, makes learning and living merge together. “The challenge will not be in how to learn, but in how to use learning to create something more, to communicate” [Dow05].

This paper is structured as follows; a brief description of flexible learning systems which also includes an overview of personalisation and the effect of the Semantic Web and Web Services to facilitate such online flexible systems, is given at the beginning. The paper then looks at emerging trends in the approach towards more dynamicty within online learning systems and the key developments in the personalisation and adaptability of systems which a number of projects are currently undertaking to conclude with some possible direction to future projects in the area of personalisation in online learning.

2 Flexible Learning Systems

In order to make learning systems more flexible and adapted to the learner needs and requirements, Dagger et al [DOL+07] have looked at service-oriented architectures and how to go about making use of them for Web Service choreography. The first emergent need for such platforms with a ‘service-vision’ stems from the number of increasing issues related to interoperability of structures as well as the efficiency and effectiveness of the orchestration and choreography of the various services. As MacManus and Porter [MP05], indicate in their article, the effects of the Web 2.0, the read-write web which is being brought to life by the new generation of demanding users, and upon which the new e-learning is being moulded, bring about a transition towards XML or Semantic markup, as well as the provision for Web Services, making users move away from the local static environment towards a domain based on the dynamic discovery and assembly of services in the learning context defined by the user according to his personal needs and requirements. Flexibility thus becomes defined in terms of the interoperability of services in the e-learning environment.

2.1 Static E-Learning vs. Dynamic Services

The evolution of a new generation of e-learning standards and platforms refers to a modification and a shift in the provision of services from the simple course management systems which are currently trying to advance more in the area of
content sharing and reusability but which however are still not entirely “learner-centric” [DOL+07] towards systems which are more adaptable to the learners’ needs. Dynamic e-learning services in addition according to the authors, not only “include traditional functionalities such as authentication, tracking, course management, scheduling, activities, tools, and assessment” but also “emerging functionalities such as personalisation, resource harvesting, context management, federated exchange, simulation, games, wiki, blogging, podcasting, and so on”. In order to come up with these functionalities a number of ongoing projects are currently working towards creating standards, frameworks and technologies upon which such service based activities can be built in order to provide the interaction necessary from different platforms using different environments.

2.2 Semantic Web Services for Education

The composition of Web Services, which can be very simply described as software components designed to be accessible by a number of applications, enhances their value in carrying out complex and multifunctional tasks which the user requires specifically [SWGS05]. Their primary benefit lies in the interoperability standards they operate under such as the use of XML-based messaging [Cav06]. Web services work in close conjunction with and are heavily dependent on Semantic markup, which provides for a machine-processable language which can be universally understood across platforms and environment. The advent of XML technologies, have in fact rendered static content displayed upon pages more dynamic, able to be shared amongst the different users using choreography for a workflow based process [MP05]. This integration of Semantic markup for the purpose of Web Services enhances their discovery and augment their inter-and intra-domain operability which lies at the basis of workflow processes. Such an integration would not be possible without “domain ontologies” which need to broadly hold the content of the e-learning sphere. Semantic Web, Web Services, and Ontologies together with the functionalities needed for collaboration, on-the-fly request processing, as well as business processes design and modification create the need for a workflow which binds all these aspects together. A whitepaper by the Workflow Learning Institute defines this workflow process as “the real-time result of collaboration between people and systems (the workforce) in the WorkSpace” [Adk03]. This would in addition mean that Web services would be modified to embed e-learning functionalities, such as personalisation through the use of service-oriented architectures, in a way that e-learning structures become events triggered by other applications.

2.3 Personalisation and Adaptivity for Learners

The need for more learner-centric adaptations in the field of e-learning stems from the evolution of new technologies which allow for the flexibility and dynamicity in such a way that the learners can choose their own course of learning in the manner which is most suitable to their needs [Att07]. These are the
thoughts generated within the report for Learning Light, regarding the generation of demand for e-learning. According to the authors of the report “the increasing pervasiveness of ICT in the workplace and home provides a platform for a wide range of learning opportunities that many organisations will be able to exploit to enhance the flexibility and motivation of their workforce, and to encourage lifelong learning generally. Searching the Internet for ‘just-in-time’ information to meet a particular need has become an everyday habit for many — with the scope to extend this to further informal and formal discovery and learning” [JKS05]. Although the report takes the idea of e-learning from the workplace for employees, this same concept can be applied to university level students who wish to build and expand on the theories within classes to supplement their own research demands which at times vary from the static and often highly specific content delivered in the classroom. Thus the ability of the learners to gain control over their own learning process lies at the basis of personalisation in learning. This makes them more participative and active within their own learning process stimulating motivation.

3 Emergent Trends in Personalisation and Adaptation for Learners

Personalisation making use of new Web technologies has given rise to a number of projects and working groups whose aim is to establish standards and frameworks needed for the integration and choreography of the Web Services in order to “modularise functionality” [DOL+07]. Such is the scope of the IMS Abstract Framework\(^1\), as well as the E-Learning Framework\(^2\) which identify core components of e-learning and the e-learning systems common functionalities respectively. The Open Knowledge Initiative\(^3\) on the other hand defines service layers for e-learning, promoting specifications which describe how software components communicate with each other and with other applications in the online environment. Various projects are in addition pioneers in the creation of a range of sustainable “e-learning services” based upon collaboration and sharing of expertise through the use of Web technologies.

3.1 Simple Knowledge Organisation Systems

SKOS, Simple Knowledge Organisation Systems\(^4\) which falls under the W3C’s Semantic Web Interest Group, also informally known as ‘Something Kool, Original and Sexy’ refers to a framework which is simple yet powerful enough for expressing knowledge structures in a machine-processable manner for use on the Semantic Web. This framework is moulded around an RDF Schema for thesauri,

---

\(^1\) IMS http://www.imsglobal.org/ep/
\(^2\) ELF http://www.elframework.org/
\(^3\) OKI http://www.okiproject.org/
\(^4\) SKOS http://idealliance.org/proceedings/xtech05/papers/03-04-01/ & http://www.w3.org/2004/02/skos/references
classification schemes and taxonomies amongst others [MMWB05]. The concept of using Resource Description Framework (RDF) is that of providing interoperability between data sources which can be “distributed in a decentralised way, but still be meaningfully composed and integrated by applications, possibly in novel and unanticipated ways”.

3.2 iClass

iClass\(^5\) stands for Intelligent distributed Cognitive-based open Learning Systems for Schools. Its main aim is that of providing an environment which is adaptable to learners’ needs, as in personalised learning systems. It aims to design, develop and implement a framework which is based on a classroom based pedagogical model emphasising self-regulated learning. The system collects and processes data from the users in such a way that it is presented back to them in manner which supports a meaningful path of their learning process. iClass makes use of services which serve to model the learner behaviour through profiling and monitoring.

![Fig. 1: The I-Class Content Flow Model [TGC05]](http://www.iclass.info/iclass01.asp)

\(^5\) iClass http://www.iclass.info/iclass01.asp
3.3 PROLearn

PROLearn\(^6\) is a ‘Network of Excellence’ which deals with ‘technology enhanced professional learning’. Thus it deals with aspects of providing insights into the world of lifelong learning when it comes to the working experience for the benefit of employees who wish to keep up with the knowledge their profession requires. PROLearn which is an FP6 funded project which is based on a number of research activities and deliverables in the area of professional learning within the ‘workplace setting’. One of its research areas focuses on personalisation in learning.

Adaptive Personalised Learning Service APeLs APeL\(^7\) is an initiative to adapt and integrate a variety of perspectives on personalised learning. The project in itself delivers standards as well solutions for personalised learning systems through the interoperability of learning components in order to provide a flexible and adaptable learning experience.

3.4 Unfold

This project which ran up until 2005 aimed to provide an implementation for the IMS Learning Design and framework for the purpose of solving interoperability issues. The UnFold\(^8\) project aims at education in the industry sector using the learning designs and standards already in place by the IMS. This project has also maintained a close working contact with PROLearn through the setting up of specialised communities of practice (CoPs) in order to share expertise and collaborate on learning design for teachers and designers themselves as well as systems development.

The concept behind the whole project was that of supporting the IMS Learning Design whose aims and objectives were to enable “flexible and sophisticated pedagogical approaches to eLearning, by providing support for:

- multiple as well as single learners and their coordination
- a wide range of present, as well as future, pedagogical models
- learning activities and learning services, as well as content” [Gri05].

Therefore the Unfold project was created to provide the support for this learning design, “conceived of as a measure to promote and coordinate the adoption, implementation and use of IMS Learning Design and related specifications”.

\(^6\) PROLearn [http://www.prolearn-project.org/index.html](http://www.prolearn-project.org/index.html)

\(^7\) APeLs [http://www.prolearn-project.org/articles/wp1/index.html](http://www.prolearn-project.org/articles/wp1/index.html)

\(^8\) Unfold [http://www.unfold-project.net:8085/UNFOLD](http://www.unfold-project.net:8085/UNFOLD)
3.5 L4All

The L4All\textsuperscript{9} project is a web-service based system for lifelong learners aiming to provide support for lifelong learners, providing easy access to information, planning for future learning, as well collaboration and sharing expertise amongst peers. The methodology which this project follows is that of eliciting User and Technical requirements for technology standards and existing services. The concept is that post-16 learners, are able to trace out their own personal learning pathways, thus creating a motivational route for these same learners to progress to higher education. This project which is also funded by the JISC Distributed e-learning programme, aims to offer “(i) interaction with a Web Portal that provides information on work-based, FE and HE courses and modules available to learners in the London region; (ii) personalised support in planning and reflecting on personal development and lifelong learning activities; (iii) advice on learning and personal development; (iv) support in designing and maintaining personal learning and development plans; (v) support for learners to share information and collaborate with peers and tutors” [FMO*06]. L4ALL thus enhances and stimulates motivation in learners for participating in lifelong education.

3.6 Tangram Project

The Tangram Project\textsuperscript{10} is better defined as being an Integrated Learning Environment for the Semantic Web. It makes use of Semantic Web technologies and Ontologies in order to build new content from existing content units to dynamically assemble learning content. Based on the number and the structure of the ontologies it makes use of, Tangram provides Learning Objects for Content Authors with little manual operations, whilst it also provides personalised learning content adapted to the students’ level of knowledge, learning style and personal preferences as well as quick access to a topic of interest. The project delivers a web application which not only makes it easier for content authors to provide their content but it also provides the necessary framework for rendering semantic annotation easier, which is then used to help the learners achieve more control over their own learning objects and individual knowledge units.

3.7 Adaptive Hypermedia Architecture (AHA!)

The AHA\textsuperscript{11} project started out in order to provide support to an online course, however the modifications which it underwent through have shaped it into an adaptive structure, built on user models based on concepts and specific user attributes. Adaptivity is reached through the utilisation of content level and link level adaptation for the same content based on user profiling. The concept

\textsuperscript{9} L4ALL http://www.lkl.ac.uk/research/l4all/
\textsuperscript{10} Tangram http://iis.fon.bg.ac.yu/tangram/
\textsuperscript{11} AHA! http://wwwis.win.tue.nl/~{}debra/ht03/pp401-debra.pdf
behind this project is that of providing users with a domain and adaptation model based on the user model using Java Servlet technologies and Applets. One of the future aims for improvements to this project lies in the development of a system which is capable of invoking other adaptive applications and cross-communicate with them.

4 Conclusion

This paper has looked at trends, ideas and concepts behind the application of systems which enhance personalisation in the online learning environment. The concern for an evolution in the virtual learning environment has stemmed from research branching out from the field of e-learning and pedagogical issues affecting learners. Learners are changing, their needs are changing and so are the technologies. Having statically presented content within a management system or virtual platform, which is more administrative rather than pedagogical is rapidly becoming extinguishable. A merge between the technologies such as Semantics, XML, and learning thus becomes a necessity for the benefit of the learners making use of the Web as a platform for their knowledge development. A number of projects are promoting lifelong learning for the work-setting and for professionals. Future developments seem to indicate an area which is still yet unexplored. The choreography and orchestration of Web Services across Universities and other higher education institutions reaching over different geographical locations and continents can be utilised to discover and invoke other services and applications which can in turn enable learners to trace their own path to learning, participating actively in furthering their knowledge levels in academic research. Semantic mark-ups, technologies such AJAX, the use of Ontologies, and Web Services are all indicative of the way forward across which the new Web is evolving. Extending this way forward to learning accessible for all, irrespective of place, location, usable technologies, and learning styles seems one of the logical steps in the evolution of e-learning.

References


Semantic Web Trust: The next step in web evolution

Steven Caruana and Matthew Montebello
Department of Artificial Intelligence, University of Malta
{scar009|matthew.montebello}@um.edu.mt

Abstract. The inception of the World Wide Web marked the beginning of a new age, an age where information is easily distributed and accessible to everyone. Years after it was conceived, the net is still growing at a higher rate than ever. It is becoming ever more apparent that the World Wide Web’s current software infrastructure will need to evolve if it is to remain a reliable and dependable resource. In this paper we will be looking into the facets that make web content accessible and reliable. We will also be proposing a structure that makes use of technologies such as the Semantic Web and Agent Technology to help resolve data access and classification issues. The approach that we are proposing will involve the creation and distribution of data tags, policies and reasoners. We will also show how these items can be used by entities such as software agents or users to specify access control, classify the resources in terms of relevance and to decide on how trustworthy the information being used is.

1 Introduction

When a platform is released to the general public, it is inevitable that users will try to break it down. The internet is no exception to this rule and has already sustained a very wide number of different attacks. During the last few years both industry and academics have embraced these issues and have been developing successful solutions to each of the identified threats. Until a few years ago, the idea of storing sensitive information such as credit cards or personal documents on the web would have sounded like a fairly ambitious goal. When this was first attempted there were numerous cases of people who had their details stolen and maliciously used both on and off the net. Nowadays users will not only trust the web with this information but will also rely on it to perform most of the functions that would usually be done by the person manually.

With security threats being ironed out regularly, it is now imperative that we start to look towards the next step in the information superhighway’s evolution. The idea is to start shifting the focus of web research from the storage and transfer of information to trying to make it more valuable to the user. In the following sections we shall be discussing the motivations for needing a new extension to the net and shortly after is a description of a number of structures that are aimed at making this system a reality.
2 Security and trust

In this paper a distinction is made between security and trust. Security is the term used to refer to how prone information is to being stolen or misused. On the other hand, trust is the term we use to represent confidence that users have in the data they are accessing. Our view of trust falls in with Cristiano Castelfranchi’s and Rino Falcone’s [CF00] view of trust. They stated that trusting an entity involves both the acceptance of a certain amount of risk and that this risk should be assessed and quantified. In their paper they also stated that only a cognitive agent can decide whether to trust another agent or not.

The definition of a cognitive agent is that of an agent which requires both goals and beliefs. The current web does not enforce a standard to define these structures. For agents and users alike to be able to decide intelligently on whether a piece of information can be trusted, we require a new structure. This structure will need to handle both the specification of such notions and to make provisions for the resolution of such dilemmas.

In the last few years there have been numerous advancements in the field of security but little progress was made in trying to make data trustworthy. There are numerous factors that can determine how trustworthy data is. We envisage that the WWW requires a structure that can represent such relations so that it can in turn be made available to users to help them decide on how essential and relevant what they are viewing is to them.

3 Looking beyond authentication

Standard authentication provides users with a means of asserting their identity on the net but makes no provisions for users to express relations associated to the resource.

Relations are important because they allow users to customise complex structures that can be used to express information relating to the data. Once defined, these relations can then be used by the servers for access control or by users for filtering out what they are really interested in viewing.

There have already been a number of proposals for systems that deal with trust resolution in social networks. [MA04] proposes a solution for the calculation of the trust level attributed to an item. These calculations are based on the feedback set by users of the collaborative system. This structure is but one of the methodologies that can be used to express trust. In [VDCP06] the authors discuss how the calculation of trust can be done using different methodologies and how these methods can be applied to different scenarios.

In the papers mentioned above, all the systems that were referred to made use of boolean quantifiers to provide results for the resolution of trust evaluation. In [VDCP06] the authors also discuss trust functions that, rather than returning a boolean quantifier return a score that will signify how reputable the source is. [WV03, Mui03] explore the possibilities of reputation systems to decide on the reputation attributed to objects in different scenarios.
In the following section we will be looking at a structure that aims at providing an extension for the web that will allow us to add trust to the World Wide Web.

4 Introducing Trust into the Web

While designing this project we realised that it would be necessary to ground our work using Semantic Web technology as one of our building blocks. Semantic Web implementations of trust resolution frameworks have been under development for a number of years now and projects such as Rein [KBL05] and KAoS [UBJJ04] have already released implementations of current drafts.

As mentioned earlier, the system we are developing requires that entities, relations and rules be described in a machine and human readable language. DAML+OIL [Hor02] and OWL [DCHH+02] are two languages born from the Semantic Web initiative. The layer responsible for the trust reasoning will be composed of two subsections. The first section will constitute an interface into which a reasoning engine can be plugged in. The other part of this repository will refer to an archive that will be used for the storage.

4.1 Semantic Web

To construct this framework we chose the Semantic Web to be one of the building blocks. What made the Semantic Web a plausible foundation is the fact that it provides languages and tools that can be used for the tagging of information. When the Semantic Web [BLHL01] was first released to the general public, trust was defined to be one of the problems that the Semantic Web Research aimed at resolving. This project goes very much hand in hand with this initiative.

The reason for us adopting these technologies is the fact that the Semantic Web provides us with a set of languages that have developed specifically for the describing of entities and the relations between them. Languages such as OWL [DCHH+02] are already being used to build social networks such as Foaf [FOAF].

4.2 Agents

Agents play a very central role in this setup. Automation of processes has always been a very important aspect of computing. To date computers are already used to automate known processes. This implies that if a process can be formalised, then applications can be written to relieve the user of the repetitive parts of their jobs.

We believe that the next step in the design of automation software is that of creating applications that can make decisions in environments that they have not been designed to deal with. The development of trust layers is envisaged to provide agents with a set of tools that will help them deal with such scenarios. If users are given a means to define what information can be trusted, then they can also specify what resources agents are to trust and how to make use of
them. It is our belief that reasoning layers are not to be used exclusively by web applications, but also by applications such as agents to define the flow that can be used to perform certain tasks.

Another important role that we foresee agents will be taking on, is that of access control to information. Different users have different needs. The system that we designed is meant to help users define these needs and to make it possible to express them to agents. Agents can be used to filter information on the net that does not fall within the criteria that the user has specified.

5 Infrastructure Design

Fig. 1: The diagram above depicts the two main modules in our design and their respective internal structures. The lines between them show the interaction that takes place between these modules and the users of the system.

The design adopted by this project is a modular one. In modularising the components of this system we believe that it will be possible to attain a number of loosely coupled structures that can interconnect at runtime and be deployed in different locations and on different platforms.

The platform we designed is made up of two major modules which are the Semantic Web Archive and the Semantic Web Reasoner. Semantic Web Archive modules can be deployed at different locations and will store the tags that users will define. Like Semantic Web Archives, the Semantic Web Reasoner modules can also be deployed at different locations. Semantic Web Reasoner modules are responsible for the storage of the reasoners that will be used in this semantic network.

5.1 Semantic Web Archive

The Semantic Web Archive is the module that will be responsible for the storage of both the policies and tags. In a typical setup, instances of these modules will
be spread across the net and users who have access to them will be offered an interface through which they can manage the archiving of policies and tags. This module is to be used by both users and reasoners. Users will be provided with an interface hosted directly through the module. This interface will help explain the policies that are stored in this module and will also provide visual aids for the user to annotate the data that is stored on the web server. Semantic Web Reasoners on the other hand will be provided with an interface that will allow them to query this module for the resources it is hosting. When a query is received, this module will first search for the data that is being requested and will then marshal it via a web service back to the Semantic Web Reasoner. For security reasons we believe that this module will require a guard. If unauthorised users or applications were to gain access to this module, they could download the policies and tags and find ways of exploiting them. The guard’s main goal will be responsible for both the authentication of the reasoners and that of the users. For users, a standard log-on mechanism is envisaged to provide the access mechanisms necessary to authenticate them. On the other hand, Semantic Web Reasoners will need to be registered with this module before they can be allowed to query the system. The registering of Semantic Web Reasoners will be done by administrators managing these modules.

5.2 Semantic Web Reasoner

The Semantic Web Reasoner module is made up of two layers. These are the communication layer and the reasoning engine layer. The reasoning engine was designed to support pluggable reasoning modules. Developers can develop their own reasoners and embed them into the Semantic Web Reasoner. The communication layer of this module will handle the communication between services connecting to the Semantic Web Reasoner and the interconnections between the Semantic Web Reasoner and the Semantic Web Archive modules. This layer will also allow for calls to be generated between other external heterogeneous modules to process parts of these requests. Web applications will use this module to determine the result of policy requests. Based on the feedback returned by the Semantic Web Reasoner, the applications will be able to classify the resource and decide on how to approach it. Like Semantic Web Archives, even Semantic Web Reasoners were designed to have a guard mechanism. In this case the guard is there for the registering of Semantic Web Reasoners between each other. This will ensure that when being called by another Semantic Web Reasoner, the calls are genuine.

5.3 Tags and Policies

The system being proposed is based heavily on the generation and parsing of tags and policies. It is for this reason that we believe that it is imperative that the process of creating tags and policies be simple whilst still powerful enough to express the necessary relations.
To address these issues we designed a system that relies mostly on web forms to provide users with a standard implementation of an editor that can be accessed from different platforms. The current web is geared at providing a rich user experience. Our design involves the use of a graphical user interface that will be responsible for the visualisation of data structures (tags) and policy pipelines. Policies stored in Semantic Web Archives are allowed to span across resources and policies that are found on remote servers. It is for this reason that if a policy is to be reasoned out properly, the Semantic Web Reasoner will need to be registered with all of the required modules.

5.4 Policy Reasoning

The approach we decided to take in policy reasoning is that of creating a virtual structure resembling a pipeline. These pipelines are to be defined by users and each pipeline is to be made up of modules. In turn, each module is to receive a set of inputs and will generate a set of outputs. Once an output is produced, it is then pumped back into the next section of the pipeline. At the end of the pipeline, a set of results will be generated and sent back to the user or service. It will then be up to the system to decide how to react to this information.

Status propagation is another important aspect of this structure. If a user runs a query that fails whilst executing (e.g. a Semantic Web Reasoner or Semantic Web Archive is not found), then the user will need to be notified of this issue. To do this we decided that at every hop the call makes, an xml message will be generated and at every call more logs will be appended to it. When a call terminates (both successfully or in an error), this message is returned to the user’s machine or agent and the appropriate action is performed.

5.5 Publishing of Policies and Tags

Distribution of information is one of the major concerns of this infrastructure. If the data is not published properly, the consequences could be twofold. The data could either be too well hidden to be accessible by users who might need to reference it, or it could be accessible to the point where users with malicious intents could abuse of it.

To solve this problem we looked again at the web’s current infrastructure. We believe that the safest structure for this kind of network is that of having a secured network of data access points that can rely on existing standards of data encryption and security. For a user to have access to the various Semantic Web Archive and Semantic Web Reasoner nodes, they will need to have user accounts that are to be created by the administrators or service owners as discussed in the previous sections.

This structure will allow anyone deploying their policy network to decide for themselves how public they can afford their networks to be. Once the networks have been established, a user base can be maintained using the same structures that are being proposed in this paper. Servers can be made to decide which users will have access to the resources it is publishing.
5.6 Adoption of Relation Tagging

Making the shift to relational data on the net has been part of the focus of this project since its inception. It is for this reason that part of this project is aimed at providing user friendly tools to help better understand the potential uses of relations and to make them accessible to as wide an audience as possible. It is a known fact that if users are presented with an unfamiliar environment they will feel reluctant to adopting new mechanisms. To address this problem, we envisage the adoption of a graphical user interface that can be called up from a browser. This interface is to be accessible from anywhere on the net and ubiquitous across most platforms.

6 Conclusions and Future work

The addition of trust to the web infrastructure is an important step in the web's evolution. As the web grows, we are slowly losing control over the authenticity of data published and trust in web applications is a topic that will become ever more important as time goes by.

Once this framework is constructed, we believe that more work will be necessary to ensure that the framework is compatible with different environments (such as mobile phones) that might not support the tools that we are offering.

References

[FOAF] The Friend of a Friend (FOAF) project.: http://www.foaf-project.org/

Embedded Languages for Origami-Based Geometry

Gaetano Caruana and Gordon J. Pace

Department of Computer Science, University of Malta

Abstract. Embedded languages have been used to support compositional descriptions for various domains. In this paper, we look at the domain of paper folding, or Origami-based geometry, in which sequences of paper folding are used to describe points and lines on the plane. Based on seven basic origami axioms, we design and develop an embedded domain specific language for the descriptions of such constructions in Haskell. We argue that the embedded language approach, that is composing a model using the basic constructors in the domain specific language, gives a compositional and concise way to describe Origami models. We look into analysis, manipulation and generation of origami models using this approach, including textual explanations of models, analysis of models to discover inherent preconditions (or constraints) in a description and basic animation of the folding of a model. Finally, we look into the tagging of blocks within a construction, enabling different evaluations at various levels of abstraction according to the user’s knowledge of Origami.

1 Introduction

Language design has been an important area of research in computer science right since its inception. It is generally accepted that for designing a specialised, domain-specific language can aid and simplify the design and specification of algorithms in that domain. Various such domain specific languages have been developed for various different areas. Rather than design a full domain specific language (DSL) from scratch, with operators to features generally found in most languages to support features such as iteration and algorithm reuse, an increasingly used approach is to embed the basic domain specific language inside another programming language. The language in which the DSL is embedded, called the host language, acts like a container for the embedded language, and is used to build functions that describe, analyse and manipulate programs written in the embedded language. This approach guarantees that the features of the host language are automatically inherited by the embedded language [Hud96]. By inheriting features from the host language, the domain-specific embedded languages designers are relieved from having to reinvent the wheel to support commonly found features in general-purpose programming languages. Furthermore, building a new embedded language does not require the implementation of new development tools by sharing the host language’s compilers and interpreters,
providing only interpretation mechanisms for the domain-specific features, making the building of a DSEL much simpler than designing and supporting a DSL from scratch. Apart from making use of existing infrastructure of the host language, one can combine different DSELS embedded in the same host language, thus enabling easy combination of languages, making them more extendable than DSLs [Hud96,LM99].

In this paper, we look at design of an domain-specific embedded language for Origami (paper-folding) based geometrical constructions. Ever since Euclid gave his axioms of planar geometry, expressing how one can construct and reason about points, lines and circles on the plane, different techniques have been explored into the expressiveness and relation between different ‘tools’ which limit what constructions can be derived from other ones. It is well known, for instance, that trisecting an angle using straight-edge and collapsing-compass is, in general, impossible. Various alternative tools have been explored, one of which is based on Origami, the Japanese art of paper folding. Using this geometrical tools, one only construct new points and lines by folding the plane (and unfolding it back) using already known points and lines for reference. We embed a small domain-specific language to express such constructions in Haskell [Jon03]. Based on these descriptions, we build a library of functions to manipulate, analyse and visualise such constructions. Since large models can become rather repetitive to explain to a user, and one should be able to simply state the name of commonly found sequences of constructions to advanced users, we explore different techniques to tag blocks in constructions to enable us to give more compact descriptions.

2 Related Work

Various embedded domain-specific languages have been developed in the literature for domains as diverse as financial contracts [Jon01], hardware-description languages [CSS01,DLC99] and stage-lighting [Spe01]. The common trait in these applications is that objects in these domains can be complex and difficult to handle as a single entity, but can be expressed into the composition of smaller objects. Such decomposition, especially if regular, enables simple descriptions of complex compound objects. A library of basic objects, and combinators for their composition thus make up the domain-specific language. These descriptions can then be manipulated and evaluated using different techniques without having to change their description. For example, in the case of financial contracts, one can analyse the expected value of a contract, or simulate it with projected future data on which it may depend (such as exchange rates), or even produce a natural-language rendition of the contract. In the case of the embedded hardware-description language Lava [CSS01] one can not only simulate circuits, but also produce VHDL descriptions and verify properties of the circuits by using external model checking tools.

Although purely-functional languages have been shown to be very appropriate vehicles for embedding languages, one recurring problem in such descriptions has always been that of identifying blocks induced by the description visible at
the host language level, but which are lost at the domain specific level. An area in which this restriction has been recurringly encountered is that of embedded hardware description languages. Although various regular, yet complex circuits can be described using a recursive algorithm which induces the gate connections [CSS01,DLC99], one loses the ‘virtual’ blocks one automatically goes through upon every function call recursive or otherwise. Some different techniques have appeared recently [ACS05,Pac07] looking at component and block oriented view of circuits. In Wired [ACS05], circuits are seen as relational blocks which can be composed together using a set of combinators, allowing various non-functional features of such circuits to be analysed. In [Pac07], another component-based approach is taken, moving away from the purely functional descriptions in Lava and combinator-based approach in Wired, looking more at the connection descriptions with explicit wire parameters. Our approach to describing blocks in our language differs from these, by providing an explicit tagging construct which can be used to tag the boundaries of a sub-construction. Although the use of explicit tagging can be tedious and prone to error, we only require tagging in few places for our domain. Furthermore, our descriptions still look functional, and less component-oriented than the other approaches we have mentioned, more in keeping with the host language.

3 Origami

The ancient Japanese art of paper folding, Origami, may seem deceptively simple and straightforward. Origami models can be constructed after a few days of practice, but advanced techniques and models require years of experience and knowledge to execute well, and the design of new models requires a high degree of creativity. Origami is based on the repeated folding of a sheet of paper, to form models, usually resembling real-life objects. Although variants of Origami allowing the use multiple sheets of paper (see, for example, [Mit77]), or glueing or even the cutting of the sheet of paper exist, the traditional approach is to use one sheet of paper and allow only creases with respect to other known creases or points (corners of the sheet of paper or intersections of creases) are allowed.

3.1 The Geometry of Origami

Usually Origami is viewed simply as the art of building models, through the performance of a sequence of folds, in some cases folding the sheet of paper and unfolding it back (to mark a crease on the paper), in others keeping the paper folded. However, the mathematical analysis of these constructions yielded an interesting geometry. Most people are familiar with straight-edge and collapsing compass geometry in which lines can be constructed through the use of an unmarked straight-edge (to draw a line between two known points), circles constructed using a collapsing compass (which allows the drawing of a circle centred on a known point, passing through another known point) and points
which can be constructed only through finding the intersection of two known shapes. It is well-known that using only such construction techniques, one cannot deduce certain positions on the plane. Most famously, one cannot trisect an arbitrary angle, square a circle or double a cube using such techniques. Origami constructions are made up of only folds (straight lines) and points. New lines can be deduced by folding the paper (in a limited number of ways) and points through finding the intersection of two lines. Folds are temporary, used only to induce a line, with the sheet of paper then being unfolded. Interestingly, certain points and lines can be constructed using the Origami operations but not using straight-edge and collapsing compass (and vice-versa).

In our Origami DSEL (OriDSEL), we consider this type of geometric interpretation of Origami:

- the paper is considered to be an idealized mathematical plane;
- a fold line is considered to be an infinitely extended line on the plane;
- a reference point is an idealized point on the plane.

### 3.2 The Axioms of Origami Geometry

As in the case of straight-edge and collapsing compass geometry, in Origami-based geometry, one can only construct lines and points in a set of well-defined ways. Origami constructions have been reduced to seven underlying axioms:\(^1\):

**Axiom 1:** Given two points \(p_1\) and \(p_2\), one can construct a line that passes through both of them.

**Axiom 2:** Given two points \(p_1\) and \(p_2\), one can construct a line, folding along which, places \(p_1\) onto \(p_2\).

**Axiom 3:** Given two lines \(l_1\) and \(l_2\), one can construct a line, folding along which, places \(l_1\) onto \(l_2\).

**Axiom 4:** Given a point \(p_1\) and a line \(l_1\), one can construct a line folding along which places \(l_1\) onto itself (in other words, is perpendicular to \(l_1\)) and that passes through point \(p_1\).

**Axiom 5:** Given two points \(p_1\) and \(p_2\) and a line \(l_1\), one can construct a line passing through \(p_2\), and folding along which, places \(p_1\) onto \(l_1\).

**Axiom 6:** Given two points \(p_1\) and \(p_2\) and two lines \(l_1\) and \(l_2\), one can construct a line, folding along which, places \(p_1\) onto \(l_1\), and \(p_2\) onto \(l_2\).

**Axiom 7:** Given a point \(p_1\) and two lines \(l_1\) and \(l_2\), one can construct a line, folding along which, places \(p_1\) onto \(l_1\) and \(l_2\) onto itself (in other words, is perpendicular to \(l_2\)).

---

\(^1\) The original six can be found in [Huz92], while the seventh, so-called Hatori's axiom, was added later — see http://origami.ousaan.com/library/conste.html for more details
4 OriDSEL — Embedding Origami Axioms

The axiomatisation of Origami constructions provides a straightforward way in which we choose to embed the language of Origami constructions. In OriDSEL, we choose to use a deep embedding, to have access to the structure of a construction, enabling us to provide different interpretations of a single model. Internally, two basic types are used to model points and lines (or folds):

```haskell
data Point =
    Intersection Line Line
  | ...
data Line =
    Axiom1 Point Point
  | Axiom2 Point Point
  | Axiom3 Line Line
  | ...
```

In keeping with the host language, we choose to show constructions to the user as functions from a structure of points and folds to a structure of points and folds. The axioms themselves are visible to the user as functions, with names indicating their behaviour:

```haskell
intersect :: (Line, Line) -> Point
intersect = uncurry Intersection

foldThroughPoints :: (Point, Point) -> Line
foldThroughPoints = uncurry Axiom1

foldPointOntoPoint :: (Point, Point) -> Line
foldPointOntoPoint = uncurry Axiom2

foldLineOntoLine :: (Line, Line) -> Line
foldLineOntoLine = uncurry Axiom3
...
```

Thus, for example, given a rectangular sheet of paper (as four points), we can give a construction to select the four edges (lines) of the sheet of paper:

```haskell
fourSidedPaper ::
  (Point, Point, Point, Point) -> (Line, Line, Line, Line)
fourSidedPaper (nw,ne,se,sw) = (n,s,e,w)
  where
    n = foldThroughPoints (nw, ne)
    s = foldThroughPoints (sw, se)
    e = foldThroughPoints (se, ne)
    w = foldThroughPoints (sw, nw)
```
We can generalise this, such that, given a polygonal sheet of paper (as a sequence of points), we can give a construction to select the edges (lines) of the sheet of paper. This can be used to give an alternative definition of `fourSidedPaper`:

```haskell
edgesOfPolygon :: [Point] -> [Line]
edgesOfPolygon vertices@(v:_)
  = [ foldThroughPoints (p,p') | (p:p':_) <- tails (vertices ++ [v]) ]

fourSidedPaper' (nw,ne,se,sw) = edgesOfPolygon [nw,ne,se,sw]
```

Using an abstract datatype to describe Origami models, constructions correspond to a tree. In practice, one can have sharing of points or lines as in the following example of a standard construction (see figure 1):

```haskell
kite (nw,ne,se,sw) = (nw, point_e, se, point_s)
  where
diagonal = foldThroughPoints (nw, se)
  (n, s, e, w) = fourSidedPaper (nw,ne,se,sw)

  l1 = foldLineOntoLine (n, diagonal)
  l2 = foldLineOntoLine (w, diagonal)

  point_e = intersect (l1, e)
  point_s = intersect (l2, s)
```

In this example, the diagonal is used twice in the construction. However, due to referential transparency, there is no way we can differentiate the construction from a similar one, but which replaces the two references to `diagonal` by `foldThroughPoints (nw, se)`. In practice, when traversing the data structure produced, we would want to identify the common parts. This will avoid, for
instance, describing how to obtain diagonal twice, when explaining how to construct the kite. Various solutions have been proposed in the literature, including explicit naming of structures [O’D92] and the use of the state monad to thread references to definitions [CSS01]. Finally, we opted for the use of observable sharing [CS99], which enables a cleaner description, albeit breaking the functional purity of the language. In practice, the use of references is hidden within the basic datatypes and the axiom definitions, thus enabling the user to write the programs we have shown with no knowledge of the underlying machinery.

The advantages of the use of an embedded language, with the host language as a meta-language of the domain specific language become more apparent when we produce regular repetitive constructions as in the following example:

```hs
cross (nw, ne, se, sw) = (ns, ew)

where
  ns = foldPointOntoPoint (nw, ne)
  ew = foldPointOntoPoint (ne, se)

subsquare corners =
  (intersect (ns,n), intersect (ew, e),
   intersect (ns,s), intersect (ew, w))

where
  (ns, ew) = cross corners
  (n, s, e, w) = fourSidedPaper corners

repeatedSubsquare 0 square = square
repeatedSubsquare n square =
  subsquare (repeatedSubsquare (n-1) square)
```

4.1 Manipulation of Origami Constructions

The availability of a meta-language to the domain-specific language enables us to provide a library for the analysis and manipulation of programs in the embedded language. We provide a number of such functions to enable the user to explore and study the constructions described.

Explaining a given construction: We provide functions which explain an Origami construction and produce a textual explanation of how it can be achieved by traversing the directed acyclic graph describing the construction. We provide both plain text and HTML descriptions, explaining the model in a step-by-step manner. HTML descriptions are richer, giving links between the different parts of the construction when referring to previously described folds or reference points. It is important for us to identify sharing in the given model, to avoid repeated descriptions of the same construction. Below is the text description of a kite fold given earlier:

Fold the paper along points NW and point NE, calling it line 1.
Fold the paper along points SW and point NW, calling it line 2.
Fold the paper along points SW and point SE, calling it line 3.
Fold the paper along points SE and point NE, calling it line 4.

Fold the paper along points NW and point SE, calling it line 5.

Fold the paper by putting line 1 over line 5, calling it line 6.
Fold the paper by putting line 2 over line 5, calling it line 7.

Find the intersection of line 4 and line 6, calling it point 1.
Find the intersection of line 3 and line 7, calling it point 2.

The result is (NW, point 1, SE, point 2)

Animation: Textual descriptions can be useful for small constructions, but tend to become too long and complex for larger ones. Having to keep track of previously identified creases and reference points can quickly get out of hand. OriDSEL provides a link to an external tool we have build to show an animation showing, step-by-step, how the construction is achieved.

Constraint Checking: The Huzita Hatori axioms are partial functions in that they are only defined for some inputs. For example, axiom 5 (given two points and a line, draw a line going through one of the points and folding along which places the other point on the line) cannot be applied to if the points lie on opposite sides of the line. Because of this, constructions are also partial, in that for certain inputs, the model will fail. We provide functions to calculate the constraints that are to be satisfied in order for a construction to be well defined. These constraints can then be checked for concrete values of the vertices.

5 Partitioning of Models

As the number of folds in an Origami model increases, so does its complexity. In most of the Origami literature, complex Origami models are not described in terms of the basic folds, but rather in terms of so called base folds — each of which being an often used sequence of basic folds. Using Haskell, the user descriptions of a model in OriDSEL can be written to resemble the ones in the Origami literature. Starting off with a library of base folds, one can describe complex models in terms of these library functions. However, the internal description of these models obviously contains no information about which parts of the model were generated by which functions. We would like to add sufficient information in the internal structure to enable concise output descriptions, using compound constructions (such as base folds) in the description. Since such base folds vary in difficulty, we would like to enable tagging of blocks not only with a name (for reference), but also with a difficulty level. The user can then request textual descriptions taking his or her expertise level into account. Various techniques have recently been proposed in the literature to resolve this issue of named blocks in embedded languages. For instance, [ACS05,Pac07] take a component, or block oriented view of circuits, composed together using a set
of combinators rather than simply functional composition. Other approaches, such as [MO06] look at the use of meta-programming features to access this information. In our case, the problem is simpler — we only want to name and tag the difficulty of a few blocks, and their access is based entirely on difficulty level which ranges over few possible values.

One possible approach is to tag all nodes in a structure with a name and difficulty level. The main problem with this solution is that without additional internal machinery, there is no way of differentiating between two blocks with the same name connected together, and one large block with that name. Furthermore, one can make do with much less information in the model, to enable outputting descriptions in terms of named blocks.

The approach we take is to label output boundaries of a block with all relevant information about the block (name, difficulty level, input and output nodes) enabling structured output descriptions by referring to blocks of the appropriate level as a whole using the name stored on the boundaries.

```haskell
data SkillLevel = Beginner | Intermediate | Expert

data Boundary = Boundary String SkillLevel [Ref] [Ref]

data Point =
  OutputBoundaryP Boundary Point
  | Intersection Line Line
  | ...

data Line =
  OutputBoundaryL Boundary Line
  | Axiom1 Point Point
  | Axiom2 Point Point
  | ...
```

As with the underlying axioms, the above data structures are abstracted away from the user:

```haskell
block skill name construction ins =
  markAsOutput skill name (structToRefs ins, structToRefs outs) outs
  where
  outs = construction ins

beginner    = block Beginner
intermediate = block Intermediate
expert      = block Expert
```

The function `markAsOutput` marks a structure of output lines and points with the appropriate constructor and `structToRefs` transforms a structure of lines and points into a list of references to the data.

Users may now name blocks and label them with their difficulty level. For instance, the `subSquare` construction given earlier may be tagged as an intermediate level base fold with the name ‘subsquare-construction’ in the following manner:
The use of \texttt{subsquareBlock} is now identical to that of \texttt{subsquare}, enabling us to redefine the repeated subsquare construction given earlier using the base fold:

\begin{verbatim}
repeatedSubsquare 0 square = square
repeatedSubsquare n square = 
  subsquareBlock (repeatedSubsquare (n-1) square)
\end{verbatim}

Now, by calling \texttt{explainIntermediate (repeatedSubsquare 10)}, we get an explanation appropriate for the intermediate user. Rather than explaining all the constructions from scratch, the system will assume that the description is aimed at someone having an intermediate skill level, who thus knows what a \texttt{subsquare-construction} is, and will give just a ten line explanation, one for each application of the subsquare construction. On the other hand, by calling \texttt{explainBeginner} one will get a full description going down to the underlying axioms.

6 Conclusions

In this paper, we have explored the embedding of Origami geometric constructions in Haskell. The axiomatization of the problem domain has provided us with the necessary building blocks upon which to build the language. Using standard techniques from embedded languages, we have built a deeply embedding of such constructions and a number of functions to analyse and manipulate constructions. The axioms of Origami geometry have been explored well in the literature. However, looking into actual model creation with Origami introduces further challenges — the folding primitives remain unchanged, but extra information needs to be added to the folds to specify whether the fold is unfolded, the direction of folding and for more advanced models, the angle of the fold. The main challenge in this domain is the abstract description of models, enabling the user to hide information away for an expert, for whom a more abstract description of a sequence of folds would be sufficient. The solution we have developed enables explicit tagging of such blocks, which can be impractical in certain other contexts and a potential source of errors in others (through the reuse of tags). In our case, the use of a small number of tags (corresponding to the difficulty levels and names of basic folds), enables a simpler solution, which is used to effectively enable different descriptions of the same model. It would be interested to explore this technique in other contexts, such as textual explanations of proofs in theorem proving.

References


Abstract. Long-lived transactions (LLTs) are transactions intended to be executed over an extended period of time ranging from seconds to days. A long-lived transaction is normally organized as a series of activities, with each activity being a discrete transactional unit of work that releases transactional locks upon its execution. The long-lived transaction commits if all its activities complete successfully. Unless an activity requires the result of a previously committed activity, there is no constraint which specifies that the various activities belonging to a long-lived transaction should execute sequentially. In this paper we present a solution that combines long-lived transactions and CSP such that independent activities execute in parallel to achieve flexibility and better performance for long-lived transactions. We introduce two composition constructs SEQ$^{\text{LLT}}$ and PAR$^{\text{LLT}}$. Very much as the occam CSP-based constructs, SEQ and PAR, allow processes to be executed sequentially or concurrently, the proposed SEQ$^{\text{LLT}}$ and PAR$^{\text{LLT}}$ constructs can be used to specify the sequential or concurrent execution of transactions. Transactional CSP Processes is a framework that makes use of these composition constructs, providing an API through which the application developer can define long-lived transactions. Concurrency and transaction handling are managed by the framework transparently from the application developer.

1 Introduction

An atomic transaction is a unit of interaction between two or more parties which must be either entirely committed (completed) or aborted (fails) as a unit. Transaction integrity is supported through the ACID properties, which are:

- **Atomicity**: ensures that all of the tasks in a transaction complete successfully and a transaction commits its changes. However, if any of the tasks fail, the transaction is aborted and all its effects are rolled back.
- **Consistency**: refers to having a legal state before a transaction begins and after it terminates. Integrity constraints of the database should be adhered to and a transaction is aborted if any of these constraints is aborted.
- **Isolation**: a transaction should be treated independently from any other transaction. Other transactions should not be aware of the intermediate states produced by the transaction before it commits.
A transactional platform makes use of locks on rows or tables to guarantee such ACID properties. The short-duration of such transactions allows other transactions competing for the same resources to be queued seamlessly until locks are released.

A long-lived transaction (LLT) also referred to as a long-running transaction, is a transaction of long duration, generally ranging from minutes to hours or even days. For this reason, it is impractical to use locks throughout the duration of the long-lived transaction to prevent concurrent access from other transactions (which would violate the ACID properties). To address this issue, transaction models for LLTs relax the ACID properties by organizing a long-lived transaction as a series of activities. Each activity is a discrete transactional unit of work which releases locks upon its execution. Activities are executed in sequence and can commit, rollback or suspend execution of the transaction. The long-lived transaction commits if all its activities complete successfully. If any of the activities fail, the long-lived transaction should roll back by undoing any work done by the already completed activities. This is normally achieved through compensating activities that essentially reverse changes which, under a normal setup, would have been handled implicitly by the underlying transactional model. As described in the JSR95 model [Com06], a transactional model proposed as part of the Java Community Process which can be used to model long-lived transactions, “In the event of failures, to obtain reliable execution semantics for the entire long-lived transaction, compensation transactions are required in order to perform forward or backward recovery” [Com06].

One limitation of traditional long-lived transactions is that activities are executed in sequence — the flow of execution continues upon the successful completion of an activity to the next. There might be cases when an activity may require a third party service and so it will be suspended, blocking other activities. This results in the long-lived transaction taking a long time to complete. There is also the possibility that after waiting for a long time to resume execution, the activity may fail causing any committed activities to undo their work. This leads us to the motivation of our research, which will be described in section 1.1.

1.1 Motivation

We shall now describe the motivation to develop a model which combines concurrent processes and long-lived transactions through an example. Consider a travel agent system which is used by clients to reserve resources such as a flight, a room and appropriate transport arrangements between the airport and the client’s accommodation.

It is rather critical that the booking of such disparate resources is synchronized — it would be useless to reserve an accommodation unless we manage to reserve also a suitable flight. Traditional ACID transactions are not suitable to model such a transaction, as typically each of the flight reservation, accommodation
reservation and travel arrangement operations are done through different third parties and each takes a substantial amount of time. A long-lived transaction with a series of activities would be more suitable.

Fig. 1: Travel Agent LLT consisting of three activities to reserve a flight, a hotel room and a taxi, and a number of compensating activities to cancel each reservation in case of failure.

Typically we would model each of the various tasks as activities (refer to figure 1). If any of them fails, then any committed activities are compensated and the entire transaction is rolled back. A typical compensating action might involve canceling a committed room reservation as no suitable flight could be booked.

It is evident that there is no constraint which specifies that the various activities described in such scenario need to be executed sequentially. On the contrary, the booking is more likely to be more successful if we clear the reservations as quickly as possible. A solution would be to define a long-lived transaction that can have independent activities running in parallel without the need for them to wait for each other to start executing. Having concurrent activities would
eliminate the case where activities in suspended state will block the following
activities.
Transactional CSP Processes is a framework implemented to achieve this objec-
tive. This framework extends on SmartPay LLT (refer to section 2) by introduc-
ing composition flow constructs similar to the sequential and parallel operators
defined in the CSP calculus which allow an application developer to define the
desired method of activity execution in a long-lived transaction.

1.2 Paper Overview

The paper is organised as follows: Section 2 provides a brief introduction to
SmartPay LLT, an implementation by Ixaris Systems (Malta) Ltd. We will then
present the Transactional CSP framework, which extends on SmartPay LLT
in Section 3. This section presents details about the elements comprising the
framework including activities, how the framework handles failure, as well as the
composition constructs SEQ,LLT and PAR,LLT to be used in order to define the
desired execution of activities in a long-lived transaction. In Section 4 we
will highlight a possible extension to the framework which can be carried out as future work.

2 SmartPay LLT

In 2005, Ixaris Systems (Malta) Ltd developed a Java implementation loosely
based on the JSR95 Model [Com06]. This implementation forms part of a generic
SmartPay platform implemented by the same company.
The motivation for such a model was brought about by the inadequacy of tra-
ditional ACID transactions to address the company’s specific circumstances.
Generally financial transactions involve a mix of local database updates (fully
within a transactional context) as well as external communication with third
parties. The communication with such third parties cannot be done within a
normal transactional context; if the third party communication falls through, it
takes a period of time for the connection to timeout; during such a period all lo-
cal resources participating in the transaction are locked. Long-lived transactions
allow for the separation of remote interactions and local transactional updates.
The SmartPay LLT implementation extends the long-lived transaction model
proposed in the JSR95 with the possibility of suspending execution between one
activity and another. There exist situations when one needs to consult with a
remote system before continuing with the transaction (for example, if we are not
sure whether we have acquired funds from a client, then we need to suspend the
transaction until we check with the remote system before depositing funds in
the user’s local account).
SmartPay LLT is implemented on standard Java technologies namely Java Tran-
saction API (JTA) and Enterprise Java Beans (EJBs). JTA provide the Java Plat-
form with standards-based closed, top-level transaction support. EJBs provide
a persistence model for persisting intermediate transactional state. SmartPay
LLT 1.0 has been deployed on a live system setup (on a JBoss application server with a MySQL backend) for the past two years, processing thousands of financial transactions.

3 Transactional CSP Processes Framework

Transactional CSP Processes is a framework which allows the application developer to define the desired method of activity execution (sequential, concurrent or a combination of both) in a long-lived transaction. This framework extends on SmartPay LLT by introducing composition flow constructs similar to the sequential and parallel operators defined in the CSP calculus. The model also allows for the suspension and resuming of activities and addresses failure of activities in terms of compensating activities. The application developer simply determines the activities to be performed and specifies their method of execution by using the appropriate composition constructs. Concurrency and transactional issues are managed by the framework implementation, transparently from the application developer.

A long-lived transaction in Transactional CSP Processes framework is defined in terms of activities and their composition structures, using the proposed sequential and parallel composition flow constructs SEQ_LLTT and PAR_LLTT. In the following sections we will present the various elements that comprise the Transactional CSP framework.

3.1 Activities

Similar to the SmartPay LLT, the application developer must implement activities to define the units of work of a Transactional CSP LLT. The method of execution for each activity is defined by adding the activity to the desired composition flow construct SEQ_LLTT or PAR_LLTT. In the Transactional CSP Processes framework, the long-lived transaction is modeled using a tree structure in which a branch determines the concurrent execution of the elements belonging to it. Activities are to be added as child elements to the required composition structures:

- SEQ_LLTT for sequential composition
- PAR_LLTT for parallel composition

3.2 Compensating Activities

This framework adopts backward recovery so when an activity fails, all previously committed activities are compensated through their corresponding compensating activities. A compensating activity essentially reverses changes which, under a traditional atomic setup, would have been handled implicitly by the underlying database model. The application developer specifies the course of action to be taken for each committed activity to undo its changes in a corresponding compensating activity.
It is our understanding that all activities related via SEQ.LLT and PAR.LLT constructs belong to the same transaction. The failure of any activity brings about the compensation of all committed activities participating in the same LLT. Compensating activities will be executed in the same order and under the same constraints as the activities being compensated. When compensating activities in a SEQ.LLT, their corresponding compensating activities are executed sequentially but in reverse order, while those activities in a PAR.LLT are compensated concurrently.

3.3 Composition Flow Constructs: SEQ.LLT and PAR.LLT

Very much as the occam CSP-based constructs SEQ and PAR allow processes to be executed sequentially or concurrently, the proposed SEQ.LLT and PAR.LLT constructs can be used to specify the sequential or concurrent execution of activities in a long-lived transaction.

Two activities that are coordinated with the SEQ.LLT construct (Figure 2) are evaluated in such a way that the second activity is executed only after the first activity commits. This corresponds to the SEQ construct which, from a concurrency perspective, executes in such a way that the second process starts its execution after the first process is complete. Therefore, SEQ.LLT requires a single thread of execution for its elements to execute in sequence.

Fig. 2: SEQ.LLT construct is used to specify the sequential execution of activities in a long-lived transaction, with each activity executing one after the other on the same thread of execution.

Similar to occam’s PAR construct, the PAR.LLT construct (Figure 3) specifies that activities can start their execution, independently from whether any other activities have committed their transaction or not. PAR.LLT will spawn a thread for each of its elements so that they will execute in parallel. PAR.LLT will then wait for all child threads to return a result to their original parent thread. In other words, PAR.LLT will wait for all child threads to join back to their original parent thread.
Fig. 3: PAR,LLT construct is used to specify the concurrent execution of activities in a long-lived transaction, with each activity being executed on a separate thread.

SEQ,LLT and PAR,LLT can be combined in very much the same way that SEQ and PAR in occam can. For example, if two activities B and C can run in parallel but require activity A to successfully commit first, the setup of SEQ,LLT and PAR,LLT as shown in Figure 4 is to be used. In the case where a PAR,LLT construct is to be followed by a SEQ,LLT construct sequentially, like that depicted in Figure 5, the nested SEQ,LLT activities start execution after all activities in the PAR,LLT have committed. PAR,LLT will first spawn threads for each of its activities. PAR,LLT will wait for all its threads to join back to their original parent thread before the enclosing SEQ,LLT can proceed to execute the following element in sequence.

3.4 Suspending and Resuming Activities

An activity can commit its updates, rollback any updates or suspend execution such that it is resumed later on. Any updates done up to that point by the activity can be committed or rolled back, as specified by the activity. An activity which suspends execution in a SEQ,LLT construct, indirectly delays the execution of any subsequent SEQ,LLT activities. Such activities cannot start their execution until the activity commits. On the other hand, an activity which suspends execution in a PAR,LLT construct, does not have any effect on other activities executing in the same PAR,LLT construct. Any activities which are synchronized to start their execution after the PAR,LLT activities commit will wait until the suspended activity is resumed and completed.
Fig. 4: A long-lived transaction made up of a nested composition of constructs in which Activities B and C first wait for Activity A to commit successfully, and then they are executed in parallel.

Fig. 5: Another example of a long-lived transaction made up of a nested composition of constructs, with a number of activities (C and D) to be executed in sequence, after a number of activities (A and B) have been executed successfully in parallel.
4 Future Work

Transactional CSP Processes framework could be extended to support communication between concurrent activities, using the same synchronization mechanisms provided by CSP. An activity which waits on a channel for communication with another concurrent activity would be automatically suspended and its transactional locks released. Subsequently, it is resumed after it synchronizes. Effectively a received message on a channel causes the activity to resume its execution and to restart the transaction.

5 Conclusion

One can conclude that through the implementation of the Transactional CSP Processes framework, we have achieved the main objective of providing a solution that allows a long-lived transaction to have independent activities running concurrently. Support for concurrent activities provides more flexibility and better performance for long-lived transactions since activities running in parallel do not affect each other, thus avoiding scenarios where suspended activities cause a long-lived transaction to take a considerable amount of time to complete. The composition constructs $\text{SEQ}_{LLT}$ and $\text{PAR}_{LLT}$, introduced and implemented by the Transactional CSP Processes framework, allow an application developer to define the desired method of execution for the activities in a long-lived transaction. Synchronization and transactional issues are managed by the framework transparently from the application developer, allowing more dedicated time towards the business logic of the transactional application.

References

High Performance Staged Event-Driven Middleware

Alan Cassar and Kevin Vella

Department of Computer Science, University of Malta
{acas034|kevin.vella}@um.edu.mt

Abstract. In this paper, we investigate the design of highly efficient and scalable staged event-driven middleware for shared memory multiprocessors. Various scheduler designs are considered and evaluated, including shared run queue and multiple run queue arrangements. Techniques to maximise cache locality while improving load balancing are studied. Moreover, we consider a variety of access control mechanisms applied to shared data structures such as the run queue, including coarse grained locking, fine grained locking and non-blocking algorithms. User-level memory management techniques are applied to enhance memory allocation performance, particularly in situations where non-blocking algorithms are used. The paper concludes with a comparative analysis of the various configurations of our middleware, in an effort to identify their performance characteristics under a variety of conditions.

1 Introduction

In this paper, we investigate the construction of highly efficient and scalable staged event-driven middleware for shared memory multiprocessors. The staged event-driven architecture (SEDA) that we focus on is a design introduced in [Wel02] for developing massively concurrent services which behave well under heavy loading. Our multiprocessor implementations of SEDA middleware are designed to be highly efficient, through the use of a variety of event queue structures, and a range of access control mechanisms including non-blocking algorithms.

The paper kicks off with a brief overview of SEDA and possible event queue arrangements. We proceed by investigating a selection of our shared queue designs and multiple queue designs. Following this, we consider the memory management techniques used in our design, and our handling of blocking system calls. We conclude with a comparative analysis of the performance of the various configurations of our middleware.

2 Background

The following section gives a brief introduction to some of the techniques applied to our staged event-driven middleware, including scheduling techniques and performance enhancement techniques.
2.1 Staged Event-Driven Architecture

The staged event-driven architecture (SEDA) is a design introduced in [Wel02] for developing massively concurrent services which behave well under heavy loading. A SEDA application consists of a network of event-driven stages connected by event queues. Dynamic resource controllers are used to ensure the efficient behaviour of stages irrespective of loading variations.

2.2 Shared Run Queue

A technique widely used in scheduling algorithms is based on a shared run queue. Cordina [Cor00] chose this technique to implement a SMP version of the MESH user level thread scheduler, which the author called SMP MESH. Debattista [Deb01] experimented with this technique in his user level thread scheduler implementations. In [Vel98], a similar technique is used for an SMP implementation of KRc [WW96]. In a shared run queue environment, a single queue shared between all processors contains all the schedulable entities and each processor will have to “fight” to acquire one or more items from the queue. Since we have one queue shared between all processors, the bigger the number of processors, the higher the contention on the queue. This queue can easily become the bottleneck of the system especially when the number of processors is relatively huge.

2.3 Per-Processor Run Queue

In a per processor run queue environment, each processor has its own run queue where the schedulable entities are placed. The biggest advantage of this scheduler is that it does not matter how many processors are accessing the run queue, since each processor has its own, no synchronisation techniques are required to get exclusive access to the queue, hence there is no need to implement complicated and not so trivial lock-free, non-blocking or wait-free algorithms. Anderson et al. [ALL89] experimented with a per processor run queue thread scheduler, however the authors kept a global pool for the reason of load balancing. Their experimentation showed that their per processor scheduler performed better than a shared run queue scheduler, mainly due to no contention on a shared run queue and to the locality achieved in scheduling the same threads on the same processor [ALL89].

2.4 Locality and Load Balancing

Locality is the notion of keeping the processes as close to their data as possible [Deb01]. One of the best ways to achieve this is to exploit the processors’ cache. Schedulers which use per processor run queue technique tend to favour locality [Deb01]. On the other hand, shared run queue schedulers usually do not emphasise on locality. Some explicit techniques have to be applied such as batching [Vel98] or cohort scheduling [LP01].
Both load balancing and locality are key factors in performance, however, most of the time tend to be mutual exclusive. In a per-processor run queue, automatic locality management is provided, but when migration policies are applied to load balance the work, locality is lost. Shared run queue offer automatic load balancing, but special techniques have to be applied to maximise locality.

2.5 Batching

Vella [Vel98] argues that much of the performance enhancements that were being studied and evaluated for the KRoC scheduler focused on how to reduce the explicit cost of context switching, but no one actually tried to improve on other implicit costs such as the effect of cache memory, which can easily boost software performance when used correctly.

A technique called batching, which tries to maximise the cache locality in fine grained multithreaded systems, is introduced in [Vel98]. In batching, the run queue does not hold schedulable entities such as threads or processes. Instead, it holds queues of threads, referred to as batches of threads. When a processor requests an entity to be executed from the run queue, the processor is given a batch of threads, where each batch has a dispatch time much longer than the dispatch time of a single process/thread.

It is argued that if the batch size is relatively small with respect to the batch’s dispatch time, each thread inside the batch will be scheduled several times on the same processor (within the dispatch time of the batch), helping each thread to find most of its relevant data in the cache memory. If a batch contains a relatively small number of threads, these will not manage to push all the data of other threads from cache, improving performance drastically.

3 Shared Run Queue Middleware

The Shared Run Queue middleware embraces only one run queue for the set of available processors, each of them demanding work from the same shared queue. Once this middleware is launched, one thread for each processor is created and bound to a separate CPU. Figure 1 depicts the design of this scheduler.

Each processor is responsible of dequeuing an available stage from the shared run queue using some synchronisation technique. If the inbound queue of the dequeued stage contains executable events, the processor will remove a set of events, reappend the stage to the run queue, and execute the events as a batch in order to maximise both data and instruction locality. Stages with empty inbound queues are removed from the run queue and added only when their event queue is populated by at least one event.

Once the run queue is short of work, each processor will block after incrementing sleepCounter. This value is consulted every time a new stage is added to the run queue, and if greater than zero, a signal is sent in order to wake up the sleeping processors to resume their execution.
The shared run queue scheduler is configurable in several ways, the run queue and the event queues can be configured to use simple queue or dummy head queue. The locking mechanism is also configurable from the operating system-provided locks and our implementation of the test&test&set algorithm. Moreover, a non-blocking shared run queue implementation is also available.

3.1 Event-Driven Scheduler

In the event-driven scheduler, instead of scheduling stages, events are directly placed on a shared run queue, each processor dequeues these events and executes them as shown in Figure 2. This behaviour can be simulated by setting the batch size of the previously discussed scheduler to one, however, the above scheduler requires three operations to dequeue one event: dequeue a stage, dequeue an event and enqueue the stage back on the run queue, where this scheduler would require only one. The aim of this scheduler implementation is to prove whether stage batching is actually a better solution or not.

4 Per-Processor Run Queue Middleware

Every processor has exclusive access to an associated run queue in this scheduler implementation. Moreover, each processor has an associated migration queue used for migration of stages between the different processors in the system. Figure 3 shows the architecture of the Multiple Run Queue Staged Event-Driven Middleware.
Each processor will start off an execution cycle by moving any stage found on the migration queue to the run queue. Once the operation is complete, a stage is obtained from the associated run queue with no use of synchronisation and a set of events are executed as a batch. When no work is available both on the run queue and on the migration queue, the processor blocks waiting for new work to reach the migration queue, at which point it will be signalled by the thread which provided the work.

Several different per-processor run queue schedulers are implemented, each with its own characteristics. The following list gives a brief description of each scheduler.

**Basic Model** This is the most primitive per-processor run queue scheduler implementation. When a stage changes state from non-schedulable to schedulable, it will be placed on the run-queue of the processor which triggered the operation. Load balancing is an issue with this scheduler implementation.

**Maximising Cache Locality** This scheduler binds a stage to one processor where it will execute for its entire life. Cache locality is improved because each time an event is executed, it has a greater probability of finding necessarily data in the cache which was pulled in by the execution of similar previous events. This scheduler however tends to suffer from poor load balancing if the system is not designed well.

**Sender Initiated Migration** The processor having extra workload triggers the routine of sender initiated migration in order to try to move some of his extra workload to other less loaded processors. This routine is initiated by a simple counter and a threshold configurable by the user.

**Receiver Initiated Migration** This is triggered by the processor which is short of work, in other words, both the run queue and the migration queue are empty.

**Hybrid** A scheduler which applies both sender initiated migration and receiver initiated migration to keep load balanced.
The run queues, event queues and the migration queues can be configured as simple queues or dummy head queues. The locking mechanism used for the event queues is also configurable from, `pthread` locks, `test&test&set`, or non-blocking.

Fig. 3: Per Processor Run Queue Middleware

5 Memory Management

The memory management used in our Staged Event-Driven Middleware is very similar to the one presented by Valois [Val96] for the implementation of his non-blocking data structures. However, a race condition in the memory management of the same author was identified by Michael and Scott in [MS95] where a solution is also given which was carefully applied to our memory management module. Each block of memory to be reused has associated a reference count integer.
value referred to as refCount. This has a dual purpose: to store the number of references pointing to the block in question multiplied by two, or to contain the value of one to denote that the memory block is deleted and ready for reuse. Memory blocks which are safe to be freed without causing the ABA problem [Val96] are actually never deleted using the free system call. Instead, a stack (called the free stack) is defined which keeps track of all free blocks in the system which can be reused by the application whenever required.

If a pointer $p$ is referencing a particular block, whenever a function will need to read or modify that location, it does not simply copy the value of $p$ into a temporarily pointer and perform the required operations. Instead, it will need to safely read that block by first incrementing the reference count, followed by the reading or modification of the contents of the block. When all operations on that block are ready, the process cannot plainly continue its execution without first releasing that block by decrementing its reference counter. Furthermore, if the reference count results in the value of zero after being decremented, that block must be freed and returned to the free stack to be reused by other operations.

One can easily see how the malloc and free system calls are avoided with this solution, but how is the ABA problem solved? If processor $n$ has a reference to block $p$, processor $n$ is guaranteed that any operation can be carried out safely on block $p$, even a compare&swap. This statement is supported by the definition of our memory management. When processor $n$ obtains a reference to block $p$, the reference count of $p$ will be incremented by two. Since in our memory management blocks will only be freed when their reference count reaches zero, the safe read executed by processor $n$ on block $p$ halts the reference count to reach zero, and never freed. If the same block is not reused, the ABA problem cannot possibly arise. This is the same solution used by Valois [Val96] to avoid the ABA problem in his non-blocking data structures and modified by Michael and Scott [MS95] to eliminate the race condition determined by the same authors.

6 Blocking Operations

In an event driven system, event handlers are executed until completion. Since only one kernel thread per processor is used by the middleware, blocking one of these threads stalls a processor until the blocking call returns. Most I/O operations present such problem including file I/O and socket I/O. In this section we present the approaches taken to solve this issue and discuss the methods applied to integrate for both file I/O and socket I/O in our middleware.

6.1 File I/O

Since we are aiming to achieve asynchronous I/O operations in order not to block our scheduler threads, why not use the technology provided by the operating system itself? The Linux kernel 2.5 and higher provides the support for asynchronous non-blocking I/O operations which they call AIO.
Welsh in SEDA [Wel02] wrapped each I/O call in a SEDA stage [Wel02], and in our staged event-driven middleware, we also prepare the main file I/O operations in stages. When the user requires a file read operation, a file read stage has to be created, where only one stage is required for any number of operations. The operation starts off by sending a special event data structure to the file read stage which includes data such as the file descriptor, the stage to be notified when the operation is completed, and other important information. The file read stage immediately starts off an AIO read operation which upon completion executes a call-back function. The job of the call-back function is that of notifying the appropriate stage that the operation is complete by handing over the data read from the file (or any errors which occurred during the operation). The file write operation is handled in almost the same way as the file read operation.

6.2 Socket I/O

At the time the staged event-driven middleware was being implemented, the socket AIO system was not a standard feature of the Linux kernel. Due to this fact, it was decided that the socket I/O had to be implemented using a different approach from the one used by File I/O.

The model used resembles the solution used by the Flash [PDZ99] web server. For a blocking operation, Flash uses a helper process which blocks until the I/O event is completed, and notifies the original process when such operation completes. Instead of using a separate process, we opted for a thread pool from which a thread is acquired and handled the blocking operation to be performed. Furthermore, Flash [PDZ99] uses IPC channels for communication between the main polling loop and the helper processors which entails a system call each time an event is completed. In our middleware implementation, when a blocking operation terminates, the stage to be notified is sent an ordinary event just as any stage would send an event to any other stage in the system in order to communicate, and everything is done in user space.

7 Performance Testing

The following section presents some tests which were performed in order to compare the different schedulers implemented throughout this project. All tests were performed on a dual Intel Xeon processor machine supporting the Hyper-Threading technology, each CPU running at 3.80Ghz with 2MB of level 2 cache per processor. This Intel processor family supports the EM64T (Intel Extended Memory 64 Technology) architecture and was running an x86-64 operating system, a Linux RedHat with kernel version 2.6.9-42.ELsmp. All memory allocation was done apriori of the start of the tests to avoid system calls.

7.1 Queue Comparison

The following test was performed on a synthetic benchmark where a total of 502 stages had to be scheduled. The job of each event handler is that of reading 8
bytes out of a 32-byte cache aligned data structure and writing 4 bytes into the same structure. The scheduler was configured as an event-driven scheduler and the results can be seen in Figure 4.

Our test&test&set implementation performed better than the operating system provided locking mechanisms in both the coarse grained locking queue and the finer grained locking queue. Moreover, the dummy head queue performed better than the coarse grained locking queue due to the separation of the enqueue operation from the dequeue operation. The non-blocking queue was outperformed by the test&test&set implementation, however we believe that with a larger number of processors available, the non-blocking queue would have performed better than any other implementation available for the event-driven scheduler.

![Event Driven - Queue Types](image)

Fig. 4: Event-Driven Scheduler — Different Event Queues

### 7.2 Batching

The event-driven scheduler never exploits batching, therefore the time taken for the test to complete remains constant independently of the batching value. It is however more efficient than the staged shared run queue scheduler with the batching size set to one due to the fewer number of queue operations performed to execute an event.

With a low batching value, the per-processor run queue scheduler outperforms the shared run queue scheduler by almost 6 seconds, and their are two main reasons for this behaviour, the first one being the contention on the shared run queue with a low batching value. Cache locality is the second reason. Binding stages to the same processor helps in decreasing the total amount of cache misses issued during the execution of the application. As the batching value is increased, both schedulers’ performance increases.
8 Conclusion

We have presented a framework which is highly configurable and general purpose, usable with any staged event-driven application. We have also shown from the results obtained that our middleware implementation is scalable and efficient. File I/O is also integrated within our middleware with the help of the operating system provided AIO while socket I/O is available through the help of thread pools.

We have also conducted a series of performance tests to identify which scheduler performs the best, but we can conclude this document by noting that each application has different requirements and characteristics, and will perform differently on different scheduler implementations.

References


KIKI — A Key to the Integration of Knowledge and Innovation

Abigail Cauchi, Jennifer Fenech, Karl Fenech, and Luana Micallef
Departments of Computer Science and Artificial Intelligence, University of Malta

Abstract. The proliferation of ICT within the educational domain is serving to overcome several barriers associated with traditional pedagogies. However, the challenge of balancing educational objectives against technical limitations and harsh financial realities is becoming more relevant than ever. Specifically, one is often faced with insufficient funds for hardware resources, lack of streamlined distribution mechanisms for software, and irreconcilable disparities in the packages offered.

In this paper, we present KIKI as a tentative solution to the aforementioned obstacles. KIKI is a prototypical system devised following extensive research on learning paradigms, including collaboration and the use of games within the educational context. It is meant to serve as a platform for deploying educational software through an extensible architecture which provides inherent support for MultiPoint functionality, inter-computer communication, user identification, and progress tracking. Seamlessly integrated, it would bind all stakeholders (developers, administration, teachers, and students) in their respective roles, amplifying the dissemination of knowledge and providing enhanced educational opportunities for all, irrespective of age and financial conditions. The system would also enable an innovative edge, giving unbounded opportunities for the development of applications to best meet the local demands.

1 Introduction

“Education provides people with the tools and knowledge they need to understand and participate in today’s world. It helps to sustain the human values that contribute to individual and collective well-being. It is the basis for lifelong learning. It inspires confidence and provides the skills needed to participate in public debate. It makes people more self-reliant and aware of opportunities and rights.” — UNESCO [UNE04]

Education is one of the fundamental human rights, and its beneficial role in society is undeniable. Providing education for all is one of UNESCO’s goals, aimed to be accomplished by 2015. Unfortunately, providing education for all is not an easy goal to achieve due to various difficulties in today’s world. As indicated by UNESCO Bangkok [UNE07], one of the most difficult challenges is that of balancing educational objectives with technical limitations and harsh
financial realities. U. S. Pawar et al. [PPT06] report a typically high student-to-computer ratio in most developing countries, stating that it is “not unusual” to have up to ten children on a single computer. One must also consider the challenge of keeping the student engaged — the process of learning requires that the student, no matter what age-group he/she belongs to, is actively involved in the learning activities he/she is participating in. Losing the student’s interest in the educational activities one is offering renders the learning process ineffective. Through ICT, one can transmit knowledge by presenting educational content to the student in the form of multimedia. Audio-visual lessons and games help in motivating the student’s interest; games in particular keep the student absorbed because of their competitive nature, especially when human opponents are involved.

“Picture this: In a classroom of 40 children with only four PCs among them, 10 students crowd around each machine. Within each group a dominant student — often the brightest, richest, or oldest child — takes center position and controls the mouse. While other students point, gesture and vie for control of the mouse, they ultimately have no direct control of the PC and often lose interest and shift their attention elsewhere. The child with the mouse is learning on his own, and the others are not learning at all.” — Microsoft [Mic06]

The distribution problem arises when it comes to installing, maintaining, and updating the software in all the educational institutions. Even if it were possible to supply a computer per student, how would the commissioning entity be able to distribute up-to-date educational software for each student in a cost-effective way?

Providing education for all is a great challenge, and success will give millions more skills to rise out of poverty. We aim to develop a complete system which is extensible and, at the same time, easy to use and implement.

1.1 Collaboration

There are many advantages in letting students work together which are directly related to the self-improvement of the student, both as an individual and also concerning how he/she works with his/her peers.

By using a technology which allows multiple input devices on one computer, it is possible to have several students participating actively on a single computer at the same time — this addresses the high student-to-computer ratio and encourages collaboration by allowing the students to learn together without having one particular student at a physical advantage over the others. Studies by K. Inkpen et al. [IBGK95] have indicated that the performance of students collaborating on a single computer using multiple mice would exceed that achieved by the same students working alone.
1.2 Educational Games and Multimedia

Researchers and teachers are realizing the potential of using games for educational purposes. Research into the use of mainstream games in education is relatively novel, but growing rapidly. A broad literature review of the topic is given by J. Kirriemuir and A. McFarlane [KM04]. Gender-specific studies, on the other hand, have been carried out by J. Lawry et al. [LUK+95] and K. Inkpen et al. [IUK+93] for boys and girls respectively.

The challenge for developers and designers, possibly with the help of teachers and researchers, is to create educational games which have the same level of engagement as games used for purely entertainment purposes — educational games which challenge the player and make him/her want to play (and consequently, learn) more.

By using new technologies, designers and developers can produce interactive educational games easily, and hence help in making the learning process more fun and engaging.

2 KIKI — System Overview

A solution to achieve UNESCO’s EFA (Education For All) goals is a fully-integrated MultiPoint system which is easy to use, extensible, and most importantly, promotes a constructivist pedagogy and an intuitive way for teachers to make their lessons more interactive, interesting, and collaborative. Through our proposed system, students may be assessed whilst engaging in educational software activities, enhancing their motivation. State-commissioned or commercial developers may upload new applications which would be seamlessly incorporated into the system and immediately available within all classrooms. We aspire for our system to serve as a Key to the Integration of Knowledge and Innovation (thus the name KIKI).

The system provides four types of clients, respectively designed for the application developers, the school administration, the teachers, and the students. The administration client provides a user-friendly interface for the members of the school’s administration to insert or update details pertaining to the school’s students, teachers, and classes. The teacher and the student clients are more oriented to the actual classroom scenario. Login is done through the use of virtual cards, and applications can be downloaded and run during the lesson with utmost ease. In this way, the teacher does not need to prepare any exercises, nor be an expert in computing. The student client is MultiPoint-enabled and, thus, introduces the idea of competition and collaboration among students residing on the same or different computers whilst, at the same time, allowing all the students to participate and control their own mouse. Moreover, the clients are localizable such that they can be modified to support any language.

The system is flexible and extensible enough to support any type of application or tool that is useful for both the teacher and the students, giving developers a range of options to choose from, such as MultiPoint-enabled or single-point
applications, .NET 3.0 or .NET 2.0 applications, and providing a teacher and/or a student mode. Tools such as slideshow designers would be particularly useful to teachers.

### 2.1 Conceptual System Layers

The high-level design of the system was inspired indirectly from the Open Systems Interconnection Basic Reference Model (more popularly known as the OSI Model). The system is divided into layers (abstraction levels), with each layer building upon the lower layers to provide a set of functionality for use by the higher layers.

![Fig. 1: The six conceptual system layers](image)

The **Underlying Technologies** layer covers all the requirements necessary for the system to run. The central authority and the schools’ administrations should ensure that the computers on which the servers and clients will be installed meet the specified system requirements, and that the deployment procedure is completed.
The Core Technologies layer provides the enabling framework which empowers all stakeholders to participate accordingly in the system. It exposes backend functionality (such as database access and inter-computer communication) through a series of WCF services to which the clients subscribe.

The Clients layer serves as the front-end for the system, and provides for user interaction such as log-in, information retrieval and presentation, and submission of data or results. It also defines an API which would allow independently-developed applications to integrate with the class clients, taking advantage of system-provided features such as student authentication, progress tracking, and inter-computer communication.

Development of the Applications layer is principally intended to be commissioned by the central authority. Applications may be uploaded to the central repository through the developers client, and will be deployed on the class clients automatically through the system architecture. The scope of applications may range over educational games, animated tutorials, homework exercises, and tools such as dictionaries and visualization aids. Each application may also be made customizable using swappable XML content files.

The Deployment layer involves the fruition of the system by the individual schools and teachers. It covers the following aspects:

- propagation: sending the selected application to connected student clients
- customization: choosing the content file to be used in a particular application
- utilization: making use of the provided tools, e.g. for constructing visual tutorials

Finally, the Experience layer concerns the students’ enhanced educational venture. Students may engage in educational games selected by the teacher, watch videos prepared by a slideshow designer, or make use of any of the available applications subject to teacher control.

3 System Architecture

The central server (as displayed in Figure 2) is maintained by the central authority, and hosts the central database and the application repository. The central server also provides two portals; one for developers, and the other for schools. The developers portal allows application developers to log onto the central server, upload new applications to the repository, download and enhance existing applications, or provide new content files for customizable applications. Since the developers portal is a WCF service, it is possible for developers to access the system remotely, thereby giving them the opportunity to work from the comfort of their own homes, or even from abroad.

Each school would, in turn, maintain a school server which hosts the school database and the repository cache. The school database holds details pertaining to all its students, teachers, and classes. The school server also hosts a number of school-based services used by the administration and class clients, including the central portal which is used to access the central server (over the internet or...
Fig. 2: System Architecture
through a wide-area network set up by the central authority) for the retrieval of applications from the central repository.
The school server would be connected to the school local-area network, enabling its services to be used by the class clients deployed on the class computers. Such services include:

- retrieving student, teacher, and class information from the school database
- recording student progress to the school database
- setting up a peer-to-peer inter-computer communication mesh amongst a given classroom’s computers
- downloading applications from the central repository through the central portal

It is worth pointing out that the entire system is integrated in such a way that all the above interactions take place seamlessly and with minimal human intervention. Suppose, for example, that an application developer has completed a new application, and uploaded it to the central repository using the developers client. Almost immediately, the new application is available for deployment in all classrooms connected to the system, without needing any installation whatsoever on the class clients or even the school servers.

4 Evaluation and Conclusion

In order to evaluate the capabilities of our prototypical system, we ran a demonstrative deployment with the help of a class of thirty students, aged between nine and ten, from a primary state school. The setup consisted of eight computers for the students, each having three to four mice, and a computer for the teacher. Each student individually controlled a mouse to which a cursor with a specific unique color was associated. The computers, on the other hand, were assigned a unique name corresponding to the soft-toy placed on the top of the monitor.

In preparation for this trial, we uploaded a number of sample applications which we had developed onto the central repository. The teacher was allowed to select any of the applications from the available list; once downloaded from the repository, all the thirty students in the class could participate in the activity. The sample applications, together with the rationale behind their conception, are described hereunder:

*ImagineAMooVee* is a tool developed for teachers to be able to build educational slideshows easily. The main idea behind this application is that visualization is the key for making learning easier. Rich Mayer, a cognitive psychologist, has developed a cognitive theory of multimedia learning, and has proposed seven research-based principles for how to design learning environments [May05]. Our idea follows his first principle, *The Multimedia Principle*, which states that “Students learn better from words and pictures than from words alone.”

*WhiteBoard* and *PaintingProg* follow the same principle. *WhiteBoard*, as the name suggests, acts as a virtual whiteboard which the teacher can use to visually illustrate concepts to the students while explaining. *PaintingProg* works on the
same idea as WhiteBoard, but instead of having the teacher as the centre of attention in the class, we have the students trying to explain words to each other through illustrations by means of an engaging Pictionary-like game which encourages collaboration and competitiveness amongst the students.

MathBalloonGame, Multiples, and Quiz are games which students can choose to play by using either a cooperative or a competitive approach. For educational games to be effective, they need a high level of engagement, as explained by J. Kirriemuir and A. McFarlane [KM04]; therefore, these three games were designed to be MultiPoint-enabled and given a colorful theme. MultiPoint also brings about the collaborative aspect which makes the educational experience more compelling, as shown by K. Inkpen et al. [IBGK95].

4.1 Closing Observations

The competitive aspect brought about by this fresh pedagogy was very successful at keeping the students engaged in the activity. They were absorbed throughout the lesson, with a constant display of enthusiasm towards the new system.

Students were not confused about the idea of multiple mice and cursors. They were not distracted or talkative. Instead, they collaborated and cooperated together so that every student was able to participate in the competitive games and enjoy this new idea of a classroom. Becoming a collaborator and no longer a dictator, the teacher was able to easily control the entire class and spend more time monitoring and analyzing the performance of the students with the help of the real-time progress-monitoring system, which works transparently behind
the applications. The sample applications illustrated the possibility of having a wide variety of activities that can be followed in a class. Notwithstanding the favorable response exhibited towards the system during the trial, one should bear in mind that this was meant primarily as a demonstration, and should by no means be considered an authoritative or conclusive evaluation of the system’s effectiveness in the long run. A more thorough study would need to be extended over the course of an entire academic term or year, incorporate applications offering a broader and deeper coverage of the curriculum, and objectively measure the performance of the participating students against a control group of similar ability (not exposed to the system) by means of common assessments.

Finally, the trial served to demonstrate that KIKI is capable of successfully overcoming the obstacles mentioned earlier. For merely €4 per mouse, we managed to actively involve a class of thirty students on just eight computers. By uploading the developed applications to a server connected on the school network, we eliminated the need of having to install them manually on each computer. We also ensured that the applications took advantage of our system-provided capabilities, such as MultiPoint functionality, inter-computer communication, user identification, and progress tracking, thereby providing a smoother and richer experience for both students and teacher alike.

References

KIKI — A Key to the Integration of Knowledge and Innovation


Kanban Scheduling System

Christian Colombo and John Abela
Department of Artificial Intelligence, University of Malta

Abstract. Nowadays manufacturing plants have adopted a demand-driven production control system also known as just-in-time production (JIT). This greatly reduces inventory costs because the buffers between the processes act as a blocking mechanism to indicate when production should stop and when eventually it can continue. One implementation of a JIT system is the Kanban system. Manufacturing in a JIT fashion, we face the challenge of flexibility to respond to changes in the customers’ demands, while at the same time remaining cost-effective. This work attempts to introduce automation to the process of Kanban scheduling in a manufacturing plant environment with a multi-level part type. The proposed solution is a Memetic Algorithm (MA) which tries to optimize schedules such that they meet the deadlines of customer orders without causing buffer overflows, while at the same time keeping setup time to a minimum and placing free periods sensibly if these occur. The evaluation shows that this was achieved and the MA gave better results than the manual system currently in use.

1 Introduction

1.1 Overview — The Kanban System

The Kanban production system is a “pull” system where production takes place if, and only if, there is demand. The advantage of the system is that the stores (buffers) are virtually eliminated and the production becomes more responsive to the customers’ demands.

Figure 1 shows how a customer’s demand propagates through the production process. When an order is received from the customer, Process 3 starts to produce the finished goods (using the items from the upstream buffer). When a container of the items used is emptied from the buffer, it is returned to the Process 2 to refill it. Similarly, Process 2 starts to produce, using the items from the subsequent buffer. Again, the empty containers are returned to the previous process. Each container returned from a process to the previous has a “Kanban” associated to it which literally means a “card signal” in Japanese. The Kanban system was in fact introduced by Toyota in the 1950s. The term was coined for the fact that the Kanban actually “signals” a process to start production. [Jap89,Mon97a,JIN,Ols,MES01]

However, if each process in the production line starts to produce as soon as it receives an empty container, most of the time will be lost in setting up machines to produce the various items for each empty container. In a real-life situation,
this problem is tackled delaying the start of production until a number of kanbans of the same item are received. Then, the setup-time needed becomes small (or acceptable) in relation to the run-time of the production. This also means that we should have buffers which are large enough to at least store the number of items produced from one runtime of the machine. Therefore the problem of scheduling kanbans is an optimization problem with aim of finding the best balance among the various costs which production entails [Mon97b].

1.2 Aims and Objectives

**Main Aims** The aim of this project is to develop an algorithm for automatically generating a schedule for the kanbans in a manufacturing plant. Furthermore, the project will help us study which factors in the manufacturing process attribute to higher production costs. This will help us generate an optimized schedule, trying to reduce the cost per item produced. The main aim of the project is to create an automated Kanban scheduling system which performs better than the manual system which is currently used. This will entail the creation of a model as close to reality as possible. Subsequently, an algorithm will be developed which, given a data set will produce an optimized schedule. The schedule should be optimized with respect to various aspects. These include the setup time, the buffer balances and the free periods in the schedule.

**Secondary Aims** Without an adequate interface, it will be difficult to show the inputs, processing and results to the user. Therefore, a secondary aim is to create a verbose interface which allows the user to enter the necessary data, show the algorithm running and finally output the optimized schedules created. Another secondary aim is to test the algorithm with different parameters to find out the best settings to be used in practice. Furthermore, testing should be carried out to show that the automatic system performs better than the manual system currently in use.
2 Literature Review

2.1 NP Hard Problems

It is well known that scheduling problems are very challenging computational problems. However, real-life applications include a lot of different scheduling problems. These include school timetabling system, workers' roster and flight scheduling. Basically the scheduling problem is a search problem. In other words solving the scheduling problem can be described as searching through the space of all possible schedules and selecting the optimal schedule. The problem is that the number of possible schedules is usually exponential in relation to the size of the input. The problem is further complicated because the search space is not ordered and we may find one of the many local-minima, but we may still be far from the global-minimum [GJ79,Mit98,Mos89].

2.2 Heuristics to Solve NP Hard Problems

When tackling such NP Hard problems, rather than trying to obtain the optimum solution (which may take millions of years to find), we try to find a “relatively good” solution which will suffice for our real-life applications. There are many of these techniques which have been tried out successfully. Memetic Algorithms are one such example of successful techniques which have been implemented to solve (even though not optimally) a wide variety of scheduling problems [GJ79,Mos89].

2.3 Genetic Algorithms

Genetic Algorithms are algorithms inspired by natural evolution. The idea is that we try to emulate the process of natural selection to come to a better solution to a given problem. Therefore, we start with a population of individuals, all having different fitness (The fitness is a measure of how much the individual possesses the desired characteristics). Then we generate “new” individuals based on the existing ones by using genetic operators such as mutations and crossovers. Once these are generated, we choose which individuals will be kept for the next generation and the whole process is repeated again and again until there is sufficient increase in the fitness of the population [GJ79,Mos89].

2.4 Memetic Algorithms

Memetic Algorithms are an extension of Genetic Algorithms, where we keep the same idea, but we try to improve the search of solutions by limiting the random element of the search and trying to find local maxima by using appropriate optimization techniques. In other words, we try to improve the individuals of the population, not simply generate the individual using the genetic operators. This method will generally make the search better, because now the individuals will all represent a local minimum in the search space, not just any possible point of the space [GJ79,Mos89].
3 Design

3.1 Generating Random Schedules

The initial step of a basic evolutionary algorithm is to generate a population of new random individuals (in our case schedules) [Mit98]. Generating random schedules is not an easy task since we are trying to emulate the real-life system where a number of autonomous “entities” are acting at the same time. These “entities” are actually the different sections which are triggering each other to work: starting from the last section till the starting section. It is important to note that we should not issue all the Kanbans from the beginning of the scheduling process, but rather let the Kanbans be issued automatically once there is space in the buffer.

Using this system we are automatically “going down” the bill-of-material tree. This is exactly what happens in a real-life situation [Mon97b] and is further explained in Figure 2.

![Fig. 2: Withdrawal cycle](image)

3.2 The Building Blocks of a Memetic Algorithm

When designing a Memetic Algorithm (or a genetic algorithm), the genetic operations should be designed with great care, since these are used in each generation [Mit98, Mos89] and hence are crucial in the evolution of the population. Another design aspect of any genetic algorithm is the choice of a fitness function [Mit98]. It must be ensured that this function includes all the aspects that we need to
optimize. If an aspect is not included in the function, the individuals cannot not be optimized with respect to that aspect. The following sections will describe the design of the fitness function and the operations used for the Memetic Algorithm.

3.3 The Fitness Function

The aspects considered in the fitness function are the following:

1. The total setup time used by the machines.
2. A penalty is also given to each time unit during which a buffer is an invalid state (either over the maximum or below the minimum). However, in this case the penalty also depends on the amount by which the buffer is invalid.
3. Furthermore, an extra penalty is assigned for the amount by which buffers are still in an invalid state at the end of the schedule generated.
4. Another part of the penalty is assigned for the number of times a machine starts and stops. The reason for this penalty is that in practice it is undesirable that a machine starts and stops with free periods in between.
5. Another penalty is assigned when the machines of a section do not start or stop together. From a practical point of view [Mon97b], it is undesirable that there is only one machine working in a section when there are three machines, which can therefore reduce the production lead time.

3.4 Mutations

The designed mutation can use one or more genetic operations for any number of times. The operations designed to be used by the mutation may be one of the following:

1. The first operation is to swap two jobs in one section. For example: a job on machine m1 at time t1 is swapped with machine m2 at time t2. The purpose of this operation is basically to try to reduce the amount of setup time for the machines.
2. The second operation also swaps a job, only that this time not with another job but with a free period. This operation will help in removing the unwanted free periods which are in the middle of jobs in a schedule.
3. Another operation is the complete removal of a job from a schedule. Sometimes it is better to reduce the production to eliminate buffer overflows and this operation in fact simply removes a job to reduce production.
4. The fourth operation is to insert a free period before a job. The purpose of such an operation is to increase the probability of having schedules which start the jobs of a section simultaneously at the same time.
5. Similar to the previous, this operation simply removes a free period from the schedule. The purpose is to reduce the number of times the machines are stopped and restarted, and may also help to start the machines in the section at the same time (as in the previous point).
3.5 Local Search

Local search is an optimization done on an individual of the population. There are two strategies which aim to improve different aspects of the schedules.

**Strategy 1 — Reducing Buffer Penalties** When we analyze a buffer to calculate the penalties of overflow or underflow, we can deduce which production jobs should be moved earlier or later to eliminate the penalties. The idea is that if we have encountered an underflow earlier, then the jobs creating the overflow should have occurred earlier; they have been "delayed". Inversely, if we meet an overflow which was not preceded by an underflow, then the jobs should be “moved” to a later time; they are “early”.

**Strategy 2 — Reducing “Imbalanced” Sections Penalties** The second strategy has the aim of reducing the number of imbalanced schedules of sections. This is done by examining the jobs which are being produced “alone” i.e. no other machine in the same section is working while that job is being carried out. Such jobs should be re-scheduled somewhere else where there are more machines at work to avoid sections working inefficiently.

3.6 Crossovers

Two types of crossovers are proposed: one is purely random while the other tries to take the best out of the two schedules randomly selected from the population.

**Random Crossovers** The random crossover generation works by selecting two random schedules from the population (after selection has taken place) and then for each machine, randomly decides whether to use the first or the second schedule. Once this is completed, we need to re-generate the buffer balances since neither the buffers of the first schedule nor those of second schedule, can be used as the buffers of the newly generated schedule. The purpose of using this purely random approach is that sometimes it is difficult to know which parts of which schedule will yield an overall better fitness. Hence, this random approach may sometimes give good results depending on chance.

**“Intelligent” Crossovers** The other type of crossover attempts to take the best out of the two randomly selected schedules. This is done by treating a schedule on a “section-by-section” basis, and calculate each section’s penalty. Then, we simply select the sections from the randomly selected schedules with the least penalty to form a new schedule with some sections possibly from either parent.
3.7 Method of Selecting Individuals

One of the main genetic operators is “selection” i.e. the way the individuals are selected to be allowed to remain in the population and generate offsprings [Mit98]. From the selection strategies listed in the literature ([JIN] p.166) we will choose a combination of the “Tournament Selection” method and the “Elitism” method. The purpose of using the “Elitism” is that it allows us to “secure” some of the best individuals of the population while that of using the Tournament selection is that is allows some of the non-best individuals to pass to the next generation so that we keep more “variety” in the population.

3.8 The Memetic Algorithm as a Whole

The Memetic Algorithm starts by first creating an initial population of random schedules. Then, a generation starts; each generation consists of the following summarized steps:

1. Apply genetic operators of mutation and crossover to produce new individuals.
2. Apply local search to improve individuals.
3. Use the selection algorithm to select which individuals will be kept to the next generation.

4 Implementation

The architecture selected for the implementation of the algorithm is the .NET framework using the C# language. The object-oriented design together with a user-friendly graphical user interface where implemented. Furthermore, the designed algorithms (described in the previous section) were implemented successfully. The following is an overview of some important implementation details.

4.1 Schedule Structure

There are two main parts of a schedule: (1) the machines’ schedule and (2) the buffers’ schedule. The machines’ schedule will contain the sequence of jobs which each machine will process throughout a period of time. On the other hand, the buffers’ schedule contains the activities of the buffers. This is obviously related to the machines’ schedule since the material is needed for production (and hence withdrawn from the buffers) and products are placed in the buffer as a result of production (hence inputting the products in the buffers).

The machines’ schedule is stored as a list of periods in an array. Therefore in order to schedule a new job, we must find a free period, update the affected transitions and insert free periods if the original free period is longer than the required time. This is shown in Figure 3.
5 Testing and Results

Testing was performed on data as close to reality as possible. An example of the results obtained is shown in Figure 4.

One should note that the schedule contains minimal setup time for machines (denoted by the pink (dark grey) boxes) and all the rest of the time units are allotted to jobs (white boxes). The green (light grey) boxes denote a change of shift. This means that the machine time is being efficiently utilized. At the same time the scheduling system also generates the schedule for the buffers. This shows what products are entering and leaving the buffer at any time. A sample is shown in Figure 5.

In this case one should note that all the boxes are white or light purple (very light grey) (except for the green boxes (light grey) which denote a change of shift). This means that there are no buffer overflows and even more importantly no buffer underflows. In other words, all the deadlines have been met without wasting material by leaving it idle in the buffers with the possibility of not finding demand for it.

When the result obtained by the Kanban scheduling system was compared to a typical result obtained by the manual system it was found out that the automated system performs better than the manual system. The main advantage of using the automated system was that all the orders were fit in the schedule, while the manual system was unable to fit in all the orders in time. However, it was noted that the manual system may do better in terms of setup time used. Nevertheless, considering the overall penalty, the automated system performed significantly better. Furthermore, one must consider that the manual system requires a lot of human effort while in the case of the automated system the effort is done by the computer.
Fig. 4: A sample of a machine’s schedule obtained

Fig. 5: A sample of the buffers’ schedule
6 Conclusion

We conclude that Memetic Algorithms are very much suited for real life optimization problems such as the Kanban scheduling problem. In the current project the problem was limited to certain boundaries such as no machine maintenance, which in practical terms cannot be overlooked. However, the proposed solution and the tests carried out show that there is a great potential in using evolutionary algorithms in helping schedulers to produce more cost effective schedules in a manufacturing environment.

References


Aspect-Oriented Programming
Runtime-Enforcement of Temporal Properties in Security-Critical Software

Christian Colombo and Gordon J. Pace

Department of Computer Science, University of Malta
{gordon.pace|ccol002}@um.edu.mt

Abstract. The Aspect-Oriented Programming paradigm has been advocated for modularisation of cross-cutting concerns in large systems. Various applications of this approach have been explored in the literature, one of which is that of runtime-verification based on assertions or temporal properties. Manually weaving temporal properties to ensure correct execution into a large code base is difficult to achieve in a clean, modular fashion, and AOP techniques enable independent specification of the properties to be automatically woven into the code. In this paper, we explore a number of applications of AOP-based runtime-verification with an emphasis on security-critical system development. Apart from weaving properties into existing programs, we show how related techniques can be used to approach security issues separately from the functionality of a module, allowing for better design of the actual system. Also, we explore AOP as a way of automatically ensuring that reusable code in a library is temporally correctly employed. An area in which not much work has yet been done is that of the use of AOP for runtime-verification of real-time properties. In our case studies we explore real-time issues and outline a proposal for automatic translation from real-time properties into code using AOP techniques.

1 Introduction

In designing software systems we would like to separate all different concerns into different modules, making the system easier to understand and maintain, making the code more reusable and more manageable for developers [KLM+97,BS06] — all advantages of good modular design. However, a problem sometimes encountered when using commonly used programming paradigms, is that a number of our concerns may not be possible modularised with the rest of the system [KLM+97,KHH+01,MKL97]. For example consider the design of a server application shown in Figure 1.

It is natural to choose to split the server into two main modules: one which validates the incoming connections and one which serves the existing connections. It may also also natural to split further the module serving requests into sub-modules, where each module would be responsible for the servicing of a particular type of requests. One may also envisage a separate module which
caters for the accounting of the server application. However, usually, statistics are gathered for accounting from all the modules of the system, spreading out use of the accounting module across the whole code. A more serious concern, is that changes to what information is logged will require maintenance of code across the whole system, and not just the accounting module. Similarly, security concerns usually cut across various of the modules of the system. With security concerns, the situation is worse than with most logging requirements, since one usually wants to identify not single events and function calls, but sequences of events which may originate from separate modules. Hence, one is faced with the same problems of unstructured code. These concerns, called aspects in AOP jargon, are said to cross-cut the system’s underlying functionality. The motivation behind AOP, lies in providing modular means of introducing such aspects into a system. Subsequently, the aspect code is automatically woven into the other modules at the points at which they cross-cut.

In this paper, we explore applications of AOP-based runtime-verification with an emphasis on security-critical system development. Using a simple server application case study, we show how AOP can be used for explicitly stating the properties of an existing system without modifying the system itself, and running these property monitors automatically together with the system, thus assuring that the properties are adhered to at runtime. Another technique we explore is the separation of security features from the functional core which may assume well-behaved communicating partners. We show how the development of a security-aware system can be split into (i) the development of a naïve system which presumes non-malevolent users; and (ii) the independent identification of security features which are then woven automatically into the functionality of the naïve module, thus allowing for better design of the complete system. Finally, we also explore AOP as a means to automatically ensure that reusable code in a library is employed correctly by having a separate contract specifying the correct use of the code in the library, including temporal dependencies. We start by exploring these different applications looking at temporal dependencies — with properties such as ‘method `connect` cannot happen twice in a row with-
out a call to disconnect in between’. We then extend this reasoning to deal with real-time properties, such as ‘no more than 10 calls to connect can appear every minute’. In both cases, we review and propose techniques using which such properties can be automatically incorporated using AOP techniques.

In Section 2 we give a brief review of AOP together with an overview of AspectJ as a prominent AOP technology. Furthermore, we give an overview of runtime verification and various flavours of temporal properties. Thus we explain the motivation behind the proposed AOP approaches. In Section 3 we describe three case studies in AspectJ, illustrating the use of AOP techniques for verifying various temporal properties at runtime. Similarly, in Section 4 we give another three case studies but this time with real-time properties. In Section 5 we give an outline of the proposed framework for specifying real-time properties using two examples from the case studies. Finally, in section 6 we conclude the work and propose future work.

2 Background

2.1 Aspect-Oriented Programming

As already briefly mentioned, AOP enables the programming of cross-cutting features in a system in a modular fashion. The AOP modules can be seen simply as other system modules but whose semantics corresponds not to actual code, but essentially as functions transforming, or modifying the rest of the code.

Different variations to AOP have appeared in the literature, depending on the different level of abstraction at which the aspect code is meant to be inserted in the code. For example some applications may need to work on low level information of method calling [MKL97] while in many other cases a higher level abstraction is more appropriate [SB06,KLM+97].

AspectJ is one of the more popular AOP languages [SB06]. It is an aspect-oriented extension to JAVA [KHH+01] originally developed by Xerox PARC to be a general purpose AOP language [SB06,LK98]. AspectJ provides a number of constructs which allow the developer to specify a variety of joinpoints within the code. For example joinpoints (where aspect code can be inserted) can be specified at the beginning or ending of method calls. Similarly, we can specify joinpoints just before or after a field access in a class. There are also other interesting constructs such as cflow(). This particular construct allows the developer to program interesting code such as do something when a particular recursive method is called but not when it is called within itself [KHH+01,SB06].

Of particular interest is that AspectJ offers two types of crosscutting: dynamic crosscutting and static crosscutting. While dynamic crosscutting allows the user to alter the execution of a system through the use of aspects, static crosscutting allows the changing of the class definitions themselves [KHH+01].

2.2 Runtime-Verification

Model checking has the power to assert that a property holds along any possible execution path of a system (whatever inputs it receives, and whatever
non-deterministic choices are made inside the system) [UT02,ZKTR07], making it a very desirable objective. However, model checking depends on having a decidable domain, and for the algorithm to be tractable when applied to systems of the magnitude one wants to analyse. Various abstraction and reduction techniques have been proposed to scale up model checking, but full verification of large-scale software systems is still largely unattainable [GH05,ZKTR07,FS01]. In contrast with model checking, in runtime-verification, one checks that a given system property holds along a particular execution path [ZKTR07]. This is particularly useful to ensure that at no time during the execution of the system, are any of the system properties violated. Conversely, it can also identify execution paths taken at runtime, and along which the properties to be verified are not satisfied [ZKTR07]. Essentially, runtime-verification links the abstract specification and the actual concrete implementation [LBK+98,STY03]. Thus, runtime-verification can be used as a protection from potential faults at runtime, by implementing monitors to react to any property violations encountered [GH05].

In general, a runtime-verification specification consists of (i) identification of the properties which are to be satisfied by the system; (ii) identification of the points during the execution when these properties are to hold; and (iii) identification of actions which should be taken when these properties fail.

Clearly, the logic chosen to write such specifications should enable straightforward expression of such properties, so as to avoid potential errors being introduced in the specification. Furthermore, the modification of the code in the concrete system to insert check for such a specification should be largely (or completely) automated, also to avoid potential errors being introduced in translating the properties into actual monitors. Finally, the properties should be cheap to verify at runtime, avoiding heavy overheads which substantially alter the real-time behaviour of a program. These two constraints limit one’s choice of logics, with the latter also being motivation for adopting AOP for automatic insertion of monitors in the code. AOP offers the advantage of specifying all such properties in separate modules (aspects) which contain all the logic concerning these properties [UT02]. Thus one avoids having the properties written as dispersed code throughout the system checking for property violations. Furthermore, it has been observed that other runtime-verification approaches for object-oriented systems, usually break the encapsulation principle from object-oriented programming [Bod05].

2.3 Expressing Temporal Properties

Most of the properties one would like to verify at runtime are temporal properties, in that they specify properties about the order of events in the underlying system. Examples of such properties are ‘ensure that initialisation takes place before other methods execute’, or ‘ensure that no more than three sequential bad logins are allowed’. Using such properties, the developer can reason about the order of and dependencies in the control flow of a system [SB06]. Unlike atemporal, or global assertions, temporal properties are concerned not only with
the current state of the system but also with its history — the sequence of states which led to the current state [SB06]. The logic to be used should thus enable the expression of properties on sequences of states.

Considerable work has appeared exploring the use of Linear Temporal Logic (LTL) as the way to express such temporal constraints for runtime verification [SB06,Bod05,BS06,SH05]. Bodden and Stolz [SB06,Bod05,BS06], use LTL for the specification, and translate it into AOP code expressing the weaving of monitors of the original LTL property into the concrete system. Sammapun and Sokolsky [SS03] use a similar approach but based on the description of the temporal properties through the use of regular expressions. Alternatives to such logics explored in the literature, are the use automata for the expression of temporal constraints — for example alternating finite state automata have been used in [SB06,Dru06,FS01], and timed automata were used in [Bou06,STY03,FH06] to reason about temporal properties. Although most logics can be naturally translated to and from automata, it is interesting to compare and contrast the complexity of expressing typical properties using the two approaches.

Sometimes, the sort of temporal reasoning one requires goes beyond the the ordering of events, adding operators to enable reasoning about the actual timing of the events. Such specifications would enable expressing properties like ‘no more than three bad logins can appear in any 30 minute period during the execution of the system’, or ‘after 30 minutes of inactivity, the server will automatically disconnect the user’. Certain temporal logics allow for the specification of such real-time properties. Adding real-time constraints to an existing system can be extremely complex, and usually it is much simpler to redevelop the system from scratch rather than adding reactivity, even if the underlying functionality is already there. Gal et al. [GSSP02] show how AOP can be used to allow the developers to separate the functional from the real-time concerns thus making code easier to develop and more reusable.

Another consideration in real-time systems is that upon instrumentation of the aspect code, the timing constraints are not violated through the introduction of the property monitors. Furthermore, one other important aspect one should keep in mind when introducing runtime monitors is memory usage. Excessive memory usage by the verification code may cause undesired effects on the system being considered, including timing violations. However, little research has been done on this issue of guaranteeing memory usage and timing performance after the instrumentation of aspect code.

3 Case Studies

In this section, we will present a number of simple case studies to illustrate the use of AOP techniques for security-critical systems. The technology employed for these case studies is AspectJ. The implemented server is a trivial one for illustration purposes, and receives requests from clients and if the requests are valid, these are served. If the server does not have a profile of a client with a particular ip address, then the client must first notify the server and then login using the
password set during the first notification. Once the client has successfully logged in, other services are available. The client can choose from the available services by specifying a request number, where each number corresponds to a particular type of request. Finally, the user can log out. Once logged out, the user only request a login while any other requests will not be granted. This is the basic specification of the server. However, on top of these properties, we would like to specify an extra property for security reasons: if a client with a certain ip address sequentially makes a number of invalid requests which exceeds a certain limit, then no more request of this client will be considered. The first two case studies give a description of two possible approaches of using AOP to secure this property.

3.1 Specifying Temporal Security Properties on an Existing System

In this first case study, we assume that the developer of the server has attempted to cater for the specified security property of blocking a user whose ip address has appeared repeatedly issuing invalid requests. The server code handling this condition is shown in Listing 15.1.

Listing 15.1: Client-blocking in the server code

```java
if (htBlackList.containsKey(c.ip) && ((Integer)htBlackList.get(c.ip)).intValue() > limit)
    {System.out.println("You are blocked");}
else
    {serve(c, reqNum);}
```

Since one would typically not want to look into the server code to verify that it is correctly implemented, we would like to independently implement the property as an aspect, which when woven into the server code will provide runtime verification of the specified property. The purpose of the aspect in this case is to act as a double check that the system is correctly implemented. Typically, in practice, the property would be concisely expressed in a temporal logic and automatically translated into an aspect. One would thus trust the property more than the underlying code.

During implementation, it was noted that if the implemented system was not well structured, then it would have been much more difficult to implement using AOP, since AOP code needs to be injected at specified joinpoints. A frequently used joinpoint is the method call. Therefore, if the system code is not well structured into methods, there would be many less jointpoints available. The main advice which can block a client is shown in Listing 15.2.

Listing 15.2: Client-blocking in AOP

```java
void around (Client cl, int r): (execution(* Server.serve(..))
    && args(cl,r)) {
    if (ht.containsKey(cl.ip)) {
        if (((Integer)ht.get(cl.ip)).intValue() > limit)
            System.out.println(cl+" :: Property violation detected. User will be blocked.");
    }
```
else proceed(cl,r);
} else proceed(cl,r);
}

The *around* advice employed in this case is a construct which allows the developer to receive control before the actual method is executed, thus enabling him or her to do anything before and/or after the actual method is called. This includes changing the parameters by which it is called and even stopping it from being called at all. In this example, the method *serve* is only instructed to go ahead (using the *proceed* construct) if the client is not blocked.

### 3.2 Specifying Temporal Security Properties as a Separate Module

A different approach from the previous scenario would be to separate the underlying functionality in the case of non-malicious users completely from the code handling security concerns. The problem in doing so using typical modularity offered by programming paradigms other than AOP is that the security features and the code expressing the ‘correct’ functionality are tightly knit together, passing control back and forth across the code. In the approach we present here, the implementation of the security features appear as AOP code, reaching into the design of the naïve server and weaving the features. Modularity is achieved since through AOP directives we can actually modify, as opposed to simply use, other code. The blocking mechanism in this case is not simply a *double check*, but the *actual check* itself.

The advantage of such a configuration is that the server code is left totally clean without any additional checks for ensuring security properties. Furthermore, all the security related code (part of which is shown in Listing 15.2) is now in one module (aspect) while in the previous case, there were two classes (apart from the AOP aspect) which contained code related to the security checks. Adding additional properties requires adding them to one module without having to understand the potentially complex logic of the server code riddled with runtime checks. Furthermore, one could imagine scenarios where the properties may be applied to different vanilla-servers, thus enabling reuse of the runtime properties.

During the design of these case studies it was noted that the decision of which properties are part of the actual server and which other properties are to be considered as extraneous, is totally arbitrary (for the developer to decide). For example, we could have decided that checking that the login is correct is not a part of the server’s functionality and implemented it as a separate aspect.

### 3.3 Specifying Temporal Security Properties on a Code Library

In the third case study we will present the use of AOP for runtime-verification in a library to stop (or warn) users of library in the case of wrong usage. Such inappropriate use of the code may lead to undesirable faults in the user’s applications. We use AOP techniques to weave temporal properties inside library code, to handle incorrect usage going beyond traditional pre- and post-condition
checking. The advantage of this approach is that the properties specified will not only hold for the currently implemented methods in the library, but also for extensions which can possibly be added in the future. In the simple example suggested here, we consider a scenario where the library should be initialised before any other method is used. Furthermore, once initialised, the library can be reset to return the library to an uninitialised state. The code which blocks any method call before initialisation is shown in Listing 15.3.

Listing 15.3: Library checking advice

```java
Object around():(execution (* Library.*(..)) && !execution(* Library.initialization(..))) {
    if (initialized)
        return proceed();
    else
        return "LIBRARY: Library must be initialized.";
}
```

Using the "*" wildcard, we have managed to check for initialisation before the execution of any possible method apart from the one whose name is `initialization`. One should note the efficiency of implementing such logic in a few line of code rather than inserting a condition at the start of all the methods in the library. Furthermore, adding further methods to the library does not necessitate any modifications to the code handling the property.

4 Case Studies with Real-Time Security Properties

In this section we will reconsider the previous examples with additional real-time constraints.

4.1 Specifying Real-Time Security Properties on an Existing System

In the case of the server, the added constraint was that after a certain period of time, a blocked client will now be unblocked once more and allowed to make requests. Therefore, the time at which a client was blocked is stored in a hash table. Then, each time a request is received from a blocked client, the time elapsed (since the denial of service) is checked and is unblocked if the time limit was exceeded. Given that our server implementation is very simply it was relatively easy to add the necessary logic for the newly introduced real-time constraint. However, in a real scenario, the consequence of adding new code in the actual server implementation may prove to be much more cumbersome. The AOP aspect was also updated to cater for the new real-time constraint. This was done by storing the time at which each client was blocked from the system. Subsequently, when a client sends a request, the current time is compared to the stored time so that if enough time has elapsed, the client is unblocked. This is shown in Listing 15.4.
158

Christian Colombo and Gordon J. Pace

Listing 15.4: AOP checking advice with real-time constraint
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18

pointcut request ( Client cl , int r ) : execution (* Server . serve
(..) ) && args ( cl , r ) ;
void around ( Client cl , int r ) : request ( cl , r ) {
if ( htBlacklist . containsKey ( cl . ip ) ) {
long currentTime = System . currentTimeMillis () ;
if ( currentTime - (( Long ) htBlacklist . get ( cl . ip ) ) . get
() < release )
System . out . println ( cl + " :: You are BLOCKED " ) ;
else {
System . out . println ( cl + " :: You are UNBLOCKED " ) ;
htBlacklist . remove ( cl . ip ) ;
if ( ht . containsKey ( cl . ip ) ) {
ht . remove ( cl . ip ) ;
ht . put ( cl . ip , new Integer (0) ) ;
}
proceed ( cl , r ) ;
}
}
else proceed ( cl , r ) ;
}

4.2

Specifying Real-Time Security Properties as a Separate Module

Adding the additional real-time constraint to the system with security as a
separate module proved to be much easier than the previous scenario. Basically,
the server code remained intact while the extra logic (the same as that in Listing
15.4) was simply added in the single module (aspect) which handles the security.
4.3

Specifying Real-Time Security Properties on a Code Library

The added real-time constraint in the library scenario was that after a certain
time period the initialised library automatically returns to an uninitialised state.
The implementation was done by storing the time at which the library was
initialised and adding an extra check (Line 3 in Listing 15.5) to ensure that
the initialisation is still valid when a user requires a method execution from the
library.
Listing 15.5: Library checking advice including real-time constraint
1
2
3
4
5
6
7

Object around () :( execution (* Library .*(..) ) && ! execution (*
Library . initialization (..) ) ) {
long currentClock = System . currentTimeMillis () ;
if ( initialized && currentClock - clock < limit )
return proceed () ;
else if ( initialized ) {
initialized = false ;
return " LIBRARY : Time expired " ;


5 Proposed Framework for Real-Time Properties

Although demonstrated in the case studies, the use of AOP may seem straightforward, in other cases it may be much more difficult to translate from the conceptual security property into the actual aspect code. We are currently exploring the use of different formal techniques for specifying real-time security properties in a natural way. The aim is to explore the use of decidable temporal notations, such as timed automata to describe properties which would then be automatically compiled to AOP code.

Timed automata, use timers, which can be used to specify when particular actions which will be performed if the system is within a certain state. To illustrate the approach we are exploring, we will present the real-time properties presented in the case studies and show how they would be represented using timed automata.

The server property is shown in Figure 2. The process of keeping count of the number of bad requests per client is hidden and the condition to enter into blocking mode is represented by the boolean input \( Br \). Once the system enters into blocking mode, the timer is reset to zero (represented by \( T:=0 \)). Eventually, the system only returns to unblocked mode when the timer reaches the desired amount (in this case 1000).

Figure 3 depicts the real-time property implemented in the library case study. In this example, the timer is reset when the initial input is set to true. Subsequently, when the timer reaches the threshold (also set to 1000), the system reverts back
Fig. 3: A timed automaton representing the real-time property implemented in the library case study.

to an uninitialised state. However, this time, the system will also return to an uninitialised state if the reset input becomes true while in the initialised state.

6 Conclusion and Future Work

In this paper we have presented ways of employing AOP for ensuring temporal security properties in a modular and succinct manner. Furthermore, we have shown how the same framework can be further modified to additionally include real-time properties. To allow for such real-time properties to be specified more easily we proposed a framework based on time automata.

We are currently exploring the use of variants of symbolic timed automata for the expression of real-time properties which can be automatically translated into aspects. In particular, we would like to explore their use on real-time security properties in security-critical systems. One interesting challenge to address, is memory usage of the monitoring code. One approach we believe could be fruitful, is the use of techniques developed for reactive systems such as Lustre [HCRP91].

References


Model Checking Concurrent Assembly Algorithms

Joseph Cordina, Stephen Fenech, and Gordon J. Pace

Department of Computer Science, University of Malta

Abstract. Model checking has been used in various domains, to enable automatic verification of properties for a given model. Especially in cases when the correctness of the the model is not evident due to the complex nature of the description, model checking can be an indispensable tool. One such domain is the use of concurrent assembly algorithms for low-level synchronisation, which can be notoriously difficult to check their correctness or even test. In this paper we look at this domain, and explore the use of model-checking in verifying a number of such algorithms, such as barrier synchronisation and wait-free CSP channel communication. We tackle the state explosion problem inherent in model checking by making use of abstraction techniques to remove redundant information in the the model, and partial-order techniques to remove redundant interleavings of actions. Finally, we also investigate the use of structural induction to reason about families of systems of arbitrary size. Making use of symmetry and induction, we verify algorithms with an unbounded number of identical participating tasks.

1 Introduction

Model checking has been extensively used in the verification of complex systems. In critical systems, analysis of properties along all possible execution paths makes it an attractive alternative to testing and simulation. In the field of asynchronous concurrent algorithms, building testing suites with high state-space coverage can be particularly challenging, since concurrency may introduce different interleavings which are difficult to control through the testing suite. In this paper, we present our recent results in the verification of concurrent wait-free assembly algorithms through the use of standard model checking techniques. We look into algorithms running on three different shared memory architectures: Single Processor, Symmetric Multi-Processor (SMP) and Asymmetric Multi-Processor (ASMP), with the main focus on the ASMP. In the SMP architecture, the processors are identical and share the same clock so they are synchronised [HP02]. On the other hand the ASMP architecture does not have such a constraint and thus we cannot know how long an instruction will take to execute on different processors. In particular, we look into wait-free algorithms used in the core of the KRoC Occam compiler [WW96], to handle communication between different threads. As with most real-life applications of model checking, the main challenge is
primarily that of controlling the state explosion problem. Due to the multiple interleavings that are possible in such algorithms, the state space grows very quickly. The second challenge is that of reasoning about parametrised systems. Rather than the verification of a stand-alone program, most of the algorithms used may interact with any number of other programs. For example, the resolution of channel synchronisation in a CSP-like domain may have any number of readers trying to take the data provided by the writer. We present structural induction techniques to reason about such families of systems using model checking.

In this paper we present the application of these techniques for the verification of the algorithms used internally by KRoc to handle thread barrier synchronisation and channel communication using wait-free algorithms [Vel98]. In both cases, we use inductive reasoning with model checking to prove the correctness of the algorithms for any number of processes taking part in the synchronisation.

2 Background

A Kripke structure is a tuple $M = \langle Q, I, t, v \rangle$, where $Q$ is a finite set of states, $I \subseteq Q$ is a set of initial states, $t \subseteq Q \times Q$ is a total transition relation between states and $v : Q \to 2^A$ is a valuation function over a set of atomic propositions $A$. The valuation function $v$ corresponds to which propositions hold in which states. We say that proposition $p \in A$ holds in state $q \in Q$ in $M$, written $q \models_M p$, if $p \in v(q)$. We extend this notation for boolean expressions. $q \models_M e \land f$ holds if both $q \models_M e$ and $q \models_M f$. Similarly, we define the other boolean operators.

The set of valid paths in a Kripke structure $M$ are infinite sequences of states $\sigma \in \text{seq}(Q)$ such that $\sigma_0 \in I$ and for all $i$, $(\sigma_i, \sigma_{i+1}) \in t$, where the subscript is the index of the path.

Linear Time Logic (LTL) is used to reason about properties along such paths. An LTL formula is either a boolean expression over atomic variables, or uses the temporal operators $G$, $F$, $X$ and $U$. The semantics of LTL over a path $\sigma$, are defined as follows ($\sigma^{+i}$ corresponds to the path identical to $\sigma$ but dropping the first $i$ initial items):

$$
\begin{align*}
\sigma &\models_M e \quad \text{df} \quad \sigma_0 \models_M e \\
\sigma &\models_M X p \quad \text{df} \quad \sigma^{+1} \models_M p \\
\sigma &\models_M F p \quad \text{df} \quad \exists i \cdot \sigma^{+i} \models_M p \\
\sigma &\models_M G p \quad \text{df} \quad \forall i \cdot \sigma^{+i} \models_M p \\
\sigma &\models_M U q \quad \text{df} \quad \exists i \cdot \sigma^{+i} \models_M q \text{ and } \forall i' < i \cdot \sigma^{+i'} \models_M p
\end{align*}
$$

A Kripke structure $M$ is said to satisfy LTL formula $e$, written as $M \models e$, if all valid paths in $M$ satisfy $e$.

Linear Time Logic without next-time operator (LTL/X) is identical to LTL, except that $X$ may not appear in properties.
### 3 Modelling

We model concurrent algorithms running on any of three different shared memory architectures: Single Processor, Symmetric Multi-Processor (SMP) and Asymmetric Multi-Processor (ASMP). In the SMP architecture, the processors are identical and share the same clock so they are synchronised [HP02]. On the other hand the ASMP architecture does not have such a constraint and thus one cannot know how long an instruction will take to execute on different processors. This can be modelled by giving each instruction the possibility to stutter so in this way an instruction may take any amount of time to finish. This also resolves the problem of knowing how many cycles each instruction takes, since the model will cover any number of cycles. In our model, the data and code will be assumed to reside in different memory areas. Furthermore, the programs will not have access to write to the code memory areas — in other words, we will be looking at static (non-self modifying) algorithms.

Algorithms are given a semantics in terms of a Kripke structure, with the states consisting of the data stored in the registers (including program counters) and the memory. The transition relation is based on the semantics of the assembly instruction found in memory at the location pointed to by the program counter. To reduce the model to a reasonable magnitude, only the memory locations used in the algorithms will be modelled.

![Kripke structure](image)

**Fig. 1:** Part of the Kripke structure of the mutual exclusion algorithm. The dotted transitions correspond to the other external process.

For instance, consider the two concurrent programs shown in listing 16.1. Both processes use memory location 1 in order to provide mutual exclusion. The state space will be made up of all the possible values of the memory location 1 and the value of the registers used (the program counters of the two processes $pc1$ and $pc2$).

```plaintext
     jne spin;                      jne spin;
   noop; // Crit.Sec.               noop; // Crit.Sec.
    mov [1] 1;                      mov [1] 0;
   jmp spin;                      jmp spin;

Listing 16.1: Mutual exclusion try
```
and pc2 and their equality, or zero flags). Figure 1 shows a small part of the reachable states of the Kripke structure derived from this program.

### 3.1 Formal Semantics of Assembly Programs

Based on the model we derive, one can try to verify that the algorithm actually guarantees mutual exclusion between the two concurrent processes — it is never the case that both processes will be in the critical section at the same time. This can be specified as the LTL property as $\mathcal{G}(\neg(pc1 = 3 \land pc2 = 3))$.

In table 1, one can find the operational semantics given to some typical assembly instructions. The transitions are between triples $(r, lm, gm)$ — with the information stored in the registers, local memory and global memory respectively. All transitions are labelled by two parameters $(\longrightarrow^M_{\text{prg}})$, the program being given a semantics, and the set of locations in global memory that are updated by the instruction. We call transitions with $M = \emptyset$, local transitions. In the case of parallel composition, the global memories are merged based on this latter parameter. Based on these semantics, we can give a Kripke structure interpretation to a program $\text{prg}$, with $q \rightarrow q'$ being defined as $\exists M \cdot q \rightarrow^M_{\text{prg}} q'$. We refer to this as $\lbrack \text{prg} \rbrack$.

**Lemma 1.** Any sequence of local transitions does not change the global memory component of the state: If $(r, lm, gm)(\longrightarrow^0_{\text{prg}})^*(r', lm', gm')$, then $gm = gm'$.

The proof follows by induction on the number of transitions taken, and the basic definitions of instruction semantics.

The state explosion problem unfortunately limits the size of the model which can be model checked in a tractable amount of time [EMCJ99]. Many techniques have been devised in order to be able to enable model checking of larger systems. We will now explain some domain specific optimisation we performed on the models produced.

The state space described in the semantics is simply far too big to handle using model checking. One of the main culprits is the fine-grain instructions, introducing far too many interleavings. Despite the fact that unless the global memory is modified, there is no interaction between concurrent programs, the semantics keeps account of all the individual instructions carried out in each thread. To relieve this problem, we introduce a less fine grained semantics, giving an abstraction of the original system, which collapses redundant individual instructions together.

We define $\longrightarrow^M_{\text{prg}}$ to be defined for sequential programs for any sequence of instructions not communicating with the global memory, followed by a single instruction that does or a loop which does not modify the global state:

\[
(r, lm, gm) \longrightarrow^M_{\text{prg}} (r', lm', gm') \overset{df}{=} (r'(pc) = r'(pc) \land M = \emptyset \land (r, lm, gm)(\longrightarrow^0_{\text{prg}})^*(r', lm', gm'))
\]

\[
\lor (r, lm, gm)(\longrightarrow^M_{\text{prg}} \circ (\longrightarrow^0_{\text{prg}})^*)(r', lm', gm')
\]
Sequential program semantics:

\[
\text{prg}\left[ r(pc) \right] = \text{noop} \\
(r, lm, gm) \rightarrow^{\text{prg}} (r[pc := r[pc + 1]], lm, gm)
\]

\[
\text{prg}\left[ r(pc) \right] = \text{mov} [m] n \\
(r, lm, gm) \rightarrow^{\text{mov}} (r[pc := r[pc + 1]], lm, gm[m := n])
\]

\[
\text{prg}\left[ r(pc) \right] = \text{jmp} n \\
(r, lm, gm) \rightarrow^{\text{jmp}} (r[pc := n], lm, gm)
\]

\[
\text{prg}\left[ r(pc) \right] = \text{jeq} n \land r(eq) \\
(r, lm, gm) \rightarrow^{\text{jeq}} (r[pc := n], lm, gm)
\]

\[
\text{prg}\left[ r(pc) \right] = \text{jeq} n \land \neg r(eq) \\
(r, lm, gm) \rightarrow^{\text{jeq}} (r[pc := r[pc + 1]], lm, gm)
\]

Stuttering:

\[
(r, lm, gm) \rightarrow^{\text{prg}} (r, lm, gm)
\]

Parallel composition:

\[
\text{prg} (r_1, lm_1, gm_1) \rightarrow_{\text{prg}_1} (r_1', lm_1', gm_1') \\
\text{prg} (r_2, lm_2, gm_2) \rightarrow_{\text{prg}_2} (r_2', lm_2', gm_2') \\
((r_1, r_2), (lm_1, lm_2), gm) \rightarrow_{\text{prg}_1 \parallel \text{prg}_2} \text{merge}((gm_1', M_1), (gm_2', M_2))
\]

Table 1: Semantics of assembly programs
Furthermore, we constrain the states of the automata to the initial states, the destinations of global transitions and jump instruction locations which enable a local loop. We can use this rule to induce more compact sequential program semantics, using the stuttering and composition rules to define a more compact semantics for concurrent programs. We will refer to such semantics of a program prg, as $\langle \text{prg} \rangle_A$.

**Theorem 1.** Given a program $\text{prg}$, and an LTL/X formula $\pi$ in which basic propositions refer only to the global memory, $\langle \text{prg} \rangle \models \pi$ if and only if $\langle \text{prg} \rangle_A \models \pi$.

The proof follows in a straightforward manner using structural induction on the temporal properties, and the fact that the leaf properties change only after a global transition (lemma 1). Furthermore, local loops are maintained in the abstract system, ensuring that deadlocks are not lost. The result of this theorem can be extended to allow the LTL/X formula to refer to program counter values at the end of a global memory access.

In practice, the abstraction reduces the reachable state space drastically. Furthermore, after collapsing such chains of instructions, we can also syntactically look at the program and identify registers and local memory locations which no longer have an affect on the execution of the program. Since we use a symbolic model checker, we prune out such state variables to reduce the state space of the resulting system even further.

### 3.2 Reasoning about Families of Processes Using Induction

One problem with most algorithms at the core of a compiler, is that the interaction is not strictly between a fixed number of processes. For instance, if one looks at a multi-way synchronisation algorithm, it should work regardless of the number of participating entities. Similarly, in the case of channel communication in Occam, one can have one single writer, and multiple reader competing for the channel. In both these examples, the problem is to prove that a system satisfies a property for any number of processes. For a property $\pi$, the property one would like to verify is of the form:

$$\forall n \cdot \langle Q\parallel P^n\rangle \models \pi$$

To prove this property for any number of copies of $P$, one can use an inductive approach, by taking the weakest process $\alpha$ satisfying $\pi$, and proving that (i) $Q \models \pi$; and (ii) $\alpha\parallel P \models \pi$. The first property can be easily verified using the techniques already presented. The second is more difficult to prove, since we need to be able to generate $\alpha$. One possibility is to consider the system with a chaotic system for $\alpha$ (chaotic in that it can perform any action), and constrain the model checking tool to work on paths for which $\alpha$ satisfies $\pi$. In general this is not straightforward to do, but by limiting ourselves to safety properties expressed as observers [HLR94], which take the input and output of the system and return one output stating whether or note the system is running correctly, we can reason about the weakest system directly, proving that if the observer running on the chaotic system has always been true in the past, the observer of
the global system will also be true. In this manner, we enable reasoning about whole families of systems.

\[ \text{chaos} \parallel P \models G \text{(observer(chaos) } \Rightarrow \text{observer(chaos} \parallel P)) \]

4 Case Studies

In this section we will look into a number of case studies we will verify using the techniques described in this paper.

4.1 Thread Barrier Synchronisation

Barrier synchronisation algorithms are widely used when a job is split between a number of processes which then wait for each other to finish in order to agglomerate the results. This algorithm makes use of two semaphores and two different threads to do the synchronisation. With \( k \) threads to synchronise, \( k - 1 \) threads will signal on a semaphore \( A \) followed by waiting on semaphore \( B \). The remaining thread (referred to as the asymmetric thread) tries to cancel the effect of the other threads by waiting for \( k - 1 \) times on \( A \) then signaling semaphore \( B \) for \( k - 1 \) times freeing the other threads waiting on this semaphore.

Using this model we model check that all the threads eventually reach the barrier before any continue any further. Applying abstraction meant a reduction of state variables by an average of 18\%, enabling us to verify the algorithm for up to 10 processes taking less than 15 minutes.

4.2 Generalised Thread Barrier Synchronisation

Induction cannot be applied directly to the model just described, since the semaphore location needs to be to hold a value of up to \( k \), for a general value \( k \). The solution we adopt, is that we encode the semaphore location and the asymmetric thread into a single module.

When, in the algorithm, one adds a new thread, one also has to add a new wait to the asymmetric thread and a new signal at the end.

In order to model this we are going to create a number of components. Module \( \text{SemA} \) will take two signals as input which outputs true once both have been received. Module \( \text{SemA} \) models the part of the asymmetric thread which waits on semaphore \( A \). Module \( \text{SemB} \) models semaphore \( B \) and the part of the asymmetric thread which signals on semaphore \( B \) in order to free the symmetric threads. The symmetric thread is modelled as module \( P \) and has three states: the signalling state where it signals semaphore \( A \); the waiting state where the thread waits on semaphore \( B \); and the ready state.

Figure 2 shows the interaction between the modules. In order to add another thread, one needs to add more symmetric threads and cascade the semaphore modules. Here, the ready signal from the first \( \text{SemA} \) is directed to \( \text{Signal1} \) of the second thus the final \( \text{SemA} \) module will issue the ready only when signals have
been sent to all the semaphores. This is then directed to the SemAReady of the SemB modules.

For this proof, a number of observers are used. One observer checks that it receives all the signals before receiving the wait requests. Furthermore, another observer checks that each signal/wait is received only once.

With these observers in place, for the base case, all one needs to model check is that their output is always true.

For the inductive case (Figure 3), we leave the modules to work in a non-deterministic manner (representing the behaviour of the $k$ processes), and add another unit (as the $(k+1)$th process). Two instances of the observers will be used to monitor the behaviour of the blocks inside (outputting $ok_k$), and that of the overall system (outputting $ok_{k+1}$. By thus need to show that restricting the behaviour of the $k$ processes to satisfy the property, the global system still satisfies the property: $G (always ok_k \Rightarrow ok_{k+1})$ (where always ok$_k$ checks whether ok$_k$ was always true in the past).

Using this approach, we model check the correctness of thread barrier synchronisation for any number of communicating processes.

4.3 Wait-Free CSP Channel Communication

In this case study we look at, and model check Vella’s wait-free CSP channel algorithm for shared memory multiprocessors [Vel98]. This algorithm is used in the kernel of the KRoC [WW96] OcCam compiler. Wait-free algorithms offer various advantages over locking algorithms but they are typically intricate algorithms, difficult to develop and to confirm their correctness under all possible interleavings.

We first present simple channel communication in order to get a grasp of the basic algorithm followed by alternate channel communication. In the CSP model of concurrency two processes may communicate only through blocking and unidirectional channels.
Simple Channel Communication In the simple channel communication the inputting and outputting tasks have three shared memory locations; the channel word, input workspace and output workspace. The channel word will store the workspace’s address of the task which has committed itself to the communication. The output workspace will store the output value whilst the input workspace is where the value is stored once received. The output task will behave similarly to the input task seen in listing 16.2.

```
1  Swap InputWorkspace location into the channel word;
2  if channel word was 0 {
3        Sleep; (wait for output task to arrive to same point)
4  } else (output task is already waiting) {
5        Reset the channel word to 0;
6        Copy value from OutputWorkspace to InputWorkspace;
7        Wake the output task;
8  }
```

Listing 16.2: Pseudocode of channel communication (input task)

This algorithm can be modelled directly using the translation we have given, and we verify that (i) the algorithm never deadlocks (the tasks always finish); and (ii) that channel communication is always successful (the correct value is transferred from the output to the input workspace).

Alternate Channel Communication In the alternate channel communication, a single inputting task receives an input from a number of channels — corresponding to

\[
(a_1, P_1 + a_2, P_2 + \ldots + a_n, P_n) \mid \tilde{a}_1, Q_1 \mid \tilde{a}_2, Q_2 \ldots \mid \tilde{a}_n, Q_n
\]

An additional memory location which in [Vel98] is referred to as a Pointer is used to hold the state of the input task. An outputting task will behave mainly like in the single channel communication except that when it finds the address of the inputting workspace it has to inform the inputting task by swapping READY into the Pointer (listing 16.3).

The input task (listing 16.4) starts by enabling the channels. Once the enabling is ready it will swap WAITING into the pointer. If the value was still ENABLING it can safely go to sleep. If on the other hand it was READY, it means that an output task has committed itself so the disabling phase will follow. Since we have already swapped WAITING into the pointer, we need to set it back to READY. This is not done atomically thus an output task may mistakenly read that the input task was in the WAITING state. We can identify such a situation by swapping READY into the pointer. If it is still WAITING it means that no output task has read this value, so we can continue normally. If not, we simply go to sleep and leave it up to the output task to reawaken the inputting task.
Swap OutputWorkspace location into the channel word
if channel word was 0 {
    Sleep (wait for input task to arrive to same point)
}
else (input task is waiting or still enabling) {
    Swap READY into Pointer
    if pointer was WAITING (input task is sleeping) {
        Wake input task;
    }
    Sleep;
}

Listing 16.3: Pseudocode of alternate channel communication (output task)

for every channel (Enabling phase) {
    Swap InputWorkspace location into channel;
    if channel was not 0 {
        set Pointer to READY;
        restore channel word to previous value;
    }
}
Swap WAITING into Pointer;
if Pointer was Ready {
    Swap READY back into Pointer;
    if Pointer had become Ready anyway {
        Sleep;
    }
} else it is ENABLING {
    Sleep;
}

for every channel (Disabling phase) {
    Swap 0 into channel
    if channel was not InputWorkspace location {
        restore channel word to previous value;
        store channel id in order to read from this channel;
    }
}
Reset the channel word to 0;
Copy value from chosen OutputWorkspace to InputWorkspace;
Wake the corresponding output task;

Listing 16.4: Pseudocode of alternate channel communication (input task)
Once the communication occurs the input task will disable the channels and then obtain the value directly from the output workspace and then wake the output task.

**Modeling the Algorithm** Abstractions are used to simplify the algorithm thus making the problem tractable. Rather than storing the workspace address in the channel word it is sufficient to store a single value storing one of three possible values: (i) the input task has committed; (ii) an output task committed; (iii) no task has yet committed. Despite the abstractions used, the model obtained was still too large to reason about. The solution we adopted was to abstract further from the implementation, reducing the state variables by an extra 80%, but separately model check that this abstraction is correct. Using the abstracted model we verified the following properties for up to six output tasks running in parallel:

1. If the processes loop, to input (and output) repeatedly, the input process will terminate infinitely often, implying that the algorithm introduces no deadlock.
2. It is never the case that every task is sleeping.
3. Every output task may output — although the algorithm does not deal about starvation, and an output process may never manage to output, there is always a possibility (path in the state graph in which) it manages to do so.
4. The communication is sound — once an output task is chosen, the correct value will be transferred to the input task.
5. No output is lost.

### 4.4 Wait-Free CSP channel Communication with an Arbitrary Number of Output Processes

Induction was approached in a similar manner as in Section 4.2. We first identify elements (both memory and code) that need to be replicated for every output task that is added. The main issues was the enabling and disabling of channels and the shared access to the pointer value since we will have an unknown number of channels. These were defined recursively as a number of communicating modules thus abstracting away from threads and processors. Using this model we proved that given any number of outputting processes (i) the input task will repeatedly receive inputs from the output tasks and will never end up in a deadlock; and (ii) an output task may eventually perform an output even when running in parallel with an unbounded number of output tasks (an output task may never be chosen to output since the algorithm does not ensure fairness).

We made use of a number of assumptions in order to prove these properties: (i) both the enabling and disabling phase terminate; and (ii) the tasks will eventually gain access to the pointer value. These assumptions were needed since we generalise how many output processes are running.
5 Conclusions and Future Perspectives

In this paper, we have looked into the application of model checking techniques for the analysis of compiler-kernel wait-free algorithms used in the KRoC Oc-cam compiler. We have presented different techniques for state space reduction including abstraction and partial-order analysis. We have also used structural induction techniques for the verification of families or networks of processes, to ensure that the compiler-kernel algorithms work well for any number of interacting processes. These techniques have been developed into a tool, which uses SMV as a back-end for verification. It has been applied on a number of algorithms used in the core of the KRoC compiler, which have been verified correct.

Leven et al [LME04,Meh06] look into the model checking of assembly code where instead of creating a model they make use of a virtual processor to avoid any potential errors in the translation. A similar approach to ours was taken by Basin et al [BFG03] in order to model check bytecode instructions. There is also a great deal of work concerning abstraction in order to prove properties over larger systems [CGL94]. We made use of symbolic methods of abstractions but another type of abstraction is abstract model checking as in [CC99,Gra94,Gra99].

Partial-order abstraction similar to the one employed here is also employed in StEAM [Meh06]. Structural induction, as used in our approach, has also been employed in a in [McM92] and is discussed in [Jha96] as employing symmetry.

In our case studies, we have looked at LTL properties of these systems. Clearly, concurrency introduces various issues which require a branching time logic to express. Certain properties, such as 'whenever the algorithm is at location start, with register r being zero, the other thread may eventually reach the location critical', cannot be expressed in a linear time logic. We plan to look into the use of other temporal logics such as CTL and \( \mu \)-calculus, to verify more properties of our systems.

We have used modular verification techniques to verify networks of processes. These techniques were also useful in the abstraction of assembly code to reduce the complexity of the model checking task. However, the use of modular verification techniques was rather ad hoc, and we plan to extend our verification tool to enable management of module properties to support this form of reasoning.

References


An Embedded Geometrical Language in Haskell: Construction, Visualisation, Proof

Maria Grima and Gordon J. Pace
Department of Computer Science, University of Malta

Abstract. Geometric constructions based on compass and straight-edge have been thoroughly studied and explored. In this paper, we present a language embedded in Haskell, to describe, manipulate and analyse such constructions. The use of embedded languages has been explored in various specialised domains, and have been shown to be an excellent front-end to describe such specialised programs, enabling, for instance, the description of families of constructions as functions in the host language, which produce different specialised programs based on input parameters. In particular, we are interested, not only in providing a framework within which one may describe a construction and families of similar constructions in an algorithmic fashion, but also in providing facilities to both test and verify certain properties of constructions, such as equivalence of constructions, or equality of angles and distances in a construction.

1 Introduction

Various tools exist, illustrating the concepts of geometric constructions, allowing the drawing and manipulation of user-given constructions. The main drawback with most interfaces to such systems is that to have a user-friendly front end to the drawing program, one sacrifices desirable features (or at least, easy access to such features), such as reuse of constructions, constructions which may require the repetition of certain actions until a condition is met, and other similar features. In this paper, we present the design of a domain-specific language for geometric constructions, and the use of embedded languages to enable the creation and analysis of constructions in this language. In particular, we are interested not only in providing a framework within which one may describe a construction and families of similar constructions in an algorithmic fashion but also in providing facilities to both test and verify certain properties of constructions, such as equivalence of constructions, or equality of angles and distances in a construction.

1.1 Geometric Constructions

Geometric construction on the plane has proved to be a fascinating mathematical area since Greek antiquity, with the idealised straight line and the circle being considered as basic figures. Since then, mathematicians have been studying these geometric constructions, where one starts with a set of points, lines and
edges, and is required to construct new points, lines and circles satisfying a given specification. By constraining the tools used to calculate the new points, lines and shapes, one is essentially setting an axiomatic base, and by investigating the expressiveness and limits of the tools, one is actually studying completeness results of that axiomatic base. Compass and straight-edge constructions limit the tools to a collapsible compass, which can be used to draw a circle centred on a known point, with another known point on the circumference (the compass is said to be collapsible in that it collapses as soon as it is pulled off the plane, essentially stopping direct transfer of distances using the compass), and an unmarked straight-edge, which can be used to draw a line between two known points (it is said to be unmarked in that measurements are not allowed using the straight-edge). Such constructions, sometimes also known as Euclidean constructions, have been explored for centuries, and various interesting results exist, both in terms of constructions which can be achieved (such as bisecting an angle, bisecting a line, and constructing a regular pentagon), and ones which cannot (most famously trisecting an arbitrary angle, squaring the circle and doubling the cube). All such geometric constructions are based on two basic concepts: equidistance established by the use of the compass and collinearity established by the use of the straight edge.

A geometric construction is an algorithm, a step-by-step process, that builds a geometric figure or model. Such constructions are taught in schools to illustrate basic geometric concepts and rudimentary notions of a constructive proofs. Some researchers claim that the use of geometric constructions can sustain proofs and help to make geometric relationships more understandable through their visual representation [San98]. From a pedagogic point of view, they can also provide the motivation required for students to solve problems by reasoning about them [EA02].

1.2 Embedded Languages

When designing algorithms restricted to a particular domain, it may be beneficial to design a language that is targeted to that domain rather than using a general-purpose one. However, the initial effort required to develop such a domain-specific language and the difficulties in evolving the language as changes are requested, are often the cause for failure in complete development of such languages, especially when its use will not be so extensive. Similarly, establishing proper semantics for a new language also requires a great deal of effort. One approach which alleviates some of these problems is that of embedding the domain-specific language inside a general-purpose programming language [Hud96a, Hud98]. Using this approach, one designs a domain-specific library, which enables programs in the domain-specific language to appear as data objects inside the host language. In this manner, one gets to borrow features from the host language automatically, for things such as sub-program definition and control structures. Although the basic components of the domain-specific language are still to be designed, and built in such a way that they do appear as part
of a normal program in the host language, borrowing features of the host language drastically reduces language development time. Furthermore, tools such as interpreters and compilers are automatically available, since the domain-specific programs are also programs in the host language. All that one needs to do is to provide an interpretation function (or multiple ones) of the basic features of the domain-specific language. Arguably, one of the major advantages of this approach, is that the host language acts as a meta-language of the domain-specific language, allowing the program to generate regular domain-specific programs, or modify or analyse them accordingly.

It is important that the host language is flexible enough (both in terms of abstraction features and syntactic features) to enable the designer of the domain-specific language to ensure that domain-specific programs are an indistinguishable part of a program which mixes domain-specific and host language features. Haskell [Jon03] has been shown to provide an excellent infrastructure to embed languages, and there have been several successful implementations of domain-specific embedded languages hosted in Haskell addressing various domains, including geometric region analysis [HJ94], animation [EH97], hardware description [BCSS98] and music composition [Hud96b].

In this paper, we look into the design of an embedded domain-specific language to describe compass and straight-edge constructions in Haskell. Geometric constructions will thus appear within Haskell programs, with a library of functions to allow the visualisation of such constructions and their analysis. In particular, we emphasise the analysis aspect, with functions to test and verify properties.

2 Embedding Geometric Constructions

In keeping with the programming style of the host language, we have embedded geometric constructions as functions. The basic constructors of the language clearly are the drawing of lines and circles based on known points, and the identification of points as intersections of known lines and circles. Although constructions are usually given in a sequential manner, the order of the instructions is conceptually not a strict one, in that one should be able to perform any instruction, as long as the inputs are already known. Consider the following typical instructions explaining how to find the midpoint of two given points $p_0$ and $p_1$ (as shown in figure 1):

1. Draw a circle $c_0$ centred on $p_0$, passing through $p_1$.
2. Draw another circle $c_1$ centred on $p_1$, passing through $p_0$.
3. Find the intersection points of circles $c_0$ and $c_1$, calling them $p_2$ and $p_3$.
4. Draw a line $l_0$, passing through $p_0$ and $p_1$.
5. Draw another line $l_1$, passing through $p_2$ and $p_3$. 
6. Find the intersection of lines $l_0$ and $l_1$. This is the mid-point of $p_0$ and $p_1$.

Clearly, the order of steps 1 and 2 is irrelevant, but both must be performed before the third step. Step 4 depends only on the input, and could thus be performed earlier. Although we could have kept the exact order in which the instructions are specified, through basic combinators which specify a strict ordering, possibly through the use of monadic constructs, we prefer the view that, unless there is a dependency, the order of instructions is not important, and thus could be done in any order. For anyone familiar with Haskell, the following description conveys precisely this meaning, and is the spirit of the constructions we sought to create:

```haskell
midpoint (p0, p1) = midpoint
    where
        c0 = circle (p0, p1)
        c1 = circle (p1, p0)

        [p2, p3] = intersections (c0, c1)

        l0 = line (p0, p1)
        l1 = line (p2, p3)

        midpoint = intersection (l0, l1)
```

A straightforward interpretation of the above piece of code would be to consider points as actual coordinates, and lines and circles as pairs of concrete coordinates. The function could thus be used to evaluate the midpoint of actual points on the plane. However, we want to provide other interpretations of the description, and thus a shallow embedding would not suffice our purposes. The construction is a deeply embedded, structural description, which can be evaluated in different ways, for different purposes.

### 2.1 Strongly-Typed Shapes

The most crucial aspect in embedded languages is probably the type system used for the domain we are dealing with. A sound type system guarantees that a user is eventually restricted to use each function with data of the right type. At the same time, it must ensure that enough flexibility is provided for the user to be able to use the available functions in all required ways. In other words, a domain-specific embedded language must provide a type system that is sound and complete. An approach which involves phantom types [Rhi03] — parameterized types whose instances are independent of the type parameters — is one possible way to meet these objectives. Another alternative would be to use type-classes to enable overloading of certain functions like `intersection`, but we opt for the former approach to avoid cluttering function types with type constraints.
data Shape a = Shape UntypedShape

data UntypedShape =
  UntypedCircle (UntypedShape, UntypedShape)
  | UntypedLine (UntypedShape, UntypedShape)
  | UntypedIntersect (UntypedShape, UntypedShape)
  | ...

data Circle = Circle
data Line = Line
data Point = Point

Since the user would only be able to view the Shape parametrised type and the three dummy types Circle, Line and Point, and create instances of shapes through constructor functions we provide, we can enforce type safety through the use of strongly typed constructor functions:

circle :: (Shape Point, Shape Point) -> Shape Circle
circle (Shape p0, Shape p1) = Shape (UntypedCircle (p0, p1))

intersection :: (Shape a, Shape b) -> Shape Point
intersection (Shape s1, Shape s2) = Shape (UntypedIntersect (s1, s2))

...

2.2 Non-deterministic Constructions

Most descriptions found in textbooks use the reader’s visual model to refer to shapes to resolve ambiguity, in particular that induced through the non-determinism found in order of the results of finding intersection points of a circle and another shape. One solution is to use conditional constructs which, for example, enable the user to identify whether two points are equivalent, or whether they lie on the same side of a line. Another solution, is to give a deterministic interpretation to intersection. One such interpretation is the ordering of points on a shape. The intersection points of two shapes will then be given ordered by the ordering on the first shape. We take the ordered point approach, with points on a circle starting on the given point on the circumference and turning clockwise while the points on a line are ordered in the direction of the vector subtended between the two given points. However, we also introduce means of reasoning conditionally about points through equivalence checking to simplify certain constructions.

For example, consider the problem of constructing an equilateral triangle — given two points $p_0$ and $p_1$, it is required to identify a third point $p_2$, such that all three points are equidistant. Clearly, as long as $p_0$ and $p_1$ are distinct points, one can find two possible solutions to this problem.

equilateral0 :: (Shape Point, Shape Point) -> [Shape Point]
equilateral0 (p0, p1) = ps
where
c0 = circle (p0, p1)
c1 = circle (p1, p0)

ps = intersections (c0, c1)

Based on this description, we can draw a regular hexagon with a given side, using the selection function differentFrom, which filters a given list of points to ones which are different from a particular given point:

hexagon0 :: Shape Point -> [Shape Point]
hexagon0 (p0, p1) = [p0, p1, p2, p3, p4, p5]
where
  (centre::_) = equilateral0 (p0, p1)
  [p2] = equilateral0 (centre, p1) 'differentFrom' p0
  [p3] = equilateral0 (centre, p2) 'differentFrom' p1
  [p4] = equilateral0 (centre, p3) 'differentFrom' p2
  [p5] = equilateral0 (centre, p4) 'differentFrom' p3

Note that here we chose an arbitrary centre (from the two possibilities) for the hexagon.

On the other hand, if we constrain the specification with the extra condition that the three points $p_0$, $p_1$ and $p_2$ turn in a clockwise direction, we can give a deterministic solution using implicit ordering:

equilateral1 (p0, p1) = p2
where
c0 = circle (p0, p1)
c1 = circle (p1, p0)

p2 = first (intersections (c0, c1))

Note that first takes the head of a list of shapes. Using this code, one can then produce a regular hexagon starting from a given edge:

hexagon1 :: Shape Point -> [Shape Point]
hexagon1 (p0, p1) = [p0, p1, p2, p3, p4, p5]
where
centre = equilateral1 (p0, p1)

p2 = equilateral1 (centre, p1)
p3 = equilateral1 (centre, p2)
p4 = equilateral1 (centre, p3)
p5 = equilateral1 (centre, p4)

Although the order of execution of the constructors is not crucial in a description of a construction, the use of shared expressions clearly is. In the code for drawing hexagons, the centre of the hexagon is computed once and used four times.
However, due to referential-transparency, the code given is identical to that with the description of how the centre is drawn replicated. In practice, the data structure constructed for a hexagon is a tree, with multiple copies of the process used to calculate the centre. If we were to compute, for example, how many circles are drawn in the description of a hexagon, the circles used to compute the centre are added on four times, which is clearly undesirable. Various techniques have been presented in the literature to identify shared nodes in a structure. Explicit tagging of nodes in which every shared node in a structure has to be given a name explicitly by the user is one solution [O’D93]. An alternative is to take a monadic approach, using a state monad to compute tags automatically [BCSS98]. The solution we adopt is that of observable sharing [CS99], which introduces non-updateable references breaking referential transparency in a limited way. The abstract datatypes presented earlier are thus all encased within an additional reference type.

2.3 Parametrised Constructions

One advantage of embedding a language, is that the host language automatically acts as a meta-language, enabling us to define functions which can produce a family of constructions depending on inputs which it is given. For example, one can produce a function, which given a natural number \( n \), returns a construction which given two points, returns a point which is \( \frac{1}{2^n} \) of the distance from the first to the second point:

\[
\text{approach} :: \text{Integer} \to (\text{Shape Point}, \text{Shape Point}) \to \text{Shape Point}
\]

\[
\text{approach} 0 (p0, p1) = p1
\]

\[
\text{approach} n (p0, p1) = \text{approach} (n-1) (p0, \text{midpoint} (p0, p1))
\]

Consider the repetition inherent in the example given earlier with the hexagon construction. One may give a general description which, given the number of sides of a regular polygon, its centre, and an edge of the polygon, produces the list of vertices of the polygon. To do this we start by giving a higher-order construction \( f \), which is given a construction from a pair of shapes to a new shape (all of the same type), and a pair of shapes \( x \) and \( y \), and returns an infinite list of shapes corresponding to a Fibonacci-like list: \([x, y, f(x, y), f(y, f(x, y)), \ldots]\):

\[
\text{repeatConstruction} \text{ construction} (x, y) =
\]

\[
x: \text{repeatConstruction} \text{ construction} (y, z)
\]

\[
\text{where}
\]

\[
z = \text{construction} (x, y)
\]

An \( n \)-sided regular polygon can now be described as taking the first \( n \) elements of repeatedly finding the next vertex circumscribing the polygon:

\[
\text{regularPolygon} n (\text{centre}, (p0, p1)) = \text{points}
\]

\[
\text{where}
\]

\[
c = \text{circle} (\text{centre}, p1)
\]

\[
\text{nextPoint} (p, p') = \text{second} (\text{intersections} (c, \text{circle} (p', p)))
\]

\[
\text{points} = \text{take} n (\text{repeatConstruction} \text{ nextPoint} (p0, p1))
\]
2.4 Visualisation of Constructions

Since we keep the whole structure of a construction in our datatype, we can use it to transform it into an explanation using textual and visual means. In our geometric construction suite, we provide various visualisation functions, including textual explanations on the lines of the natural language explanation given in section 2, graphical step-by-step explanations in a Postscript document, and an HTML document combining the textual and graphical descriptions. We also enable the user to view constructions in a three dimensional animation by generating input for an external 3D rendering application written for our domain-specific language [Sal07] (see Figure 2).

3 Analysis of Constructions

Geometric constructions can become rather large and difficult to ascertain their correctness. We provide a number of techniques for the analysis of a given construction, both through the use of testing and theorem proving.

3.1 Testing Constructions

Since we can already evaluate the result of a construction with concrete inputs, testing can be performed simply through the random generation of input points, and checking that the outputs satisfy a given property. We provide two structured approaches to testing constructions:
QuickCheck Testing: As with all Haskell programs, QuickCheck [CH00] can be used to test properties of geometric constructions. To aid the user in testing, we provide a number of functions to calculate information such as distance between points, and angles subtended by three points. Properties can then be written in QuickCheck style in a straightforward manner. For example, to check that the equilateral triangle construction works, we can write the following property:

```haskell
prop_distance (p0,p1) =
  distance (p0,p1) == distance (p0,p2) &&
  distance (p0,p1) == distance (p1,p2)
where
  p2 = equilateral1 (p0,p1)
```

The function checkProperty can then be used to invoke QuickCheck to test the property:

```haskell
> checkProperty prop_distance
OK, passed 100 tests.
```

External Testing: We also provide another means of testing a construction by generating optimised C code to try to falsify the property through random testing. Whereas in QuickCheck, properties are written in plain Haskell, since they are evaluated directly by the Haskell interpreter or compiler, in this second approach, arithmetic and geometric comparison operators which one may need for the property specification language are also deeply embedded in Haskell. The upside of this approach is that expressions can be massaged and optimised before compilation, whereas in QuickCheck, the expressions have to be interpreted in every iteration. Furthermore, since both the constructions and properties are all relative to the inputs, thus guaranteeing that properties are invariant under transformation and scaling of the construction, we optimise further by fixing the first input point to lie at the origin, and the second at \((1, 0)\)\(^1\). This last optimisation improves testing drastically, especially when the construction has a small number of inputs. In the case of the equilateral triangle, it turns out that with a single testing point, the property can in fact be exhaustively verified correct.

### 3.2 Verifying Constructions

Although testing can be beneficial when trying to find bugs in a construction, one can benefit from the mathematical foundations underlying geometric constructions to actually prove that a construction works as intended. We have implemented an automatic geometry theorem prover, applying the full-angle [Wu87,CG96] method, to enable such reasoning.

\(^1\) Here, we consider the case when the input points are all distinct. The other cases can be checked separately.
The full-angle method enables one to reason about equality of angles, and collinearity of lines. It works by transforming the proof goal into an equation in terms of full-angles, where a full-angle is the rotation required for a particular edge to become parallel to another. The method then proceeds by farming geometric information predicates from the construction, for example deducing that any two points lying on the same circle are equidistant from the centre of the circle. A forward chaining process then follows, in which a number of inference rules are applied to the known predicates, producing more information predicates. Rewrite rules are then used to transform the predicates into full-angle equations to try to prove the original conjecture. The full-angle proof method is not complete on its own, but can be made complete through the use of the full-area method.

Verification proceeds in a very similar manner to testing, in which the user builds a conjecture, where the inputs are taken to denote universally quantified:

\[
\text{conjecture} \ (p_0, p_1) = \text{collinear} \ (p_0, p_1, p_2) \\
\text{where} \\
P_2 = \text{midpoint} \ (p_0, p_1)
\]

Applying the `prove` function to the conjecture starts off the proof search, which will result in a proof consisting of the initial predicates (and derived ones), and followed by a proof script of how the result follows in terms of full-angles.

4 Conclusions

The aim of this paper is to discuss the challenges and possible solutions when building a validation and verification based geometrical construction assistant. Our approach was to start with an algorithmic view of the constructions, which can then be visualised at a later stage. The use of embedded languages allowed us to produce regular constructions through a two-staged interpretation of the parametrised constructions. Apart from the actual construction process, we emphasise the analysis of their correctness, enabling straightforward access validation and verification methods.

Various tools exist for the description and manipulation of geometric drawings. For instance, as in our case, Cabri Geometry [PA96] allows drawing of objects on a geometric basis rather than on a perceptual basis. Constructions can also be instantiated and reused as the input to other functions for higher-order constructions. This approach is very similar to the approach that we took to provide the functionality for a user to define constructions as well as being shown the corresponding explanations, but without the capability of analysis. GeoView [BGP04], on the other hand, emphasises verification, and provides constructive capabilities to generate statements representing geometric theorems, and interactive means to build proofs via constructions. The reasoning is all done using the general purpose theorem prover Coq as a back-end. Finally, Geometry Explorer [WF05] also provides access to a full-angle method based prover implemented in
an embedded geometrical language in Haskell is partially based on Prolog, but provides only a graphical front-end, thus limiting the complexity of the constructions.

Currently, all parametrised constructions in our system are generated by Haskell programs, using recursion on parameters from the Haskell side of the program. It would be interesting to add stronger condition checks on geometric constructions (such as whether two shapes intersect, or whether two points lie on opposite sides of a line), and thus enable stronger interaction between the Haskell and geometric parts of the code. Given a construction which given two adjacent vertices produces the next vertex of the polygon, one could, for instance, generate the remaining vertices iterating the construction until a vertex is repeated.

We believe that the use of embedded languages, providing us with a meta-language in which to generate, transform, and analyse constructions is a very strong contender for a front-, to middle-end for an educational tool for geometrical constructions. Building a front-end, enabling beginners to draw constructions (which can be automatically translated into programs in our embedded language) would clearly be needed. However, the possibility to describe constructions in a programming environment is too strong to hide away as a possible means of input from the user. Finally, one can connect the language to multiple validation and verification tools, as for example is the case with Lava [BCSS98], which is connected to various model-checkers.

References


An e-Commerce Framework for Wearable Devices

Nicholas Micallef and Matthew Montebello

Department of Artificial Intelligence, University of Malta
nicholasmicallef@gmail.com, matthew.montebello@um.edu.mt

Abstract. The main drawback of pervasive computing is the lack of infrastructure on which ubiquitous applications should be deployed on. The deployment of the resources required by pervasive computing require expensive hardware. These considerable disadvantages led to the area of wearable computing. In this poster we briefly describe our work in the area of wearable computing, were we apply some major concepts of ubiquitous commerce to achieve a generic e-commerce framework that can be used by any wearable device.

1 Introduction

Wearable computing facilitates a new form of humancomputer interaction comprising a small bodyworn computer (e.g. user programmable device) that is always on and always ready and accessible. In this regard, the new computational framework differs from that of hand held devices, laptop computers and personal digital assistants (PDAs) [WeCo]. Mobile computing means that the computing device is not continuously connected to the base or central network. Mobile devices include PDAs, laptop computers, and many of todays cell phones (aptly called smart phones) [TRTX]. Ubiquitous Commerce is a form of commerce in which devices embedded in all terminals and goods are interworked [DigT]. Hence the primary objective of this research is to amalgamate the fields of Wearable Computing, Mobile Computing and Ubiquitous Commerce together. This will formulate new theories that will be used to construct an innovative framework that will run on any type of wearable device. This framework will assist users in performing any complex e-commerce transactions from their own wearable device.

2 Framework

Figure 1 illustrates how the main components of this framework will be connected to each other. The marketplace portal will allow suppliers to set up a profile. The system will publish the selected supplier services to users of wearable devices. The suppliers will set up their profile through a web interface. The web interface will subsequently trigger an engine which will generate a dynamic set of services...
for the particular supplier. Subsequently these dynamic set of services will be forwarded to the chosen location based servers.

The task of these location based servers is to advertise a variety of supplier services to users of wearable devices. Users of wearable devices will need to have a client based application which will handle any communication with the location based servers. Once a transaction between a client and a supplier service has been completed the location based server will notify the marketplace portal that a particular transaction was successfully completed so that it updates the suppliers account and notifies the supplier directly through email.

2.1 Marketplace Portal

The Marketplace portal will be the main component of the system. All the other components have to either communicate with this component or are triggered from this component. The marketplace portal will mainly consist of a web interface which suppliers will use to sign up to the wearable commerce marketplace, define the services which they require and define which location based server will propagate their service. Suppliers will also use this portal for statistics gathering and adjustments to their profile. The customers will use this portal to download the wearable software application that will be used to interact with the location based servers and to identify what transactions they conducted through this wearable e-commerce marketplace.

The marketplace portal will also initiate the engine that will create the supplier services. The communication with the location based servers will also be handled by this portal. Finally this marketplace will also have an admin interface for the
administrators of this system. This will give admin users the possibility of tuning the functionalities of the portal and location based servers.

2.2 Dynamic Service Creation Engine

The service creation engine will be launched exactly after that a supplier activates his subscription to the wearable e-commerce system. It will use the information entered by the supplier to create the required services. This engine will also be initiated when the supplier or administrator of the system perform some changes to the existing services.

Once that the engine dynamically creates the required services, it will determine which location based servers will publish the service and consequently it will transmit the service to the selected location based servers. At this stage the location based servers will plug in the newly received service until they receive further modification requests from the engine.

This engine will have to be as extendable as possible because it will need to cater for new behaviors that might be inserted into the marketplace portal. This means that if the marketplace administrators decide to upgrade the system with new features then only the engine and the relevant user interfaces will have to be upgraded. From the location based server’s side there will be no updates. The client application software will have to be upgraded to deal with the new behaviors of the wearable commerce server.

2.3 Wearable Client Application

The primary objective of this application is to communicate with the location based servers. This is the only technique which can be used to transmit data to the location based servers and therefore have access to supplier services. The wearable client application will fulfill all services that will be accessible from the marketplace portal. The services will only be available when the wearable application will be in the area that is covered by a particular location based server. Only the services that are registered to the particular location based server will be published to the wearable client applications in that region. The wearable client application will send various type of messages to the location based servers because it will need to present the user with the essential information.

The location based servers will periodically advertise their existence through message sending. Once that a client application enters an area covered by a location based server the communication between the location based server and the wearable client application will be initiated. The location based server will send the services according to the clients location except for when a client requests for specific details. (e.g. if a client is in front of a music store the location based service will send the details of that music store).

When a client enters a particular store the location based server will send the details of that particular store together with competitors information. Once in a store the user might request the location based server for characteristics of a specific product without querying the shop attendant. The location based server
will also supply the pricing of the competition. The client may decide to buy a specific item from a store without notifying the shop attendant. Once that a client wearable application exits the area covered by the location based server the connection will be terminated.

2.4 Location Based Servers

The location based server will be placed in a strategic business location. The primary idea is to cover a business district such that users will conduct business through the use of their wearable device which will be in constant contact with the location based server that is taking care of that specific business area.

Each location based server will have to be installed on a machine which will be located in a distinct business district. The location based server will need to have the location details of each shop that is located in that business district. The compulsory details that the location based server will need to know about each store in a particular business district include the name and the global positioning of that store.

All this information will be entered during the installation of the location based server. Afterwards the location based server will need to register its existence to the marketplace portal. This step is crucial because at this stage the marketplace portal will send the details of the services and suppliers that will be advertised by the just registered location based server.

3 Conclusions

This work takes e-commerce technology to a new level, through the deployment of major principles of pervasive computing on a wearable computing architecture. It is expected that a situation similar to that already experienced with mobile phones will occur for wearable computing. This means that wearable devices will become an integral part of every day life such as going to eat, shopping and all sorts of daily situations. Areas such as sports will require wearable devices because such devices will be capable of offering a massive amount of information concerning the event that is being followed [Tsu04]. This functionality will also be extended to museums, amusement parks, stores, shops, bars, restaurants and public services.

References


HeDLa: A Strongly Typed, Component-Based Embedded Hardware Description Language

Gordon J. Pace

Department of Computer Science, University of Malta
gordon.pace@um.edu.mt

Abstract. Over the past years, various techniques for the embedding of hardware description languages within general purpose languages have been developed and explored. In particular, numerous HDLs embedded in strongly typed functional languages have been developed and used for different applications. A common trait of most of these languages is that they treat hardware components as functions or relations between the inputs and outputs of the circuit. The alternative view, of viewing the circuits as components which can be instantiated, composed and transformed has been a relatively less well explored area in this context. In this paper we present HeDLa, a component-based hardware description language embedded in Haskell, and show how features such as strong-typing and higher-order functions enable us to design and compose circuits in a safer and more abstract fashion. Furthermore, the component-based approach allows access to circuit structure directly, enabling us to reason about non-functional aspects of the component, such as placement, area and power consumption more easily. Finally, we discuss some initial experiments in multi-level simulation of circuits which enable testing and more effective simulation of large circuits.

1 Introduction

The utility of domain-specific languages has been widely accepted by the programming language design community and has increased in popularity over the past few decades. General purpose languages are excellent media to express algorithms that work in a variety of contexts, but the need for a context-dependent, domain-specific core language is extremely useful when designing solutions for a constrained problem-domain. One area which saw the rise of various (text-based) domain-specific languages in the seventies and eighties was that of hardware description and design. Languages such as Verilog [Ope93] and VHDL [LMS86] enabled modular design of hardware, allowing reuse of designed components, and abstraction in hardware design. Such languages recognised the need for different interpretations of hardware descriptions, allowing both structural descriptions (in which components are described in terms of their constituent parts, decomposing them into subcomponents until the gate or transistor level is reached) and behavioural descriptions (in which the behaviour of a component is given algorithmically). The former was necessary to allow actual circuit instantiation
based on the descriptions given in the language, while the latter enabled easier testing, more efficient simulation, and better structured design approaches. Due to the finite nature of hardware, algorithmic constructs, especially ones for loops were not allowed in structural descriptions, hence ensuring that every structural description is a finite one (that is, terminates) simply by ensuring that a structure does not (transitively) refer to a copy of itself — hence resulting in modular descriptions which are in the shape of a tree (or a directed acyclic graph if one considers reuse of modules). This restriction, however reasonable it may seem, did not allow algorithmic descriptions of regular shaped circuits in a concise manner through the use of loops, recursion and conditional selection based on values of static (compile-time) parameters. Certain such circuits could be described through the use of arrays of data shifted in smart ways to ensure that a repetative structure is produced. A classic textbook example would be that the array of carry-outs of the full-adders in a serial-carry adder can be shifted to the right and passed on as the carry-ins of the full-adders. However, more complex regular circuit structures, such as trees, and non-constant shifts of the arrays are simply not possible, or possible only in a convoluted unintelligible manner.

This restriction was addressed in various tools, providing non-standard extensions to VHDL and Verilog, allowing certain general-purpose constructs to be used in the circuit structural descriptions. Different tools provided different languages, some simply allowing regular repetition or structure-composition formats, others providing an essentially full-blown meta-language sitting above the structural descriptions. This two-tier meta-language approach worked well in the description of large regular circuits, and as long as the meta-program terminated, it produced a finite circuit that could be analysed, tested, verified and fabricated.

This two-tier approach is, however, useful in most other domain-specific languages, and was independently adopted in other areas. This approach of building a Turing-complete meta-language above the domain-specific language has various drawbacks. Domain-specific language design is clearly more challenging with this approach, since the language designer must also look into the design and implementation of the general-purpose meta-language. Furthermore, the end-users are expected to learn different syntax, different languages each with their different quirks, and in some cases different programming paradigms, for different domain-specific languages they are working in.

Embedded languages, a technique developed in the programming language community, was found to be an excellent approach to alleviate a number of these problems. In embedded languages, one builds a domain-specific library using a general-purpose language, called the host language, enabling a programmer to describe ‘programs’ in the domain-specific language to appear as though they were part of the host language. The programmer may then generate, analyse and manipulate programs in the domain-specific programs as though they were part of the host language itself. The host language thus automatically becomes a meta-language for the embedded language. Clearly, the more flexible and high-level the host language and its syntax are, the more difficult it becomes to
distinguish where the domain-specific program ends and where the rest of the code starts. From the language designer’s point of view, the main advantage of designing a domain-specific language and embedding it within a host language is that he or she needs not reinvent the wheel and create a new general-purpose language, while from the end-user’s perspective, the main advantage is that the meta-language is a standard language with which he or she may already be familiar and knowledge of which goes beyond the use of the domain-specific language. Some argue that embedded languages are nothing more than domain-specific libraries, and certainly, the dividing line between a domain-specific library, and a domain-specific language is blurred. However, one may look at embedded languages as a structuring principle for domain-specific libraries, to provide structures allowing first-order programs in the domain-specific language to be written without resorting to the host language.

Over the past years, various embedded hardware description languages have been designed and used (see, for instance Lava [CSS03], Hydra [O’D96], Hawk [DLC99], reFlEcT [MO06], Wired [ACS05] and Dual-Eval [WAHR04]). In particular, functional languages have proved to be particularly suited for this purpose. A common trait of most of these languages is that they treat hardware components as functions or relations between the inputs and outputs of the circuit. The alternative view, of viewing the circuits as components which can be instantiated, composed and transformed has been a relatively less well explored area in this context [WAHR04,ACS05]. Circuits are produced as a result of function calls in the host language, and a common challenge in most embedded hardware description languages is that of having access to the structure induced from the description of the circuit in the host-language. For example, a circuit induced through a linear chain of recursive calls could be viewed as a nested chain of similar components, which information can be used to aid reasoning about the circuit or for inducing hints for placement tools. However, without additional machinery, this information cannot be created and accessed directly. Being able to refer to such blocks, would also enable a hardware designer to reason about non-functional aspects such as power consumption and wire lengths of such compound components, and use other standard development techniques such as refinement in the design process, all within the same language.

In this paper we present initial experiments in building a component-based hardware description language embedded in Haskell [Jon03], and show how features such as strong-typing and higher-order functions enable us to design and compose circuits in a safer and more abstract fashion. Furthermore, we discuss the use of refinement and multi-level description and simulation of circuits which enable testing and more effective simulation of large circuits. Essentially, this enriches the approach the embedded hardware description language with a design scripting language. Finally, we discuss how the component-based approach allows more direct access to circuit structure, enabling us to reason about non-functional aspects of the component, such as placement, area and power consumption more easily.
2 HeDLa: Embedding Yet Another Structural HDL in Haskell

2.1 Breaking Away from Circuits as Functions

When embedding a language, one would want the embedded programs to look similar in style to the rest of the code. It is thus natural to emulate the host language paradigm in the embedded language. Most probably for this reason, one finds that in most hardware description languages embedded in Haskell, circuits are described as functions from inputs to outputs. Circuit reuse simply becomes function application, as for example can be seen in the following example in Lava [CSS03]:

```haskell
halfAdder (a, b) = (s, c)
  where
    c = and2(a, b)
    s = xor2(a, b)

fullAdder (cin, (a,b)) = (s, cout)
  where
    (c1, s1) = halfAdder(a, b)
    (c2, s)  = halfAdder(s1, cin)
    cout    = or2(c1, c2)
```

In HeDLa, we take a different approach, in which circuits are objects which can be tagged, manipulated and structurally modified. Structurally describing a half-adder in terms of basic gates, and a full-adder in terms of half-adders is done using the following code:

```haskell
halfAdder = Circuit
  { name = "halfAdder"
      , inputs = ("a", "b")
      , outputs = ("s", "c")
      , description = use xor2 ("a", "b") "s"
                      & use and2 ("a", "b") "c"
  }

fullAdder = Circuit
  { name = "fullAdder"
      , inputs = ("cin", ("a", "b"))
      , outputs = ("s", "cout")
      , description = use halfAdder ("a", "b") ("s1", "c1")
                      & use halfAdder ("s1", "cin") ("s", "c2")
                      & use or2 ("c1", "c2") "c"
  }
```

Each circuit is defined as a Haskell record, with not only a Haskell function name, but also a string as a name. The wires are not viewed as Haskell parameters to the actual circuit objects, but are labelled and cross-referenced using strings. The
inputs and outputs themselves of a circuit are expressed as structures (tuples and lists) of variable names, and are referred to string names. Finally, the description of the circuit behaviour is given (in this case) as a structural description — a combination of instances of basic components and compound components defined elsewhere in the code. The use keyword enables the creation of an instance of a circuit component (with the input and output wires given as additional parameters), and these can be combined together using the \& operator.

From this example, it is clear that there is syntactic overhead in using HeDLa to describe the structure of a circuit as opposed to Lava functional-style descriptions. The primary syntactic distractions are the two names given to each component (the name of the Haskell object and the string used in the record) and the clunkiness of quoted wire names. Despite the additional syntax, the use of both the component and the wire names is particularly useful when the descriptions are exported to VHDL or other external formats, since they give a way of relating the simulation and verification with the original HeDLa descriptions. In fact, the component name is only used for this purpose.

2.2 Strongly-Typed Circuits

In HeDLa, inputs and outputs of a circuit are not directly equivalent to inputs and outputs in the Haskell sense. Instances of a circuit, take both the inputs and outputs of the circuit component being instantiated as inputs in the functional sense. Although it creates a dichotomy between the Haskell code and the embedded language code, the resulting structure is closer to VHDL and Verilog descriptions of circuit instances. To enable structuring the input as tuples and lists, Haskell type classes are used to allow different circuits to take different types of structures as inputs and outputs — in the above example, the basic gates, the half-adder and the full-adder all have different structures of inputs and outputs.

Furthermore, Haskell types are used to ensure that circuit instantiation is done in a type-safe and correct manner. The types of the circuits given and the basic gates are:

\[
\text{xor2, and2, or2 :: (Bit, Bit) } \Rightarrow \text{Bit} \\
\text{halfAdder :: (Bit, Bit) } \Rightarrow \text{Bit, Bit} \\
\text{fullAdder :: (Bit, Bit, Bit) } \Rightarrow \text{Bit, Bit}
\]

The infix parametrised type \(\Rightarrow\) is used to describe circuits taking inputs with the structure appearing as the first type parameter, returning a structure given as the second type parameter. This enables us to use Haskell type checking to ensure that all circuits are instantiated in type-safe manner.

Currently HeDLa supports only wires carrying boolean streams, but it is planned to be extended to support other streams carrying integers and other data types. Streams can be structured not only in tuples, as we have seen in the examples so far, but also using lists as shown in the example below\(^1\):

\(^1\) The function \text{combine} takes a list of circuit instances and combines them together using \&
nBitAdder :: Int -> (Bit, [Bit], [Bit]) |=> [Bit]
nBitAdder n = Circuit
  { name = "nBitAdder"
    , inputs = ("c"!0, bus "a" n, bus "b" n)
    , outputs = bus "s" n ++ ["c"!n]
    , description
      = combine
        [ use fullAdder ("c"!m, ("a"!m, "b"!m)) ("c"!(m+1), "s"!m)
          | m <- [0..n-1]
        ]
  }
where
  bus v i = map (v!) [0..i-1]
v!i = v ++ show i

2.3 Circuit Interpretations

Starting with these descriptions, one can interpret them in different ways. The most straight-forward interpretation is that of simulation, where the given circuit is (recursively) interpreted in terms of its underlying components, until the basic gates are encountered, and interpreted using a default interpretation in Haskell:

> simulate fullAdder (high, (low, high))
(low, high)

Based on the description, one can also produce VHDL output of a circuit. However, unlike embedded HDLs such as Lava and Hawk, we produce a modular description following the structure of the circuit as defined, rather than one flattened netlist with no structure. Apart from outputting to VHDL, HeDLa also allows exporting a circuit to model checkers\(^2\) to verify safety properties of circuits.

3 Behavioural Descriptions, Specifications and Refinement

3.1 Behavioural Descriptions and Simulation

The underlying gate components in HeDLa are used no differently than the compound components the user may define. The real difference is the interpretation of the components during simulation — whereas compound circuits are decomposed into their subcomponents and simulated separately, the basic gates have a behavioural semantics associated to them, which enables direct simulation. HeDLa allows the description of new components with a behavioural, as opposed to structural description. In languages such as VHDL, this necessitated the definition of a sub-language for the description of behavioural code. In embedding HeDLa in Haskell, we reuse the host language to describe behaviour of

\(^2\) Currently we support only SMV as a model-checker.
circuits. For instance, the behaviour of a two-input xor gate can be defined in the following manner:

```haskell
xor2 = Circuit
    { name = "xor2"
    , inputs = ("a", "b")
    , outputs = "z"
    , description = behaviour (\(a,b) \rightarrow a \neq b)
    }
```

The simulation of circuits simply uses the behavioural description to interpret the gate when it is encountered in a circuit. Behavioural descriptions of higher level circuits are frequently used in standard HDLs to test the structural description, by simulating the two side-by-side for different inputs. Rather than keeping different descriptions for the different functionalities of a circuit separately, we enable dual descriptions of circuits, with can be simulated in either structural or behavioural modes:

```haskell
fullAdder = Circuit
    { name = "fullAdder"
    , inputs = ("cin", ("a", "b"))
    , outputs = ("cout", "s")
    , description
        = use halfAdder ("a", "b") ("c1", "s1")
        & use halfAdder ("s1", "cin") ("c2", "s")
        & use or2 ("c1", "c2") "cout"
    } 'setBehaviour' (\(cin, (a, b)) \rightarrow
    let (cin', (a', b')) = (bit2int cin, (bit2int a, bit2int b))
    in (cin' + a' + b' >= 2, isOdd (cin' + a' + b'))
```

By considering the behavioural description to be the specification, and the structural to be the implementation, we enable testing of circuits through the use of QuickCheck [CH00]. Furthermore, in larger circuits, the designer can switch between modes of the constituent subcircuits to enable more efficient simulation through the use of the specification, rather than the structural description of large, but trusted subcomponents.

### 3.2 Observer-Based Testing and Verification

One strength of behavioural descriptions is that they can not only be used as specifications against which to run tests, but also used to simulate the actual circuit directly (as opposed through the interpretation of its subcomponents). This means that the specification has to be a deterministic one, to enable the calculation of the correct outputs based on the inputs. Furthermore, sometimes it is easier to write a specification checking that the inputs and outputs are correct. HeDLa supports the use of observers, which given the input and output of a circuit, return a single boolean value stating whether the circuit is working.
correctly. Both structural and behavioural observers can be defined, with behavioural observers used in testing, while structural observers can be used both in testing and when producing output to model-checkers.

The following example shows a behavioural observer for a full-adder, asserting that the interpretation of the output as a two-bit number gives the same value as the addition of the three input bits:

```haskell
fullAdder = Circuit
{ name = "fullAdder"
  , inputs = ("cin", ("a", "b"))
  , outputs = ("cout", "s")
  , description
    = use halfAdder ("a", "b") ("c1", "s1")
    & use halfAdder ("s1", "cin") ("c2", "s")
    & use or2 ("c1", "c2") "cout"
    } 'setBehaviouralObserver' (\((cin, (a, b)), (cout, sum)) ->
      let (cin', (a', b')) = (bit2int cin, (bit2int a, bit2int b))
          (cout', sum') = (bit2int cout, bit2int sum)
      in cin' + a' + b' == 2 * cout' + sum'
)
```

### 3.3 Multi-Level Refinement and Data Refinement

Frequently, behavioural descriptions and observers use different data representations than the structural description, requiring translation to and from the different representations. For instance, in the case of an $n$-bit adder, the concrete inputs (one carry-in bit, and two bitstrings of length $n$) can be translated into a more abstract interpretation — as a list of three numbers. The outputs can be translated back from a number into $n + 1$ bits. Once this data refinement is specified, the behaviour is defined to work on the abstract interpretation of the inputs, and producing abstract outputs. In this case, the specification simply becomes the Haskell function `sum`:

```haskell
nBitAdder n = Circuit
{ ...
} 'usingDataRefinement'
  ( \((c, (as, bs)) -> [bit2int c, bits2int as, bits2int bs]
    , int2bits (n+1) )
  ) 'setBehaviour' sum
```

Rather than constrain descriptions to just a specification and an implementation, we are currently experimenting with extending HeDLa with multi-level refinement, to enable stepwise refinement of a component for design, testing and verification. In this context, the use of explicit data-refinement specification is much more useful, since it allows the designer to go up and down the refinement layers in multiple steps.
3.4 Non-Functional Circuit Properties

The major advantage of using a structure, rather than a functional view of circuits, is that we maintain a hierarchical view of the circuits and their components. Furthermore, one can add circuit information as the circuit is constructed. We are currently looking into the use of these features for the description of non-functional properties of circuits. One area of application is adding placement information, or hints through the use of combinators for combining circuits. Consider the operator \( -\rightarrow - \), which connects two circuits next to each other as shown in figure 1. An implementation, of such an operator can be written as follows\(^3\):

\[
\text{c1} \rightarrow\text{c2} = \\
\text{let } (\text{left1}, \text{up1}) = \text{inputs c1} \\
(\text{left2}, \text{up2}) = \text{inputs c2} \\
(\text{right1}, \text{down1}) = \text{outputs c1} \\
(\text{right2}, \text{down2}) = \text{outputs c2} \\
\text{in } \text{Circuit} \\
\{} \\
\quad \text{name } = \text{name c1 ++ "-\rightarrow-" ++ name c2} \\
\quad \text{inputs } = (\text{left1}, (\text{up1}, \text{up2})) \\
\quad \text{outputs } = (\text{right2}, (\text{down1}, \text{down2})) \\
\quad \text{description } = \text{use c1 (inputs c1) (outputs c1)} \\
\quad \quad \quad \& \text{ use wire right1 left1} \\
\quad \quad \quad \& \text{ use c2 (inputs c2) (outputs c2)} \\
\}
\]

As it stands, the only information we maintain is that of how wires are connected. However, one can easily tag the component \( \text{c1} \) to lie to the right of \( \text{c2} \). In this manner, we can actually produce concrete placement and wire length information from such a description.

4 Future Work and Conclusions

In this paper, we have presented the basic functionality of an experimental structural hardware description language embedded in Haskell. Various HDLs embedded in Haskell and other functional languages have been developed and used

\(^3\) This implementation allows for any pair of structure of wires as input and as output, but assumes that there are no name clashes between wire names in the subcircuits. Note that the \text{wire} circuit simply connects the input to the output.
over the past years. Languages such as Lava [CSS03], Hawk [DLC99] and Hydra [O'D96] have followed a strictly functional view of circuits — circuits appear as functions in the host language, and their inputs and outputs are identical to the inputs and outputs of the functions. The descriptions tend to be cleaner in these languages, but lose information about the structure due to the functional view of the circuits. The challenge to incorporate information about the structure of the circuit description has spawned a number of other embedded languages. Wired [ACS05] uses a component-based approach to circuits, with combinators used to describe circuits, keeping structural and layout information. As opposed to our approach, the combinators lie at the core of the language making it much better at describing information about layout, though less oriented towards actual circuit behavioural description. Just like Wired, Dual-Eval [WAHR04] is also similar to HeDLa in that it takes a component-based approach to circuit description, with explicit wire names and connections. reFLect uses a functional meta-language to embed a HDL, with the use of language reflection features used to have access to the circuit generators themselves. HeDLa loses some of this information — for instance, when function calls are used to generate families of circuit instances (in the description field). On the other hand, we have access to the nested structure of circuits defined explicitly.

In HeDLa, we have tried to find a balance between using a component description of circuits, but still keeping as close as possible to the functional view. There are still various issues we are working on resolving in HeDLa. On one hand, we are looking into extending the functionality of the language, by introducing multi-level refinement, and looking into techniques to enable the user to add general non-functional features. In particular, we plan to look at the use of HeDLa to reason about placement, wire length and area analysis. In this paper, we have only used small examples using combinational circuits. Although HeDLa can also handle sequential circuits, the interaction between the behavioural descriptions and observers with sequential circuits still needs to be refined, to enable us to test run the language on a number of large case studies thus assessing how effective it is in the design of large hardware systems.

References


Embedding a Hardware Description Language in a Functional Meta-Programming Language

Gordon J. Pace and Christian Tabone

Department of Computer Science, University of Malta
{gordon.pace|christian.tabone}@um.edu.mt

Abstract. General purpose functional languages such as Haskell, have been widely used as host languages for the embedding of domain specific languages. In particular, various hardware description languages have been successfully embedded in Haskell and other functional languages. More recently, meta-programming languages have also started being used for the embedding of such languages, where the meta-language features allow us to access the structure of data objects in a shallow-style embedding, while retaining the characteristics of a deep-embedding. In this paper, we discuss the application of meta-functional languages for the embedding of a hardware description language, based on \texttt{reFLect}, a functional meta-language which provides an alternative approach for embedding a hardware description language by means of built-in reflection features. Through the use of code quotation and pattern matching, we use \texttt{reFLect} to build a framework through which we can access the structure of our circuits by means of reflection.

1 Introduction

Designing and developing a new language for a specific domain, presents various challenges. Not only does one need to identify the basic underlying domain-specific constructs, but if the language will be used for writing substantial programs, one has to enhance the language with other programming features, such as module definition and structures to handle loops, conditionals and composition. Furthermore, one has to define a syntax, and write a suite of tools for the language — parsers, compilers, interpreters, etc — before it can be used. One alternative technique that has been explored in the literature is that of embedding the domain-specific language inside a general purpose language, borrowing its syntax, tools and most of the programming operators. The embedded language is usually developed simply as a library in the host language, thus effectively inheriting all its features and infrastructure.

Modern functional languages have been shown to excel as host languages for the embedding of specific domain languages. Features such as strong typing, lazy evaluation, pattern matching and higher-order functions, all make them ideal for the development of small languages within them [Hud96]. One domain for which functional languages have been extensively used to embed languages in is that of hardware description and design [BCSS98,LLC99,BWAH97,O'D06,ACS05]. An
overview of the use of functional languages in hardware design can be found in [She05].

The need for hardware description languages (HDL) emerged as the size and complexity of circuits increased beyond the point where the manual design of circuit systems became unfeasible, creating a need for an infrastructure to describe circuits textually and enable reuse and instantiation. This led to the development of languages such as VHDL [LMS86]. Since various large circuits have regular structures, various extensions and tools for these HDLs appeared, providing features such as iterative descriptions, and static (compile-time) recursion in the description of circuits. These extensions provided a simple meta-language, sitting above the basic HDL, enabling the algorithmic generation of regular circuits through a two-stage language. One main advantage in embedding a HDL in a general purpose programming language, is that these meta-language features essentially comes for free — it is simply the host language. Furthermore, having a full meta-language enables the analysis and transformation of circuits, enables general manipulation of circuits as with any other data objects. Therefore we can not only generate circuits, but also analyse (such as static information gathering, simulation, testing, verification), transform (such as retiming) and interpret (such as netlist generation).

Functional programming languages have proved to be excellent vehicles for embedding languages in a two-stage language approach, enabling allowing access to the HDL description, but do not offer access to the host language code creating the domain-specific objects. This may be useful since certain structuring information inherent in the control structure of the code generating the domain-specific program may be useful in its analysis. Recently, the use of meta-programming techniques for the embedding of HDLs has started to be explored [MO06,Tah06,O’D04]. A meta-programming language enables the development of programs that are able to compose or manipulate other programs or even themselves at runtime, through reflection.

In this paper, we explore the use of reFLect, a meta-programming language, to embed a hardware description language in such a manner that we can not only access and manipulate the circuit descriptions, but also the circuit generators themselves. We plan to use these features to access and control the structure of the circuit generated. In particular, in the future, we plan to use this to optimise circuits produced by hardware compilers, maintaining a compositional view of the compiler, but at the same time having access to information as to which parts of the circuits resulted from which features of the compiled language.

2 Functional Meta-Programming in reFLect

reFLect [MO06] was developed by Intel, based on the functional language FL, but extended with reflection features. reFLect is the main programming language used with the Forte tool [SJO+05]; a hardware verification system used by Intel. Forte together with reFLect was purposely developed for the development of applications in hardware design and verification, and is mostly used for access
to model checking, decision making algorithms and theorem provers for hardware analysis.

The reFLect language is a strongly-typed functional language with added meta-
programming features, such as quotaion and antiquotation constructs used to
compose or decompose expressions written in the reFLect language itself. This
provides a form of reflection within a typed functional paradigm setting. The
reFLect meta-programming constructs provides the developer with an access to
the structure of programs written in the whole of the reFLect language itself
as data objects. Quoted program expressions are considered to be of a special
type term, representing the abstract syntax tree of the program expression.
Traditional pattern matching can be applied on the type term, allowing uneval-
uated expressions to be inspected and interpreted according to the developer’s
requirements. By combining the pattern matching mechanism with the quotation
features, the developer is also able to modify or transform the quoted expression
at runtime before evaluation. An in-depth overview of reFLect can be found in
[GMO06].

2.1 Reflection Operators in reFLect

Expressions in reFLect are quoted by enclosing them between {| and |}. The
whole expression is typed as a term, denoting the abstract syntax tree for the
enclosed expression. For instance, consider the expression 1 + 2. Normal func-
tional features would evaluate this expression resulting to be semantically equal
to 3, and there is no way one can distinguish between the two. However, the
application of quotation marks around the expression, {| 1 + 2 |}, halts the
evaluation by capturing its syntax tree. Note that, the expression {| 1 + 2 |}
is therefore semantically, and not just syntactically, different from {| 3 |}.
The antiquotation construct is expressed by the prefix operator ′. The antiquo-
tation mechanism essentially raises its operand one level outside the quotation
marks. Antiquoted terms always appear within quotations, and have two main
applications. Firstly, it is usually to embed a quoted term within another term.
To avoid nested quotations, one uses the antiquotation operator to splice one
abstract syntax tree into another, thus allowing the construction of terms. For
example, given a term, the function below constructs a new term, representing
the addition of the the original term with 1.

let incTerm a = {| 1 + ′a |};

A typical functional call would be as follows, where the input should also be of
type term.

incTerm {| 2 + 3 |};

In reFLect, this term would reduce to the expression to {| 1 + (2 + 3) |}.
Another application of antiquotation is term decomposition, and used to enable
pattern matching on the abstract syntax tree. For example, the function below
decomposes the given term into the two operands applied to the addition oper-
ator, binding the left term to the variable \( x \) and the right term to the variable \( y \).

\[
\text{let decompose } \{ | x + y | \} = (x, y);
\]

Consider, for example, pattern matching with \( \{ | 5 \times 4 + 2 \times 3 | \} \) — \( x \) would be bound to \( \{ | 5 \times 4 | \} \) and \( y \) to \( \{ | 2 \times 3 | \} \). Note how the antiquote is needed to extract the sub-expression as a term. If the function had to be defined without antiquotes using the pattern \( \{ | x + y | \} \), the variables \( x \) and \( y \) would be non-binding, thus this would match the expression \( \{ | x + y | \} \) literally.

The \textit{reFLect} language offers a number of built-in evaluation functions, to allow total control over the evaluation of the terms being constructed. The most elementary is the \textbf{eval} function, which is used to evaluate a given term, and returns the result as a quotation. The \textbf{value} function is similar, since it also evaluates the given term, but the result is returned as the specified type. A \textbf{lift} function is also available, and it can be applied to any \textit{reFLect} expression. This works by first evaluating the given expression and then by applying quotation marks around the resulting expression, conclusively lifting the evaluated expression to a higher level of quotations.

### 2.2 Embedding Languages in \textit{reFLect}

The \textit{reFLect} language, together with the meta-functional features that it offers, provides interesting grounds for the implementation of hardware description languages. Typically, when embedding a language, a deep-embedding is required, since one would want not only to generate programs, but given them different interpretations as may be required, and have access to the underlying syntax of the domain-specific language.

Since, in a meta-programming language, one may quote language constructs, and antiquote terms, one has access to the actual programs as data objects. In \textit{reFLect}, the possibility to pattern match over programs also gives the possibility to look at the structure of an expression. Consequently, in a language like \textit{reFLect} one can build a deep embedding mechanism, simply by using quotations and antiquotations to represent the embedded language using the term datatype, over a simple shallow embedding of the embedded language. Term manipulation is easily achieved through the use of quotations and antiquotations. The ability to directly control the terms of quoted expression, can be applied to expressions representing elements within a circuit model.

Furthermore, using this style of embedding and nested quotations, one can actually reason about marked (quoted) blocks of code hence giving access to the structure of generator of the domain-specific program, effectively enabling reasoning about the embedded language itself at a higher level of abstraction.
3 Embedding a HDL in \textit{reFLect}

3.1 Shallow Descriptions

The simplest way to develop an embedded hardware description language is to define a number of functions that represent the circuits’ behaviour. If one uses the boolean values \texttt{true} and \texttt{false} to represent the circuit constant streams \texttt{high} and \texttt{low}, the description of the primitive and-gate will simply be an application of the built-in conjunction, thus modelling the logical behaviour of the hardware. The evaluation of such functions, when applied to a set of inputs, would result in the simulation of the circuit. Such a shallow embedding can be implemented in a straightforward manner in \textit{reFLect}. For the sake of simplicity, we consider two basic gates, and-gates, and inverters.

\begin{verbatim}
let and2 (x, y) = x AND y;
let inv x = NOT x;
\end{verbatim}

In a shallow embedding approach the circuits are represented as programs within the host language, thus circuits can be described using the more simple functions that have already been defined. Note that the use of the in-built and-gate is no different from the use of the user-defined components:

\begin{verbatim}
let or2 (x, y) = inv (and2 (inv x, inv y));
let xor2 (x, y) = or2 (and2 (x, inv y), and2 (inv x, y));
let mux (s, (x, y)) = or2 (and2 (s, y), and2 (inv s, x));
\end{verbatim}

The shallow embedding approach offers a straightforward technique for the implementation of a hardware description language. There is no need for the programmer to learn new syntax or programming paradigm since these are inherited directly from the host language, and the default interpretation of the embedded programs, in this case that of simulation, is achieved directly through the interpreter of the host language itself. Nevertheless, as already discussed, through such a shallow embedding one loses all information about the structure of the circuit, and unless the basic gates are overloaded with other interpretations, one loses the option to apply non-standard interpretations of a circuit.

3.2 Using Reflection for a Deep Embedding

Usually, in a language without reflection, to achieve a deep embedding of an embedded language, one creates a datatype to which descriptions are reduced. Using the reflection features, one can take a shallow embedding, as described in the previous section, and quote the circuit descriptions, thus maintaining the structure of the circuit using the structure use of the shallow embedding in \textit{reFLect}. Thanks to pattern-matching on terms in \textit{reFLect}, one can inspect and traverse such circuit descriptions within the language. Signals can thus consist of either (shallow) values, corresponding to booleans, or (deep) structures, corresponding to terms. In the following datatype definition, \texttt{Value} corresponds to the raw boolean value, while \texttt{Structure} represents the whole
structure of the circuit operations given as a term. Note that the latter can be
evaluated to result in the actual simulation style interpretation of a boolean
value.

lettype signal = Value bool | Structure term;

The primitive gates now have two possible behaviours — the shallow simula-
tion semantics, and the deep quoted version of the shallow interpretation. Using
pattern matching one can distinguish between boolean values and structures:

forward Declare {inv :: signal -> signal};
let inv (Value a) = Value (NOT a)
/\ inv (Structure a) = Structure { inv 'a'};

forward Declare {and2 :: (signal, signal) -> signal};
let and2 (Value a, Value b) = Value (a AND b)
/\ and2 (Structure a, Structure b) = Structure { and2 ('a', 'b') };

Other primitive gates are defined using functions similar to the above, which
can be presented to the end user to be used for other circuit descriptions. The
constant expressions high and low are defined for Value T and Value F respec-
tively. Additional constants are also defined to hide quotation constructs from
the end user.

let high = Value T;
let low = Value F;

let shigh = Structure { high |
let slow = Structure { low |

The structure embedded in the above manner enables circuits to be described
in a functional style. Furthermore, the use of user-defined blocks is identical
to the use of the basic primitive gates. Consider the following definition of a
multiplexer:

let mux (s, (a, b)) = or2 (and2 (s, b), and2 (inv s, a));

Such a description can be interpreted in different ways. Passing a boolean value,
one obtains the result of simulating the circuit, while passing a structure, one
obtains the internal structure of the multiplexer circuit.

: mux (high, (low, high));
: high;

: mux (shigh, (slow, shigh));
: Structure { or2 (and2 (high, high), and2 (inv high, low)) |

An alternative approach, which we are also considering is the overloading of
high and low, then adding simulation, and structure creation functions, which
would enforce one, or the other interpretation of high and low.
3.3 Representing Signals

A crucial design decision that is needed when developing a HDL is the way circuits inputs and outputs are considered to be structured [CP07]. In the previous section, we have presented the signals used by the circuit descriptions as structure of signals, similar to how signals are represented in Lava [BCSS98]. In other words, an and-gate takes a pair of two wires as input, each carrying a boolean signal. Another form of representation, the one adopted in Hawk [LLC99], is to consider only circuits with one input and output wire, but carrying a signal of structures upon it. Contrast the Lava and Hawk types of a two-input and-gate below:

// Signals in Lava
and2 :: (Signal bool, Signal bool) -> Signal bool

// Signals in Hawk
and2 :: Signal (bool, bool) -> Signal bool

Currently, we are using the signal of structures representation, primarily since it simplifies language design (although not necessarily language usage). An advantage of this representation is that all circuits defined in a language using this representation will always have the same type — taking a single input and producing a single output. This makes the design much cleaner, and the interpretations work seamlessly even when describing complex circuits built from smaller circuit descriptions. On the other hand, the user has to handle the wrapping and unwrapping of the signal type whenever the inner vector values are required. For this we provide functions to convert the signal structure back and forth to the structure values.

// From signal values to signal structure
zipp :: (Signal bool, Signal bool) -> Signal (bool, bool)

// From signal structure to signal values
unzipp :: Signal (bool, bool) -> (Signal bool, Signal bool)

3.4 Marking Blocks in Circuits

In reReflect, as in most other HDLs, one views and defines circuits as functions. As a circuit description is unfolded, all the internal structure is lost, and all that remains is a netlist of interconnected gates. To enable marking such sub-components inside a circuit, we introduce the concept of a block, which a hardware designer may use at will. Such blocks are used in netlist generation, and are planned to be used also in other non-functional features of circuits we plan to implement, including modular verification, placement and local circuit optimisation. For example, one may mark a halfAdder as a block, and then use two instances to define a fullAdder, which may itself be marked as a block (thus containing two sub-blocks inside).
When the abstract circuit description corresponds to a good layout, or describes together related components, preserving such information can be useful. Adding block information to the whole structure of the circuit, adds a higher level of abstraction over the circuit description, enabling not only the possibility to reason about the structure in terms of primitive gates, but also in terms of blocks. For instance, information gathering functions could be defined to count full-adders or half-adders, or any other block. The placement of circuits will also benefit, since this can be organised into blocks, hence decreasing the level of complexity. Block information is handled by the meta-programming features of reFLect by using nested quotations to represent levels of a blocked circuit structure. A function block is defined to create a lambda expression equivalent to the given function, which is then lifted with a higher level of quotation marks, marking the lambda expression as a block.

```
let multiplexer = block mux;
```

The circuit multiplexer can now be used in the same manner as mux, but with an extra level of nesting being automatically added to enable us to identify blocks as we traverse a circuit. One can also name a block through an extra string parameter which is used in netlist generation.

3.5 Circuit Interpretations

Although the underlying interpretation of a circuit, as we develop it in our HDL is that of simulation one can provide various other interpretations of a circuit description.

**Simulation:** The simulation interpretation works similar to how a shallow-style embedding operates. Since reflection is used to maintain control over the structure, the simulation is achieved by the reFLect interpreter, thus an interpretation function is not required. Therefore, a quick simulation can be achieved by evaluating a circuit description using raw values as inputs. However, if the structure is retained by the use of structured inputs an evaluation function is required to simulate the structure. This simulation function does not interpret the structure but rather it handles the signal structure before applying the built-in evaluation function of reFLect.

**Information gathering:** Information (such as a gate or block count) can easily be gathered about a circuit using pattern matching functions, which follow through quoted circuits to identify sub-circuits and evaluate the information accordingly.

**Netlist generation:** To enable outputting a circuit description in a format which can be used by external tools, it is of utmost importance to be able to generate a netlist of a circuit description in reFLect. The default description is a flat netlist, which does not take into account blocks. One way blocks may be used is to modularise descriptions, giving a separate description of sub-circuits marked as blocks, and referring to that description when used. In the future we also plan to use blocks to mark primitive components for reducing the description into a netlist for a particular gate technology.
Postscript generation: Descriptions of circuits can also be translated into basic Postscript figures, marking labelled blocks for reference. We are currently exploring the use of placement operators in the language, which would also affect the presentation in Postscript. Furthermore, sharing of circuits creates additional complications which still have to be resolved.

3.6 A Illustrative Example: Serial Carry Adders

In this section, we will present the development of an n-bit serial carry adder. We start off by defining a half-adder. Note that, since the signal carrying pairs is passed on to the underlying gates, the the unzipping of the inputs into two separate signals is not required in this case.

```
let halfAdder a_b =
  let sum = xor2 a_b in
  let carry = and2 a_b in
  zipp (sum, carry);
```

Next we declare the function `halfAdder` as a block, using the function `namedBlock`:

```
let halfAdder = namedBlock "halfAdder" halfAdder;
```

Based on the description of the half-adder, we can now define a full-adder circuit structurally, also declaring it as a block. Note that in this case, the circuit designer has to handle the wrapping and unwrapping of the signal structure explicitly.

```
let fullAdder inps =
  val (carryIn, a_b) = unzipp2 inps in
  val (sum1, carry1) = unzipp2 (halfAdder a_b) in
  val (sum, carry2) = unzipp2 (halfAdder (zipp (carryIn, sum1))) in
  let carryOut = xor2 (zipp (carry1, carry2)) inv
              zipp (sum, carryOut);

let fullAdder = block "fullAdder" fullAdder;
```

Finally, we can define an n-bit adder is defined as a recursive function:

```
letrec nBitAdderAux (carryIn, ([], [])) = ([], carryIn)
/\ nBitAdderAux (carryIn, (a:as, b:bs)) =
  let inps = zipp (carryIn, zipp (a, b)) in
  val (sum, carry) = unzipp2 (fullAdder inps) in
  val (sums, carryOut) = nBitAdderAux (carry, (as,bs)) in
  (sum:sums, carryOut);

let nBitAdder cin_as_bs =
  let (cin, as bs) = unzipp2 cin_as bs in
let rec nBitAdderAux (carryIn, ([], [])) = ([], carryIn) /\ nBitAdderAux (carryIn, (a:as, b:bs)) = 
    let inp = zipp (carryIn, zipp (a, b)) in 
    val (sum, carry) = unzipps (fullAdder inp) in 
    val inps= zipp2 (carry, zipp2 (zipps as, zipps bs)) in 
    val (sums, carryOut) = nBitAdderBlock inps in 
    (sum:sums, carryOut);

let nBitAdder cin_as bs = 
    let (cin, as_bh) = unzipps cin_as bs in 
    let (as, bs) = unzipps as bs in 
    zipp2 (nBitAdderAux (cin, (unzipps as, unzipps bs)))

let nBitAdderBlock = block nBitAdder

4 Related Work

HDL implementations like Lava [BCSS98], Hydra [O'D06] and Hawk [LLC99], differ from the work presented in this paper, since these have been developed using the deep embedding technique within the functional language Haskell, while our approach is that of using reflection within reFLect as a replacement for deep embedding. Deep embedding allows the developer to provide multiple semantically interpretations of the defined circuits, which is clearly seen in Lava, Hydra and Hawk. These HDLs provide several alternative interpretations of a circuit. For example, an inverter gate can have alternative interpretations defined for simulation, netlist creation and timing analysis. Unlike this approach, our implementation uses quotations to capture the circuit structure as an unevaluated expression. Note that, given a different setting, this expression would have been used to simulate the circuit. However, by delaying the evaluation and by having access to the abstract syntax tree of the expression, we are able to traverse this structure and output additional semantically interpretations. The advantage is that the different semantic interpretations operate on the same instance of the quoted expression. However, this needs to be done in two separate stages, first to compose the structure, and then to interpret the structure.

The meta-programming features found in reFLect, provides not only the possibility to manipulate terms representing primitive gates, but also to manipulate terms representing whole circuit definitions. Embedding a HDL using such features can result in an advantage over other HDL embeddings, since the access
and manipulation of whole circuit definitions (the circuit generators), should aid in the reasoning of non-functional aspects of circuits. The hardware description languages mentioned earlier have shown that the deep embedding approach offers more advantages over the shallow embedding approach, yet, these don’t have full control over certain circuit features, especially over the non-functional aspects. For instance, an important non-functional aspect of circuits is the placement of the primitive elements. Pebble [LM98], a small language similar to structural VHDL, defines circuit components in terms of blocks. The end-user can describe how the blocks are positioned, meaning that a block can be defined to be placed above or beside another allowing blocks to be placed either vertically or horizontally to each other. In our implementation we adopted this idea of blocks, by means of the meta-programming features provided by reFLect. However, the challenges are different from those of Pebble, since Pebble is not an embedded language within a function language. In Pebble, language constructs where developed to define blocks and the placement of these blocks, while our implementation uses nested quotation constructs to represent a block in a functional setting, by abstracting away the details of whole circuit definitions.

Wired [ACS05] is another embedded HDL, built upon the concept of connection patterns, in a certain way extending Lava to enable reasoning about connection of circuit blocks. The concepts behind Wired are mostly inspired by Ruby [JS94], more precisely on the adoption of combinators for the placement of circuits. We plan to follow certain features of Wired, for instance to use combinators at the abstract level of blocks.

Our work is based on similar work done in embedding a Lava-like HDL in reFLect [MO06]. As in their case, we base our access to the structure of the circuit descriptions on reflection features of the host language. One difference in our approaches, is that we use structures to represent signals, as opposed to raw boolean values used in reFLect. One of the reasons for this variation is that we try to conceal the use of quotation marks in the circuit descriptions, hence making the reflection features used only in the underlying framework — not forcing the end user to use these constructs. In our approach we emphasize the concept of a marked block in a circuit, which we plan to use for placement and circuit analysis. We still have a number of features unimplemented — such as the lack of component sharing and implicit wrapping and unwrapping of structures of signals — which we plan to develop in the near future.

5 Conclusions and Future Work

In this paper, we have presented a rudimentary HDL embedded in a functional meta-programming language. Our main motivation behind the use of reflection is to enable the creation of tagged blocks by looking at the structure and control-flow of the circuit generator. By adding circuit combinators, similar to the ones used in Ruby [JS94], we plan to use the control given to us into looking at the circuit generators to aid the generation of placement hints as used, for instance, in Pebble [LM98].
Another area we intend to explore is that of optimisation of circuits produced by hardware compilers. The use of embedded HDLs for describing hardware compilers has been explored [CP02]. Despite the concise, compositional descriptions enabled through the use of embedded languages, the main drawback is that the circuits lack any form of optimisation or information. Furthermore, introducing this into the compiler description breaks the compositional description, resulting with a potential source of errors in the compilation process. If one still has access to the recursive structure of the control flow followed by the compiler to produce the final circuit, one can perform post-compilation optimisation, without having to modify the actual compiler code. We plan to investigate this further through the use of the features provided by reFLect.

References


Adaptive Jukebox: A Context-Sensitive Playlist Generator

Matthew Rizzo
Department of Artificial Intelligence, University of Malta

Abstract. Nowadays, a lot of users own large collections of music MP3 files. Manually organising such collections into playlists is a tedious task. On the other hand random playlist generation may not always provide the user with an enjoyable experience. Automatic playlist generation is a relatively new field in computer science that address this issue, developing algorithms that can automatically create playlists to suit the user’s preferences. This paper presents our work in this field, where we suggest that playlist generators should be more context-sensitive. We also present Adaptive Jukebox, a context-sensitive, zero-input playlist generator that recommends and plays songs from the user’s personal MP3 collection. Initial experiments suggest that our system is more accurate than both a random generator and a system that does not take context into account.

1 Introduction

The increase in popularity of music downloads, coupled with the increasing storage capacity of personal computers has given rise to users having large music collections on their computers. Manually organising such collections into playlists is a tedious and time-consuming task. On the other hand, playlists generated using random shuffle fail to satisfy the criteria of a good playlist. A good playlist is one that can adapt music to suit the user’s current mood and the activity he is involved in [OK06], and caters for the user’s desire for repetition and surprise [PRC00].

The solution to this problem lies in automatic playlist generation, a relatively new field in computer science concerned with the development of algorithms that can automatically create playlists to suit the user’s preferences. This paper presents our research [Riz07] in the field of playlist generation. We suggest that playlist generation needs to make a greater use of context and user modelling and we propose the Adaptive Jukebox — a zero-input, context-sensitive adaptive playlist generator that recommends and plays music while the user is using his computer. We are still in the process of evaluating our system, but initial results are encouraging and suggest that for two out of three participants in a user experiment, our system not only performs better than a random generator but also performs better than a system that does not take context into consideration. In the following section we will go over literature in the area of playlist generation as well as the context and user modelling fields. In section 3 we describe the
design of our system and in section 4 we discuss some implementation issues. In section 5 we describe our evaluation approach and present our initial results. Finally, section 6 contains some concluding remarks.

2 Literature Review

2.1 Playlist Generation

Early research in the field of playlist generation mainly focused on the notion of music similarity and on generating a sequence of songs (playlists) that satisfy some user-defined criteria [PRC00] [AT00]. Researchers then started experimenting with signal processing in order to define song similarity which allowed them to generate sequences of songs that sound the same [Log02] [AP02]. Logan [Log02] however suggested that there was more to playlist generation and stated that an ideal system would incorporate context, user modelling and user feedback.

Context-sensitive playlist generators include Affective DJ [HPD98] which recommends songs based on skin conductance, a feature that correlates well with emotion, and PersonalSoundtrack [ET06] which adapts the music to the user’s pace while the user is walking. Other systems such as PATS [PE02] and Recommend ’n Play [AH06] use pre-defined contexts of use or allow the user to manually state what context he is in. The most interesting work in this area is that by Oliver and Kreger-Stickles [OK06] who developed PAPA, a framework that caters for a broader definition of context. However, the only system built on it, MPTrain, only takes into account the user’s heartbeat and current pace.

User modelling in playlist generation was first introduced by Field et al. [FHM01] who suggested storing the user preferences in a long-term and a short-term user model. Pampalk [PPW05] uses a short-term user model that stores the songs that the user accepts or rejects during the current session, eventually using it to recommend songs similar to the accepted ones and different to the rejected ones. In Recommend ’n Play [AH06], songs that are accepted together in a playlist are considered to be similar while rejected songs are assumed to be dissimilar. Thus the system builds a long-term model of what the user considers to be similar songs. PAPA [OK06] has the most sophisticated user model which stores demographic data, user history and mappings between context and user preferences.

Playlist generators that adapt their recommendations to suit users, take into account user feedback and use it to update their user model. Pampalk et al. [PPW05] and Elliott and Tomlinson [ET06] learn that songs skipped by the user should not be recommended in the future. Recommend ’n Play [AH06] lets the user inspect the playlist before it is played, and allows him to add or remove songs from the playlist, thus learning which songs are liked or disliked.

We believe that playlist generation can be improved by giving more importance to context because most existing systems only use one source of context (such as the heart beat or user pace) or a defined context of use. We also think that it is important to develop a sophisticated user model that can learn user preferences.
in various contexts. Methods of collecting user feedback, on the other hand, have already been addressed successfully in literature and therefore our focus will be on context and user modelling which are introduced in the following sections.

2.2 Context

Context is any information about the circumstance, objects or conditions surrounding a user that is considered relevant to the interaction between the user and the computing environment [RC03]. Context includes:

- **Mood**: Research indicates that music and emotional state are strongly related [HA02] [LO03]. Emotion can be detected in various ways including facial expression recognition [PR00], audio-based emotion recognition [IC05], recognition through physiological inputs [PVH01] or through mouse and keyboard [ZGDG03].

- **Activities**: Musical preferences also depend on the activities that a user is involved in [OK06]. For a computer user, the activity is strongly related to the programs that are open and the windows that have focus.

- **Social Context**: The style of music may also depend on who the user is with [AH06].

- **Physical Context**: Physical context includes location and time both of which can have an effect on user preferences. In fact, Oliver and Kreger-Stickles [OK06] suggest that preferences might depend on whether the user is at work or in the car and Andric and Haus [AH06] state that preferences might depend on the time of day.

- **Environmental Context**: Environmental context such as weather as well as light and sound levels [RC03] can have various effects on musical preferences, and probably vary from one individual to another. Some people’s moods might be effected by the weather, which in turn effect musical preferences. Light and sound levels might give other indications as to what the user is doing, what time of day it is and the general atmosphere that the user is in or is trying to create.

2.3 User Modelling

Jameson [Jam01] states that when designing a user model we need to understand its functions, the properties it will store, the inputs and the techniques used to construct it.

For playlist generators the function of the user model would be to learn user preferences to be able to recommend songs to the user. In literature, user properties modelled include beliefs, goals, knowledge, level of expertise and user interests. In our case we need to store the user’s musical preferences, as in Music IR systems where user models store songs that the user considers to be good or bad [HMI03] and the definition of similarity [Rol01]. User model inputs can be self reports, response to test items and naturally occurring actions. The latter is the most common approach in playlist generation (e.g. user skipping a song). User
modelling techniques include stereotypes, logic-based approaches as well as machine learning and Bayesian methods. The latter is probably the most applicable to playlist generation.

3 Design

Adaptive Jukebox is our zero-input, context-sensitive playlist generator that plays music from the user’s MP3 collection while the user is using his computer. It operates in two modes: indexing mode and training and playback mode. The indexing process loads songs from the song repository (e.g. the user’s hard disk) and extracts features that are stored in a song database and later used for recommendation. During the training phase the system allows the user to select songs and build playlists manually and learns the user preferences while during playback the system plays songs to suit the user’s current context. Figure 1 shows the design of our system. Our main hypothesis is that through our approach we can play music that the user likes and that the system performs better than a random generator, or a generator that does not take context into account. We also aim to develop a user-friendly application that does not require any input (other than that during the training phase) and a system whose performance improves over time.

Fig. 1: Overview of Adaptive Jukebox
3.1 Music Feature Extraction

This module is responsible for indexing the songs in the user’s MP3 collection and storing the data in the song database. The module first converts the MP3 file into a WAV file using JLayer\(^1\). The system then uses the libofa library to download the song title and artist’s name from MusicDNS\(^2\). The WAV file and the song title and artist’s name are then fed through a number of components that extract audio information and meta data respectively. Audio information includes loudness and fluctuation patterns (both extracted using CoMIRVA [Sch06]) and the beats-per-minute (which uses BeatRoot [Dix06] to detect the note onsets). Meta data includes the year the song was released (downloaded from www.musicbrainz.org), the lyrics, as well as genre and artist similarity. From the lyrics downloaded from www.lyricwiki.org, we first extract the song’s language. We store lists of English, French, Spanish, German and Italian stopwords and count the number of occurrences of these words in the lyrics. The song’s language is defined as the language with the highest count. For English songs, we check whether the song has explicit lyrics by passing the lyrics through a filter that detects bad words. Genre is detected by querying the Yahoo web service\(^3\) for artist + music + genre. The summaries of the first ten documents are extracted. The number of occurrences of ten predefined popular genres is then calculated. The genre with the highest occurrence is taken to be the song’s genre. We also use the Amazon web service\(^4\) to detect the song’s genre and also download artist similarity. Two artists are considered similar if their work appears in each other’s recommended products on Amazon.

We therefore extract eight features (loudness, beat, fluctuation patterns, year, language, explicit lyrics, genre and artist similarity). The music feature extraction module then stores these features in the database. It also creates and stores a song similarity matrix that defines the similarity between all songs, along the eight features. So for every pair of songs, we store eight values that define the level of similarity between the songs.

3.2 Context Server

The context server is responsible for reading data from the environment and then detecting the user context. We read data from nine sources (applications that are open, mouse activity, keyboard activity, facial expressions, time, day of the week, brightness level, sky conditions and temperature).

For applications we record all windows that gain focus and during training we cluster this data using an agglomerative hierarchical clustering algorithm so that we can define the windows that make up the various activities that the user is involved in. As suggested by Oliver et al. [OSTS06], the similarity function used for clustering, is based on the times the windows were accessed and the titles

\(^1\) www.javazoom.net
\(^2\) www.musicip.com
\(^3\) www.yahoo.com
\(^4\) www.amazon.com
of the windows. During playback, the windows that have focus are compared to the application clusters in order to determine the current application context. For mouse we record data such as acceleration, deceleration, click number, overshoot number and overshoot length [Mae05], average speed, efficiency and uniformity [Mae05], movements per minute, distance travelled per mouse movement and per minute and active time per minute. Data recorded for keyboard activity includes keystrokes per minute, average keystroke duration, and the number of different keys pressed. During training, both mouse and keyboard data are clustered using K-means into five categories, each of which defines a different pattern of mouse or keyboard activity. During playback, the system uses these clusters to determine the current mouse and keyboard activity.

Facial expressions are detected by first extracting 22 facial features from webcam snapshots using FSearch [CC06]. We then normalize the points read and calculate distances between points (such as between the eyebrow and the eye). During training we calculate the averages of these readings and then compare each reading to these averages to detect ten action units using rules that are similar to those used by Pantic and Rothkrantz [PR00]. For example, we detect that the distance between the left eyebrow and eye is greater than average and therefore signifies a particular action unit. During training we then cluster the action units into three categories using K-means. During playback the detected action units are compared to the clusters to determine the user’s facial expressions.

The algorithms to detect the other types of context are simpler. The time and day of week are read from the system clock, the brightness level is the average RGB value of a pixel from a webcam snapshot and the sky conditions and temperature are downloaded from a web service.

3.3 User Model

Although we experimented with probabilistic user models, the best approach was to use context clustering. During training, the nine context values detected by the context server are clustered using a hierarchical clustering algorithm to determine the various scenarios the user is in. The songs played in each context cluster are then analysed and used to build the user model. We store two types of user properties for each context: the weights in the similarity function (that define how the similarity between two songs is computed given the similarity in terms of the eight features) and the list of songs that the user likes to listen to in that context. The weights in the similarity function are initially learnt through a genetic algorithm whose fitness functions outputs a high value if two songs played consecutively are considered similar by the function. The weights are then updated using an algorithm that is similar to the one proposed by Rolland [Rol01]. The preferred songs is a list of song-value pairs where the song at the top of the list (highest value) is the one that is played often in that context.

We also store the user history and a set of filters that determine whether songs by the same artist should be repeated in a session or whether the system should repeat songs. These two pieces of data are not dependent on context.
3.4 Recommendation Engine

Given the user preferences in the current context, the recommendation engine outputs the song to be played next. The system first creates an ordered list of suitable songs depending on the user history, the preferred songs in the current context and the songs that were accepted and rejected during the current session (Songs that are similar to accepted songs are given a higher position in the list than songs similar to rejected songs). The recommendation engine then outputs the first song on the list that does not violate one of the filters in the user model.

3.5 MP3 Player

The MP3 player plays the recommended song and reports any user actions to the feedback engine. The interface used to play songs is jGui\(^5\) because it is user-friendly and sophisticated.

3.6 Feedback Engine

When the user accepts or skips a song during playback, the feedback engine updates the user model. If a song is accepted, it is added to the list of preferred songs for the current context and the weights in the similarity function are updated to reflect that the current song and the last song played are similar. On the other hand if the song is rejected, the song’s value in the preferred song’s list is decreased and the weights in the similarity function updated so that they reflect the fact that the current song and the last song played are not similar.

4 Implementation

The system is implemented as a Java application so that it is highly portable. The only components that are dependent on the operating system or the hardware are the context server and the interface. The context server is divided into two components such that the component that reads data from the environment is not part of the Java application. It is in fact implemented as a .NET application and runs on Windows. The .NET application communicates with the main Java program by sending XML messages over TCP/IP. Having said that, the system can easily be ported to another operating system or hardware setting. For example, by developing an interface that plays music on a car MP3 player and by creating a system that detects a driver's context, the system can be used in a car environment.

\(^5\) www.javazoom.net
5 Evaluation and Results

We are currently evaluating the system on three participants who volunteered to use the system for a few weeks. The evaluated system is as described above, although we do not take into consideration facial expressions and brightness due to the unavailability of webcams. During the first weeks the users were asked to use the system to listen to music of their choice. We recorded the context the users were in and the songs played in that context. This data was then used for offline evaluation. It was divided into a training set and an evaluation set and the system was trained on the first set. The context readings from the evaluation set were then passed onto the system which output the recommended song for that context. This was compared to the actual song heard and if the two songs are similar\(^6\) we marked the recommendation as correct. In the event of an incorrect recommendation, the system made one more attempt to suggest another song (simulating the user skipping the song). The system’s accuracy is then calculated as the percentage of correct recommendations.

For the three participants, we ran the above experiment ten times and calculated the average accuracy. We compared the performance of three systems — a random generator, Adaptive Jukebox, and our system without a context server. The results are shown in table 1. As can be seen our system performs better than random for all three participants and for two participants we achieve a slightly better accuracy result when taking context into account.

<table>
<thead>
<tr>
<th></th>
<th>Participant A</th>
<th>Participant B</th>
<th>Participant C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive Jukebox</td>
<td>43.45%</td>
<td>21.03%</td>
<td>39.08%</td>
</tr>
<tr>
<td>Without Context</td>
<td>39.84%</td>
<td>22.22%</td>
<td>37.30%</td>
</tr>
<tr>
<td>Random</td>
<td>19.76%</td>
<td>17.66%</td>
<td>32.13%</td>
</tr>
</tbody>
</table>

Table 1: Evaluation Results

We also plan to run more experiments on the data collected in order to determine whether all context types are important or whether we can remove some sources of context. In the meantime the participants are also using the system in playback mode, where the system recommends songs itself. Evaluation results will be presented in our thesis [Riz07].

6 Conclusion

The low number of participants in our evaluation experiment prevents us from stating any strong claims about our system. However our experiment indicates that incorporating context-sensitivity and using a user model to store musical

\(^{6}\) In order to check if two songs are similar, we use the user defined similarity between the songs' artists. The user defined similarity is high if the training and evaluation sets contain songs by these artists that are close to each other in the playlists.
preferences in various contexts could help playlist generators make better recommendations. Future work should include better evaluation experiments (e.g. effect of facial expressions) with more participants. Other enhancements might include the incorporation of more features in the song database (such as timbre and instrument detection) and more sources of context (such as physiological inputs for emotion recognition).

References


Jameson, A. Systems That Adapt to Their Users: Description of an IJCAI 01 tutorial. Tutorial given at the International Joint Conference on Artificial Intelligence (2001)


Abstract. Statistical Machine Translation (SMT) developed in the late 1980s, based initially upon a word-to-word translation process. However, such processes have difficulties when good quality translation is not strictly word-to-word. Easy cases can be handled by allowing insertion and deletion of single words, but for more general word reordering phenomena, a more general translation process is required. There is currently much interest in phrase-to-phrase models, which can overcome this problem, but require that candidate phrases, together with their translations, be identified in the training corpora. Since phrase delimiters are not explicit, this gives rise to a new problem; that of phrase pair extraction. The current project proposes a phrase extraction algorithm which uses a window of $n$ words around source and target words to extract equivalent phrases. The extracted phrases together with their probabilities, are used as input to an existing Machine Translation system for the purpose of evaluating the phrase extraction algorithm

1 Introduction

Machine Translation (MT) was one of the first applications envisioned for computers. As early as 1949, Warren Weaver described translation as a decoding problem. In 1954 IBM demonstrated the concept in a basic word-for-word translation system.

Interest in MT spans from the academic to the commercial domain. Over the years it has gained popularity on the web, resulting in its being one of Google’s most used features. The European Union spends more than 15% of its annual budget on translation, so even partial automation of the translation process could lead to huge savings. MT uses a number of other Natural Language Processing (NLP) technologies including parsing, generation and word sense disambiguation amongst others.

Amongst the problems with MT are word order, word sense, pronoun references, tense and idioms. A number of different approaches to MT have developed over the years. The choice of approach depends on the domain for which MT is being considered. Is the system going to be used for a single or multiple languages? Is there a constrained vocabulary or will it be required to translate any text? Are there existing resources which can be used? How much time do we have to develop the system? What quality of translation is sufficient for the application at hand?
Word-for-word translation systems make use of a bilingual dictionary to translate every word in the text. This is a simplistic approach which is easy to implement and provides a rough idea of the nature of the source text. However, there are problems with word order which result in low quality translations. Syntactic transfer systems such as ELU (e.g. Estival et. al [EBRS90]) have been used to solve the word order problem, since the source sentence is parsed, its constituents rearranged and then translated. However, such systems require transfer rules for each of the languages under consideration. Other approaches exist, including Example-based Machine Translation (EBMT) which uses the concept of analogy to perform translation. The source sentence is decomposed into a sequence of fragments. Each fragment is then translated individually and then composed properly to form the target sentence. Statistical Machine Translation (SMT) (Brown-et.al. [BCDP88]) with which this paper is concerned, uses probability to find the most probable target translation given a source text.

The remainder of this paper is structured as follows. Section 2 and 3 give an overview of SMT and Phrase Alignment respectively. Section 4 proposes an algorithm for phrase identification, extraction and alignment. Section 5 discusses how the proposed method will be integrated in an existing MT system to test and evaluate the output of the phrase alignment module. We conclude in section 6 with an overview of some outstanding issues and suggestions for future work.

2 Statistical Machine Translation

Statistical Machine Translation (SMT) finds the most probable translation of a sentence on the basis of a model which is inferred from training data consisting of large quantities of translated text. It has a number of advantages over transfer-based and example-based MT. It is data-driven, language-independent and does not require any linguist or language experts. In addition, a new system can be prototyped quickly and at low cost provided that parallel training corpora are available for the language pair in question.

SMT rests upon a remarkably simple insight: given an occurrence of a sentence $m$ in, say, Maltese, and any other sentence $e$ in, say, English, there is a non-zero probability that given $m$, $e$ expresses what the speaker of $m$ had in mind when $m$ was said. We write this probability $P(e|m)$, which intuitively represents the probability that $e$ is a translation of $m$.

Now suppose that for the same $m$, I am offered two candidates: $e_1$ and $e_2$. How will I choose which is the better translation? Obviously I will work out $P(e_1|m)$ and $P(e_2|m)$ and then choose $e_1$ if $P(e_1|m) > P(e_2|m)$, else I will choose $e_2$. This gives us the basis for a theory of translation. In order to find the translation of $m$, find $\hat{e} = \text{argmax} P(e|m)$, i.e. the $e$ which maximises $P(e|m)$. Bayes’ theorem allows us to decompose the latter probability into $P(e)P(m|e)/P(m)$, so that $\hat{e} = \text{argmax} P(e)P(m|e)/P(m)$, which simplifies to

$$\hat{e} = \text{argmax} P(e)P(m|e)$$

because the denominator is independent of $e$. 
In a classic paper Brown et al. [BCDPDP93] refer to this equation as the “Fundamental Equation of Machine Translation”, since it summarises three computational issues that need to be addressed in the design of SMT systems. These are:

- Estimating the language model probability $P(e)$;
- Estimating the translation model probability $P(m|e)$;
- Designing a search method to identify the English string which maximises the product.

The last issue, which is usually referred to as “decoding”, brings the translation process firmly back to the familiar territory of search optimisation, which we will not discuss the process further here, except to say that (1) it normally source-sentence driven, proceeding by translating successive segments of the source sentence, and (2) at any given stage, extension of a partial translation to the next segment involves a probabilistic calculation that eliminates the least probable of the possible extensions. Such calculations make use of the (already existing) translation model.

In an ideal world, a translation model $P(m|e)$ would take the form of a gigantic lookup table that associated a probability, i.e. a real number between 0 and 1, to every possible pairing of a Maltese string and an English one. The question is therefore, how we go about estimating those probabilities. Clearly, there is neither enough data nor computing power in the world to estimate the probabilities by counting the frequencies of every string pair individually. The general solution is therefore (1) to divide the translation model of the whole sentence into smaller parts for which translation probabilities are more readily available and (2) to combine the translation probabilities of the parts. The simplest way of doing this, and the basis of word-based SMT systems, is to consider the translation probabilities between individual words in $m$ and $e$. In such cases, for a sentence of length $k$:

$$P(m|e) = P(m_1|e_1) \times \ldots \times P(m_i|e_i) \ldots \times P(m_k|e_k)$$

where $m_i$, $e_i$ represent the $i$th word of $m$ and $e$ and $1 = i = k$.

Clearly, such a model will work well when there is a 1:1 correspondence between the words of source and target sentences. However, this is a simplistic assumption, and when it fails, this is the system may assign a high probability to a low quality sentence.

By considering the empty string as a word, systems based on word translation models can handle insertions and deletions. However, they cannot capture local word reordering. Even simple cases like noun/adjective ordering differences between source and target cannot be dealt with, and the net result is low translation quality.

## 3 Phrase Alignment

As SMT evolved, concerns about translation quality as well as better availability of data have led to development of SMT based on other kinds of statistical model.
Och and Ney [ON04]) present an “alignment template” approach to translation which allows for many-to-many relations between source and target words, whilst Huang et al. [HKJ06] adopt a syntax-directed approach based on the relationship between the nodes in a syntax-tree on the source side and a derived target-language string. This paper is concerned with the problem of extracting phrase-translation models from bilingual data. Such models, once available, can then be used by a statistical translation system to actually carry out translations. The basic problem is that, given a source and target sentence, the number of possible phrase alignments is much too large for the probability of each to be individually calculated. Therefore, we need some principled way to identify the most promising candidates. The essence of the principle is the following hypothesis: words that are translation equivalents are more likely to be embedded in phrases that are translation equivalents than in phrases which are not. If this hypothesis is true, we can use words that are known a priori to be translation equivalents to generate related phrases in on both source and target sides, and hence, to generate phrase alignments.

4 Proposed Methodology

We already have in place a system for the paragraph, sentence and word alignment of Maltese and English bilingual corpora (Bajada [Baj04]). The method being proposed uses previously aligned sentences and words as the starting point for phrase identification and alignment.

The method has two key elements; phrase identification and phrase alignment. Essentially, phrase identification takes a word and delivers a set of phrases that involve that word; phrase alignment takes a phrase within a sentence and delivers the most probable translation equivalent of that phrase. The next two sections describe these two processes in more detail.

4.1 Phrase Identification

The proposed algorithm assumes the existence of a word translation model in the sense defined above, and starts off considering a source word and its target translation. At this stage each word in the source vocabulary is considered individually. Suppose we are considering a particular source word, w.

Definition 1. An n-m-phrase is considered to be a contiguous sequence of words $l_1, \ldots, l_n, w, r_1, \ldots, r_m$ m, n = 0 focused on the word w.

The function get-phrase($w, s, n, m$) extracts an n – m-phrase containing w from sentence s so that by setting particular values for n and m we can, for example, focus on left or right context, or a combination of both. If s has insufficient words, the result is padded out with null strings working outward from w, thus maintaining the relative position of w within the phrase whose total length is $m + n + 1$. 

An $n - m$-phrase $p$ of length $k$ focused on word $w$ and given $n$ and $m$ is deemed to be interesting for sentence $s$ if

- $w \in s$
- $p = \text{get\_phrase}(w, s, n, m)$ for $n + m + 1 = k$
- The probability of $p$ exceeds a certain threshold which is defined with respect to all other $n - m$-phrases in the source corpus.

For a given corpus, the best values of $n$ and $m$ are established empirically, and ultimately judged by the quality of the phrase pairs extracted as described in section 5. Our initial hypothesis is that $n = k - 1$ and $m = 0$ i.e the given word $w$ is at the extreme right of the phrase whose total length is $k$.

There are many different ways in which the probability of $p$ might be established with reference to a training corpus e.g. with respect to an $n$–gram language model of the source corpus or possibly a subcorpus of sentences containing at least one occurrence of $w$. We are currently experimenting with some simple models.

### 4.2 Phrase Alignment

The process of phrase identification presented in the previous section is central to the phrase alignment process. However, a number of steps are required to put the whole process of phrase extraction and alignment together.

As mentioned previously we have available a set of word alignments and their respective translation probabilities. We need to identify those phrases which are of interest to us and which, if extracted and aligned, will be of benefit to the translation process. We assume that those source words having a combination of a high word count in the training text and a high translation probability are those words of interest to the alignment process. The word translation equivalents are sorted and filtered according to these criteria. The following steps are then carried out on the sorted list to complete the alignment process.

For each word pair $(m, e)$ in the list

- $A = \{ (ss, st) \mid m \in ss \& e \in st \}$
- $B = \{ (ps, pt, P) \mid (ss, st) \in A \& ps = \text{phrase\_identify}(ss, m, k, 0) \& pt = \text{phrase\_identify}(st, e, k, 0) \& P = P(ss|st) \}$
- Return $B$

The function $\text{phrase\_identify}$ simply invokes the phrase identification process outlined in the previous section, yielding a phrase whose probability is above threshold.

The final step in the alignment process is the calculation of the translation probabilities for the resulting phrase pairs. There are various ways of doing this, the simplest being to take the product of the translation probabilities of the words which are part of the phrases. This works because the words of the two phrases starting with the $m$ and $e$ are mostly aligned according to our principle.
Alternatively bigram probabilities might be used: as with any statistical model, other factors can be introduced, such as the relative frequency of the collected phrases, as discussed in Koehn’s lectures.

Outputting a phrase table which contains the translation probabilities of the extracted phrases is an important step which allows the results to be used as the Translation Model component of a SMT system.

The above steps extract a phrase table which is suitable for evaluation. The system needs to be as flexible as possible to cater for different language pairs. The phrase extraction algorithm has to be tuned to determine the optimal extraction method — words before, words after or a combination of words before and after the initial word.

In addition a cutoff point for phrase extraction needs to be determined; that is, when to stop processing the phrase. This may be done by requiring the phrase to have a probability greater than a threshold value of by setting the maximum length of the phrase.

The aim of the current project is to develop a highly configurable system, the parameters of which will be used to determine the extraction of phrases from a training corpus.

Developing the system this way will allow us to experiment with and modify these parameters until an optimum set is found. This will be determined by applying the extracted phrase table to and SMT system and using evaluation metrics to measure the quality of the output translations.

5 Discussion

5.1 How the System is Used

The system described above yields a series of phrase translation models, i.e. tables whose entries are triples of the form \{source phrase, target phrase, probability\}. By varying the parameters of get-phrase, we obtain different translation models, and our aim is to investigate the settings which produce the best translations.

The proof of the pudding is in the eating — and in our case the eating corresponds to how well the model actually translates a set of test sentences. To put this into practice we require a translation system into which we can plug the translation table, and a way to assess the quality of individual translations. Fortunately, both of these are available off the shelf.

Moses (Koehn et. al [KHB+07]) is a state-of-the-art factored phrase-based beam-search decoder for SMT which is freely available. It is a traditional SMT system in that it relies on a language model and a translation model to perform translation. In addition, Moses offers a number of advanced features and additional tools for machine translation, such as scoring and analysis tools.

5.2 Translation Model

The Translation Model used by Moses is a phrase translation table, with entries such as those shown in Figure 1.
The entry indicates that the probability of translating the German phrase *das ist* into the English phrase *this is* is 0.3. The integration and evaluation of our results requires that they be written to a phrase translation table formatted as shown in Figure 1.

### 5.3 Evaluation

The Moses toolkit also provides a number of support tools enabling further analysis of output translations. Among these is a simple BLEU scoring tool. BLEU (Papineni et al [PSWZ88]) is an automatic evaluation metric used for MT proposed by the IBM MT research group. The central idea of automatic evaluation is to use a weighted average of variable length phrase matches against the reference translations. BLEU averages the precision for unigram, bigram and up to 4-grams and applies a length penalty if the generated sentence is shorter than the best matching (in length) reference translation.

Application of the BLEU metric to the translations output by Moses using our phrase translation table will indicate the quality of the translations. This will in turn provide indication of the validity of the aligned phrases.

### 6 Conclusion

There are a number of unknown variables which factor in the identification and extraction of phrases. For two different languages there is no guarantee that equivalent phrases will have the same attributes, such as length and word order. Developing an algorithm to automate such a process requires that the algorithm be flexible enough to cater for such variation. The automatic extraction of word-based translation models from corpora is performed by assuming that every pair of source/target words chosen from two aligned sentences have an equal probability of being equivalent. More refined models allow for the positions of words within the aligned sentences contribute to the probability of their alignment. Such principles can be extrapolated to phrase alignment; that is, we assume that only phrases from equivalent sentences are a correct match. In addition, having aligned the texts at word level, we have a dictionary of words and their possible alignments which may be extended to phrases.

The default assumption is that phrase extraction from Maltese and English texts may be carried out in the same way because of the similar structure between the two languages. A different language pair might require a different methodology. We are also able to adopt this approach due to the level of confidence in the sentence and word translation equivalents obtained by the current system. The
developed system needs to be as flexible as possible to cater for different language pairs. The aim of the current project is to develop a highly configurable system which depends on a number of parameters to determine the identification and alignment of phrases from a training corpus. Developing such a system will allow us to experiment with and modify these parameters until an optimum set is found.

Manual evaluation of results obtained using an initial implementation of the proposed algorithm on a set of bilingual texts gave positive results. The results indicated that parameterisation of the algorithm would improve the results. Having extracted a set of equivalent phrases offers a number of processing options. It would be interesting to attempt template extraction, whereby the resulting phrases are analysed for similarities and used to build templates to offer more generic translation options.

References


Automatic Interface Generation for Compositional Verification

Sandro Spina, Gordon J. Pace and Frédéric Lang

1 Department of Computer Science, University of Malta
2 INRIA Rhône-Alpes, Grenoble, France

Abstract. Compositional verification, the incremental generation and composition of the state graphs of individual processes to produce the global state graph, tries to address the state explosion problem for systems of communicating processes. The main problem with this approach is that intermediate state graphs are sometimes larger than the overall global system. To overcome this problem, interfaces [JL97], and refined interfaces [Lan06], which take into account a system’s environment have been developed. The number of states of these interfaces plays a vital role in their applicability in terms of computational complexity, which is proportional to the number of states in the interface. The direct use of complete subcomponents of the global system as interfaces, thus usually fails, and it is up to the system designer to describe smaller interfaces to be used in the reduction. To avoid having to verify the correctness of such manually generated interfaces, we propose automatic techniques to generate correct interfaces. The challenge is to produce interfaces small in size, yet effective for reduction. In this paper, we present techniques to structurally produce language over-approximations of labelled transition systems which can be used as correct interfaces, and combine them with refined interfaces. The techniques are applied to a number of case-studies, analysing the trade-off between interface size and effectiveness.

1 Introduction

Over the past years, the verification of computational systems has taken up considerable interest and support. Techniques in both symbolic and enumerative strategies made many advances in terms of what can be verified. However, the main problem still remains that of the state space explosion arising when composing together components of a system. Many solutions have been proposed and used to counter the problem. Our work focuses on enumerative compositional verification in which the state graph of a system made up of a number of processes, is generated incrementally. In compositional verification, instead of generating the global system, the state graphs of the constituent processes are individually generated, reduced up to some equivalence relation which preserves global system behaviour, and incrementally composed together until the global system is obtained. The problem with this approach is that intermediate state graphs are sometimes larger than the global system, simply because the behaviour of the processes is not constrained by the other processes which have not
yet been composed. The challenge is thus one of constraining the state graph of these intermediate processes without losing anything from their behaviour within the global system.

Interfaces [GSL96, GS91, CK93], provide a means of overcoming this problem. An interface, essentially represents part of the environment of a particular subcomponent of the global system. This environment essentially poses behavioural restrictions imposed on the subcomponent through synchronisation. For instance, in system $S$, composed of processes $P_1$ to $P_n$, processes $P_2$ to $P_n$ (or abstractions of them) can be seen as interfaces of $P_1$. Krimm and Mounier [JL97] use interfaces to constrain on-the-fly the state graph of the process being generated and introduce the semi-composition operator which is implemented as the *Projector* tool in the CADP toolkit [GLMS07] which we are using$^3$. In their approach, the interface represents the only possible interactions between the process and its environment. Semi-composition is used to reduce the process states and transitions which will anyway not be reachable in the global system given the knowledge in the interface.

Refined interfaces [Lan06] extend these interface techniques to take into account whole families of concurrent processes as an environment. Lang [Lan06] proposes a technique to automatically generate interface processes from a subset of processes that are composed with the process of which we want to produce the state graph. In a number of case studies it has been shown that refined interfaces allow a better reduction of the process state graph we want to generate. The higher the number of processes used to produce a refined interface, the better this interface would describe the environment of a process. However, the higher the number of processes used the higher the number of states the interface will end up with. In both techniques, the computational complexity of the reduction for a given interface is proportional to the number of states in the interface. The direct use of complete processes of the global system as interfaces, may thus fail, and it is up to the system designer to describe smaller interfaces to be used for the reduction. However, when using an abstraction of the communicating processes, it is important to guarantee that the abstraction is, in fact, correct. Techniques to verify the correctness of an interface have been proposed [JL97], but come at a price. Furthermore, the design of these interfaces requires expert knowledge of the system being verified in order to be effective. In this paper, we propose techniques for the automatic construction of interfaces which are guaranteed to be correct by construction. The techniques are applied to a number of case studies, analysing the trade-off between interface size and effectiveness.

2 Background

2.1 Preliminary Definitions

The behaviour of a process can be modelled as a labeled transition system (LTS), consisting of a set of states and a labeled transition relation between states.

---

$^3$ Available from [www.inrialpes.fr/vasy/cadp](http://www.inrialpes.fr/vasy/cadp)
Each transition describes the execution of the process from a current state with a particular instruction (label).

In what follows \( A \) is the global set of labels and \( \tau \) a particular label representing a hidden or unobservable instruction \( (\tau \notin A) \). Given a set of labels \( A \) \( (A \subseteq A) \) we will write \( A_\tau \) to denote \( A \cup \{\tau\} \). For a set of labels \( A \), \( A^* \) will represent the set of finite sequences on \( A \).

**Definition 1 (LTS).** A Labeled Transition System (LTS, for short) is a quadruple \( M = (Q, A, T, q_0) \) where \( Q \) is a finite set of states, \( A \subseteq Q \) a finite set of actions, \( T \subseteq Q \times A \times Q \) is a transition relation between states in \( Q \) and \( q_0 \in Q \) is the initial state.

We will use the notation \( q \stackrel{a}{\rightarrow} q' \) to mean \( (q, a, q') \in T \), and \( q \stackrel{a_1a_2\ldots a_n}{\rightarrow} q' \) to mean that there exist states \( q_1, q_2 \ldots q_n \) with \( q_1 = q \) and \( q_n = q' \) such that for all \( i, q_i \stackrel{a_i}{\rightarrow} q_i+1 \). We abuse the notation to allow \( a^* \) to appear in the string, indicating any number of repetitions of \( a \). We will also leave out \( T \) when it is clear from the context. Given a state \( q \), we will use the notation \( \text{incoming}(q) \) to denote the set of immediate actions which can be performed just before \( q \) \( (\text{incoming}(q) = \{a \mid \exists q' \cdot q \stackrel{a}{\rightarrow} q'\}) \), and similarly \( \text{outgoing}(q) \) to indicate the set of immediate outgoing actions from \( q \). We now define the language generated by a LTS from a particular state \( p \in Q \).

**Definition 2 (Language generated by an LTS).** Given an LTS \( M \), such that \( M = (Q, A, T, q_0) \) and \( q \in Q \), the (observable) language starting from \( q \) in \( M \), written \( L(M) \) is defined as follows \{ \( w \mid \exists q' \cdot w = a_1a_2\ldots a_n \tau \tau^* \tau^* \tau^* \ldots \tau^* q' \} \).

Two LTSS can be composed together with the parallel composition operator synchronising on a set of labels common to both LTSS.

**Definition 3 (Parallel Composition, Hiding).** Let \( S_i = (Q_i, A_i, T_i, q_{0i}) \) \( (i = 1, 2) \) be two LTSS, and \( G \subseteq A \). The parallel composition of \( S_1 \) and \( S_2 \) over \( G \), written \( S_1 \parallel_G S_2 \), models the concurrent execution of \( S_1 \) and \( S_2 \) with forced synchronisation on \( G \), and is defined as the LTS \( (Q, A_1 \cup A_2, T, (q_{01}, q_{02})) \), where \( Q \) and \( T \) are the smallest sets satisfying both \( (q_{01}, q_{02}) \in Q \) and the following rules:

\[
\begin{align*}
(q_1, q_2) &\in Q, \ q_1 \stackrel{a}{\rightarrow}_{T_1} q'_1, \ q_2 \stackrel{a}{\rightarrow}_{T_2} q'_2, \ a \in A \\
(q'_1, q'_2) &\in Q, \ (q_1, q_2) \stackrel{a}{\rightarrow}_{T} (q'_1, q'_2)
\end{align*}
\]

\[
\begin{align*}
(q_1, q_2) &\in Q, \ q_1 \stackrel{a}{\rightarrow}_{T_1} q'_1, \ a \notin A \\
(q'_1, q'_2) &\in Q, \ (q_1, q_2) \stackrel{a}{\rightarrow}_{T} (q'_1, q'_2)
\end{align*}
\]

\[
\begin{align*}
(q_1, q_2) &\in Q, \ q_2 \stackrel{a}{\rightarrow}_{T_2} q'_2, \ a \notin A \\
(q'_1, q'_2) &\in Q, \ (q_1, q_2) \stackrel{a}{\rightarrow}_{T} (q'_1, q'_2)
\end{align*}
\]

Note that, by construction, the states belonging to \( Q \) are reachable. A state \( q_1 \) of \( S_1 \) (respectively \( q_2 \) of \( S_2 \)) is said reachable in \( S_1 \parallel_G S_2 \) if there is a state \( (q_1, q_2) \) in \( S_1 \parallel_G S_2 \). Similarly, a transition \( q_1 \stackrel{a}{\rightarrow} q'_1 \) of \( S_1 \) (respectively \( q_2 \stackrel{a}{\rightarrow} q'_2 \) of \( S_2 \)) is said reachable in \( S_1 \parallel_G S_2 \) if there is a transition \( (q_1, q_2) \stackrel{a}{\rightarrow} (q'_1, q'_2) \) in \( S_1 \parallel_G S_2 \). The system hide \( A \) in \( S_1 \), corresponds to the LTS \((Q_1, A_1 \setminus A, T'_1, q_{01})\), where \( T'_1 \) is defined as follows:
Consider for instance modelling a communication protocol with an LTS \( T \) representing a transmitter, and LTS \( R \) representing a receiver’s behaviour. \( T \parallel_G R \) is the LTS generated by concurrently executing LTSs \( T \) and \( R \) synchronising on the labels in \( G \). Any transitions not in \( G \) are performed independently by the two LTSs.

### 2.2 Compositional Verification Using Interfaces

The semi-composition operator takes an LTS \( M_1 \), and reduces its behaviour up to another LTS \( M_2 \) communicating on an alphabet \( G \). The effect is that of removing sequences of actions which \( M_2 \) would never allow \( M_1 \) to engage in. Formally, the definition of semi-composition is the following [Lan06]:

---

**Definition 4 (Semi-Composition).** Let \( M_i = (Q_i, A_i, T_i, q_{0i}) \) \( (i = 1, 2) \) be two LTSs, \( G \subseteq A \), and \( (Q', A', T', q'_0) = M_1 \parallel_G M_2 \). The semi-composition of \( M_1 \) and \( M_2 \), written \( M_1 \parallel_G M_2 \), is the LTS \( (Q, A, T, q_0) \), where

\[
\begin{align*}
q \xrightarrow{a, T} q' & \quad q \xrightarrow{a, T'} q' \\
q \xrightarrow{\tau, T} q' & \quad q \xrightarrow{\tau, T'} q'
\end{align*}
\]

\( G \) is called the **synchronization set** and the pair \((G, M_2)\) is called the **interface**. We say that an action \( a \in A \) is **controlled** by the interface \((G, M_2)\) if \( a \in G \).

From this definition it is clear that the resultant LTS is a sub-LTS of \( S_1 \). This guarantees that semi-composition never increases the number of states of its first operand. We also know from [JL97] that one can replace the expression \( M_1 \parallel_G M_2 \) with \( (M_1 \parallel_G M_2) \parallel_G M_2 \) without losing any temporal properties of the system \( M_1 \parallel_G M_2 \). Semi-composition also guarantees that \( M_1 \parallel_G M_2 \) is branching bisimilar to \( M_1 \parallel_G M_2 \) if \( L(hide (A \setminus G) in M_2) = L(hide (A \setminus G) in M_2') \). This means that one can first hide uncontrolled actions and then minimize the LTS modulo a relation preserving observable traces (for instance branching or safety equivalence reductions). Minimization of the interface is very important since this reduces the cost of semi-composition, the complexity of which is the same as parallel composition.

Furthermore, these results can be extended to work with language inclusions:

**Conjecture 1** Given LTSs \( M_1 \) and \( M_1' \), such that \( L(M_1) \subseteq L(M_1') \), then for any LTS \( M_2 \), and action list \( G \), \( (M_2 \parallel_G M_1') \parallel_G M_1 \) is branching bisimilar to \( M_2 \parallel_G M_1 \).

The proof of this conjecture is not given for the time being, but will be included in future work.
2.3 Refined Interfaces

In our work we make use of refined interface generation to produce interfaces which take into account a subset of the processes neighbouring the process whose state graph we want to generate. Generation of refined interfaces is based on the network of Ltss concurrent model in which the composition hierarchy is completely flattened. The network of Ltss model is more general than the parallel composition operator defined in the previous section, and the parallel composition, renaming, hiding and cutting operators from many process algebras can be translated into networks of Ltss [Lan05]. We first define vectors, from which networks of Ltss are composed.

Definition 5 (Vectors). A vector of length \( n \) over a set \( S \) is an element of \( S^n \), written \( t \) or \( (t_1, \ldots, t_n) \). For \( 1 \leq i \leq n \), \( t[i] \) denotes the \( i \)th element \( t_i \) of \( t \), and \( t[i] \leftarrow t'_i \) represents a copy of \( t \) where \( t[i] \) is replaced by \( t'_i \). Given \( t \in S \), we write \( t^n \) the vector of length \( n \) such that \( (\forall 1 \leq i \leq n) t^n[i] = t \). Given \( I \subseteq \{1,2\ldots,n\} \), the projection \( t_I \) is defined by: \( t_I = (t[k_1],\ldots,t[k_m]) \) where \( \{k_i | 1 \leq i \leq m\} = I \) and \( (\forall i < j) k_i < k_j \).

Definition 6 (Network of Ltss). Let \( \bullet \not\in A_r \) be a special symbol denoting that a particular Ltss has no role in a given synchronization. A synchronization rule is a pair \((t,a)\), where \( t \) is a vector over \( A_r \cup \{\bullet\} \) (called a synchronization vector) and \( a \in A_r \). The components \( t \) and \( a \) are called respectively the left- and right-hand sides of the synchronization rule. A network of Ltss (or simply network) \( N \) of dimension \( n > 0 \) is a pair \((S,V)\) where \( S \) is a vector of Ltss of length \( n \) and \( V \) is a set of synchronization rules, whose left-hand sides are all of length \( n \). Each left-hand side \( t \) expresses a synchronization constraint on \( S \), all components \( S[i] \) where \( t[i] \neq \bullet \) having to take a transition labeled respectively \( t[i] \) altogether so that a transition labeled with the corresponding right-hand side \( a \) be generated in the product. More formally, let \( S[i] = (Q_i, A_i, T_i, q_{0i}) \), \( (1 \leq i \leq n) \). To \( N = (S,V) \) corresponds an Ltss \((Q,A,T,q_0)\), written \( sem(N) \) or \( sem(S,V) \), such that \( A = \{a \mid (t,a) \in V\} \), \( q_0 = (q_{01},\ldots,q_{0n}) \), and \( Q \) and \( T \) are the smallest sets satisfying both \( q_0 \in Q \) and:

\[
q \in Q, \ (t,a) \in V, \ (\forall 1 \leq i \leq n) (t[i] = \bullet \land q'[i] = q[i]) \lor q[i] \xrightarrow{t[i]} q', \ q' \in Q, \ (q,a,q') \in T
\]

Note that, by construction, the states that belong to \( Q \) are reachable. Synchronization rules must obey the following admissibility properties, which forbid cutting, synchronizations and renaming of \( \tau \) transitions and therefore ensure that safety equivalence and stronger relations (e.g., observational, branching, and strong equivalences) are congruences for networks of Ltss [Lan05]:

\[
(\exists 1 \leq i \leq n) \ (\tau \text{ is reachable in } S[i]) \implies (\exists (t,\tau) \in V) \ t[i] = \tau \\
(\forall (t,a) \in V) \ ((\exists 1 \leq i \leq n) \ t[i] = \tau) \implies (a = \tau \land (\forall 1 \leq j \leq n \setminus \{i\}) \ t[j] = \bullet)
\]

The refined interface generation technique as defined and described in [Lan06] is used to generate interfaces from a network of Ltss. In our work we use the
EXP.OPEN 2.0 tool [Lan05] which allows for the description of concurrent systems as a composition of LTSs, using either synchronisation vectors, or standard parallel composition, hiding, renaming and cut operators from several process algebras. The following syntax describes the generation of a refined interface $M_{1..n}$, from the synchronisation vectors of LTSs $M_1$ to $M_n$.

$$\begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_n \end{bmatrix} \Rightarrow M_{1..n}$$

Clearly, the higher the number of processes used to generate a refined interface, the higher the number of states this interface will end up with. This means that one can end up with a very good description of the environment of a process which however cannot be used because of its size. The ideal scenario is a trade off between the size of the interface and its effectiveness in representing the environment of a process.

3 LTS Reductions

As discussed in section 2, one can constrain the generation of an LTS state graph by using other LTSs in the network as interfaces, even if, in most cases the problem is intractable due to the large size of the base LTS. However, conjecture 1 gives us the option to use language over-approximations to gain access to smaller automata. In this section, we propose a number of LTS reduction techniques especially designed to work for effective interface generation. Rather than design each technique independantly of each other, we provide an infrastructure to reason about a class of reduction techniques, enabling us to present a number of solutions guaranteed to be correct.

We start this section by presenting the concept of an LTS structural reduction which guarantees language inclusion.

**Definition 7 (LTS Reduction).** We say that $M_2$ is a structural reduction of $M_1$ with respect to a total function $eq \in Q_1 \rightarrow Q_2$, written $M_2 \sqsubseteq_{eq} M_1$, if the following conditions hold: (i) $A_1 = A_2$; (ii) $Q_2 \subseteq Q_1$; (iii) $q \xrightarrow{a_1} q'$, implies that, $eq(q) \xrightarrow{a_2} eq(q')$; and (iv) $q_0 = eq(q_{01})$.

We simply say that an LTS $M_2$ is a reduction of another LTS $M_1$, written $M_2 \sqsubseteq M_1$, if there exists a total function $eq$ such that $M_2 \sqsubseteq_{eq} M_1$. Also, we say that $M_2$ is the reduction of $M_1$ with respect to $eq$, if $M_2$ is the (unique) solution satisfying $M_1 \sqsubseteq_{eq} M_2$.

**Proposition 1.** $\sqsubseteq$ is a reflexive, transitive relation over LTSs.

The property of structural reduction we will be using most is that it guarantees language inclusion:
Lemma 1. Given two Lts $M_i = (Q_i, A_i, T_i, q_{0i})$, with $i = 1, 2$, related with a total function $eq$, then $q \xrightarrow{1} q'$ implies that $eq(q) \xrightarrow{2} eq(q')$

Proof: The proof follows directly by induction over string $s$. □

Language inclusion follows directly from the previous lemma and the fact that $eq(q_{02}) = eq(q_{01})$:

Theorem 1. Given two Lts $M_i = (Q_i, A_i, T_i, q_{0i})$, with $i = 1, 2$, if $M_2 \subseteq M_1$, then $L(M_1) \subseteq L(M_2)$.

Consider a small system, for the purposes of illustration, whose behaviour is modelled by the first Lts $S$ as depicted in figure 1. From this Lts we know that the system is capable of first performing $a$, after which it can only perform $b$ and then sequences of $c \ b^*a$. This Lts has four states. $S'$ is the Lts generated after performing a reduction on $S$, where $eq(0) = 0$, $eq(1) = eq(2) = 1/2$ and $eq(3) = 3$. This guarantees that $L(S) \subseteq L(S')$. In $S'$ we still know that the system is initially only capable of performing $a$. However we’ve now lost the information which said that only $b$ can be performed from state 1. In $S'$ after the initial $a$ the system can now produce either a sequence of $b$ or $c$. Finally $S''$ is produced from $S'$ using function $eq$, with $eq(0) = eq(3) = 0/3$ and $eq(1/2) = 1/2$. Thanks to theorem 1 we know that $L(S') \subseteq L(S'')$. In this third language over approximation we lose further information from the behaviour of $S$. From the initial state, the system can now perform either $a$ or $b$. However we still know that from the second state following an $a$ transition, we can perform only either a sequence of $b$ or just one $c$. From this simple example, one can easily see that applying the functional composition of the two $eq$ functions applied, one gets that $S \subseteq S''$.

We are currently exploring the definition of different structural reduction techniques for the generation of interfaces. In the following sections, we describe a number of such techniques, which are then used in the case studies in section 4.
3.1 Chaos

The most straightforward structural reduction technique is that of keeping a number of states of the original system (including the initial state), and coalescing the remaining states into one which can behave chaotically, by emulating any of the remaining states:

**Definition 8.** Given an LTS $M$, and a subset of its states $Q_{in}$, $CH_{M}[Q_{in}]$ is defined to be the reduction of $M$ with respect to function $eq$, defined as follows ($\chi \notin Q$):

$$eq(q) = \begin{cases} q & \text{if } q \in Q_{in} \\ \chi & \text{otherwise} \end{cases}$$

Typically, when implemented, this reduction is applied by exploring the state-space of $M$ progressively (in a breadth-first manner), for a fixed depth, beyond which everything else is collapsed together:

**Definition 9.** Given an LTS $M$, we define $CH_{M}[n]$ to be $CH_{M}[Q_{in}]$, where $Q_{in}$ is the set of states reachable in no more than $n$ steps from the initial state.

3.2 Partition Based on Action Capability Similarity

The main idea behind this reduction technique is that of creating a state partition which groups together states that exhibit a similar local behaviour. States can be compared by checking that they have the same outgoing transitions.

**Definition 10.** Given an LTS $M$, we define $TR_{M}[out]$ to be the reduction of $M$ with respect to function $eq = outgoing$.

In this manner, for each set of possible outgoing actions, we coalesce all states with that particular capability into one state, and abstract transitions accordingly. Similarly, $TR_{M}[inout]$ looks at both incoming and outgoing actions:

**Definition 11.** Given an LTS $M$, we define $TR_{M}[inout]$ to be the reduction of $M$ with respect to function $eq(q) = (\text{incoming}(q), \text{outgoing}(q))$.

A weaker version of comparing outgoing actions, is to identify a state with a set of outgoing actions with another state with a superset of those actions. Let the maximal outgoing transition sets of an LTS to be the sets of action capabilities of states for which no state can perform a superset of:

$$\{ A \mid (\exists q \ | \ A = \text{outgoing}(q)) \land (\neg \exists q \ | \ A \subset \text{outgoing}(q)) \}$$

**Definition 12.** Given an LTS $M$, with maximal outgoing transition sets $S$, we define $TR_{M}[out]$ to be the reduction of $M$ with respect to some function $eq$ satisfying $eq(q) \in \{ A \mid A \in S \land outgoing(q) \subseteq A \}$.

Clearly, in an implementation, $eq$ has to be fixed. The most straightforward implementation, when performing the analysis on-the-fly, is to use the implicit enumeration induced when traversing the state space, and take the first solution to the constraint. Note that the computational complexity of creating the state partition is linear in the number of states of the interface we want to reduce if a hash map is used to store state partitions.
3.3 Partition Based on Label Similarity

Finally, we present a third reduction technique, similar to action capabilities, but comparing actions up to an equivalence relation, which is typically weaker than equality.

**Definition 13.** Given an $Lts\ M$, and an equivalence relation on actions $\approx$, we define $LS^\approx_M[\text{out}]$ to be the reduction of $M$ with respect to function $eq$ defined as follows:

$$eq(q) = \{a \mid \exists a' \cdot a' \in \text{outgoing}(q) \land a' \approx a\}$$

Note that $TR_M[\text{out}]$ is simply a special case of this, with $TR_M[\text{out}] = LS^{\approx}_{T}M[\text{out}]$.

Typically, in a system, labels are strings — for example, when translating a language such as LOTOS into an $Lts$, one gets labels consisting of a string describing (amongst other things) the gate over which the communication takes place. For this reason, partitioning labels based on equality of prefixes of labels provides a straightforward equivalence relation. For example, consider the $Lts$ with the following eight labels:

$$\{\text{AAA, AAB, ABA, BAA, ABB, BAB, BBA, BBB}\}$$

Matching the first two symbols of the label strings, we can create a label partition of four classes, partitioning the labels as follows:

$$\{\{\text{AAA, AAB}\}, \{\text{ABA, ABB}\}, \{\text{BAA, BAB}\}, \{\text{BBA, BBB}\}\}$$

**Definition 14.** Given an $Lts\ M$, we define $PREFIX^n_M[\text{out}]$ to be $LS^\approx_M[\text{out}]$, where $\approx$ is defined as follows:

$$l_1 \approx l_2 = \text{first}_n(l_1) = \text{first}_n(l_2)$$

Note that $\text{first}_n(s)$ gives the first $n$ symbols in string $s$.

Since we usually would rather put a limit on the number of classes in the partition, we define $PF^n_M[\text{out}]$ to be $PREFIX^n_M[\text{out}]$, maximising $i$, such that $\approx$, the partition used contains no more than $n$ label classes.

4 Case Studies

4.1 Using $Lts$ Reductions with Refined Interfaces

$Lts$ reduction techniques are well suited to be combined with refined interface generation, since the size of a refined interface sometimes makes it impossible to use in practice. Since in our $Lts$ language over-approximations we guarantee that our interfaces include the traces of the environment, we can replace $M_1 \parallel_G M_2$ with $M_1 \parallel_G M_2'$, where $M_2 \subseteq M_2'$.

Consider for instance, the composition expression $E_1 = (M_1 \parallel_G M_2) \parallel_G (M_3 \parallel_G M_4)$. If the number of states in $M_1$, $M_2$ and $M_3$ is small enough, we can generate a refined interface for $M_4$ from their $Lts$. If this interface is not small enough such that it can be used with the semi-composition operator,
we can reduce the interface and use the resulting over-approximated interface in order to constrain the generation of $M_4$.

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} \Rightarrow M_{123} \Rightarrow \text{reduction of } M_{123}$$

In general, the higher the number of LTSS involved in the creation of a refined interface the better that interface would be at representing environment constraints. However, sometimes, the generation of a refined interface from the state graphs of $M_1$, $M_2$ and $M_3$ might itself not be possible due to the individual sizes of the constituent LTSS meaning that we would either produce a refined interface from two of the environments, or generate a full refined interface over reductions on the individual environments:

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \end{bmatrix} \Rightarrow \begin{bmatrix} \text{reduction of } M_1 \\ \text{reduction of } M_2 \\ \text{reduction of } M_3 \end{bmatrix} \Rightarrow \text{reduction of } M_{123}$$

After generating the refined interface from the reduced LTSS, one can, depending on the size of $M_{123}$, apply another reduction technique on $M_{123}$. Clearly the more language over-approximations applied to an interface the more generic the interface becomes. In our experiments we make use of this procedure to verify the effectiveness of our LTS reduction techniques in the generation of an ODP trader.

### 4.2 Open Distributed Processing Trader

In this section we describe the experiments carried out on the generation of an Open Distributed Processing (ODP) trader. ODP is an ISO standard (International Standard 10746, ISO — Information Processing Systems, Geneve, 1995) whose purpose is to serve as a reference model for distributed processing. The ODP framework which has been modelled in the LOTOS process algebra consists of one trader (implementing ODP) which communicates with a number of objects. These objects can either be clients, servers or both of particular services. The trader LTS consists of roughly one million states when generated on its own without environment constraints. In the experiments described here we make use of three objects to generate the refined interface for the trader. Table 1 illustrates the results achieved while incrementally increasing the complexity of the objects (by allowing them to offer and request more services). The interfaces are generated using the EXP.OPEN 2.0 (with the -interface option) tool available with the CADP toolkit. The full description and specification of the case study can be found on the CADP website\(^4\). The first trader state graph is generated through the normal process of first creating a refined interface out of the objects and then using this interface in semi-composition with the generation of the trader (Trader in table 1). This should clearly give the best

---

\(^4\) Demo 37 at [http://www.inrialpes.fr/vasy/cadp/demos.html](http://www.inrialpes.fr/vasy/cadp/demos.html)
possible reduction since the interface is giving the exact picture of the trader environment, but is also the most expensive. We then test how reduction TR performs, by first applying TR\text{\[out\]} to the LTSS of the objects. A refined interface is then generated out of these over-approximated objects. This refined interface is further reduced by applying TR\subseteq\text{\[out\]}, and is then used in semi-composition with the generation of the trader LTSS (Trader\text{TR in table 1}).

The final ODP experiment documented here is composed of three objects with a complexity (measured in number of states) of 384, 256 and 192 states respectively. If no reductions are applied to these objects, a refined interface of 15M states is produced. A state graph of this size cannot be used as an interface. On the other hand, we are able to produce a refined interface of 2.5M states when the objects are over-approximated (using reduction TR\text{\[out\]}) prior to the generation of the refined interface. TR\subseteq\text{\[out\]} is then applied on this interface to produce a final interface of 11,676 states. This interface manages to constrain the generation of the Trader state graph to 8,192 states. This shows that (in this particular case) as the complexity of the objects increases, without applying LTSS reduction techniques, we would not be able to produce a refined interface out of three objects.

### 4.3 Directory Based Cache Coherency Protocol

The second case study describes a standard cache coherency protocol for a multiprocessor architecture. The system consists of five agent processes composed in parallel with a directory process. The protocol specification guarantees the coherency of the cache maintained on the directory across the five agents that are concurrently writing to it. Table 2 reports on the time and memory usage results achieved when generating this directory based cache coherency protocol. The LOTOS specification is available online\footnote{Demo 28 at http://www.inrialpes.fr/vasy/cadp/demos.html}.

The standalone directory state graph consists of one million states and 40 million transitions. Its reduction modulo strong bisimulation produces a LTSS of 2,862 states and 1,132,544 transitions. In this experiment we first generate a refined interface from the five agents which are executing in parallel with the remote directory. This interface consists of a LTSS of 1.8 millions states and 14 million transitions. When reduced up to branching equivalence we get a directory interface of 2,560 states and 40,576 transitions.

<table>
<thead>
<tr>
<th>Obj1</th>
<th>Obj2</th>
<th>Obj3</th>
<th>Interface</th>
<th>Trader</th>
<th>Interface</th>
<th>Trader\text{TR}</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 (122)</td>
<td>64 (50)</td>
<td>64 (50)</td>
<td>901K (17K)</td>
<td>1024 / 13K</td>
<td>141K (7106)</td>
<td>1024 / 13K</td>
</tr>
<tr>
<td>256 (122)</td>
<td>128 (83)</td>
<td>96 (74)</td>
<td>2.6M (48K)</td>
<td>2048 / 30K</td>
<td>242K (1088)</td>
<td>2048 / 41K</td>
</tr>
<tr>
<td>128 (84)</td>
<td>128 (54)</td>
<td>384 (193)</td>
<td>3M (96K)</td>
<td>2048 / 20K</td>
<td>553K (1079)</td>
<td>2048 / 36K</td>
</tr>
<tr>
<td>384 (187)</td>
<td>256 (123)</td>
<td>192 (107)</td>
<td>15M (\infty)</td>
<td>\infty</td>
<td>2.4M (11676)</td>
<td>8192 / 165K</td>
</tr>
</tbody>
</table>

The possible reduction since the interface is giving the exact picture of the trader environment, but is also the most expensive. We then test how reduction TR performs, by first applying TR\text{\[out\]} to the LTSS of the objects. A refined interface is then generated out of these over-approximated objects. This refined interface is further reduced by applying TR\subseteq\text{\[out\]}, and is then used in semi-composition with the generation of the trader LTSS (Trader\text{TR in table 1}).
Table 2: Directory-based cache coherency protocol with five Agents and one Directory Generation Technique

<table>
<thead>
<tr>
<th>Generation Technique</th>
<th>Interface Size (States/Trans)</th>
<th>Directory Size (States/Trans)</th>
<th>Time</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Safety Reduction</td>
<td>48 / 1008</td>
<td>49 / 278</td>
<td>5min 24sec</td>
<td>82Mb</td>
</tr>
<tr>
<td>CH[1]</td>
<td>1 / 14</td>
<td>50 / 350</td>
<td>1 sec</td>
<td>1Mb</td>
</tr>
<tr>
<td>TR[out]</td>
<td>56 / 1355</td>
<td>49 / 278</td>
<td>2 sec</td>
<td>1Mb</td>
</tr>
<tr>
<td>PF* [out]</td>
<td>36 / 612</td>
<td>49 / 292</td>
<td>2min 10sec</td>
<td>41Mb</td>
</tr>
</tbody>
</table>

Since interfaces can be reduced up to safety without changing their behaviour, we apply different interface generation techniques and compare them with respect to time and memory consumptions against the standard safety equivalence reduction. The reduction of the full automaton has 48 states, but takes 5 minutes 24 seconds to generate. The most significant result is that achieved with TR[out], where the interface is generated and reduced in only two seconds, with only 8 additional states. It is also important to note that with this reduced interface, we generate exactly the same directory state graph as with the full interface. We use CH[1] in order to check how much of the reduction is due to the absence of labels (which are otherwise present in the state graph we want to generate) in the interface. In this particular case we notice that most of the directory reduction is induced simply by labels which are absent in the interface. When applied with semi-composition we get a 50 state, 350 transitions directory. The main purpose of this case-study, however, is to show that different reduction techniques produce different results when applied with semi-composition. In this particular case, the effectiveness of the interface is measured on the number of transitions which are blocked in the generation of the directory. Here PF*[out] produces 292 transitions which is slightly less than the chaos interface which produces 350.

4.4 Experiments on the VLTS Benchmark Suite

Finally, we report on two benchmarks which have been set up to measure the effectiveness of LTS reduction techniques. We make use of the VLTS (Very Large Transition Systems) benchmarks\(^6\) state graphs which have been obtained from various case studies about the modelling of communication protocols and concurrent systems. Many of these case studies correspond to real life, industrial systems.

In both benchmarks, the original VLTS graphs are used to create their own interfaces. For both, the interface is created by randomly removing some of the transitions from the original VLTS graph and minimising the graph modulo branching equivalence. In tables 3 and 4, the first column indicates the technique used to generate the interface which is then used (using the PROJECTOR tool) in order to constrain the generation of the VLTS graph. The second column

shows the interface size in number of states, while the third column indicates
the percentage reduction in number of states of the interface. The fourth and
fifth columns show, respectively, the number of states (and transitions) of the
projected VLTS state graph and the percentage state reduction of the projected
Lts. In the first benchmark, PF\textsuperscript{2}[out] generates the same projected VLTS state
graph with a decrease of 16\% in the size of the interface. In the second bench-
mark, using PF\textsuperscript{4}[out] we achieve, with an interface that is 55\% smaller, the same
projected VLTS graph. Moreover, with TR[out], which is 91\% smaller than the
original interface, we achieve a 75\% state reduction of the original VLTS graph.

4.5 Discussion
The results achieved so far indicate that we can use Lts language over approx-
imations in order to produce interfaces which are effective in constraining the
 generation of Lts. In the ODP case study, we achieve interface effectiveness
which is very close to the one achieved when the most specific environment is
used. The two benchmarks which we present here, indicate that there is no one
particular language over-approximation technique that performs best in every
experiment. This was to be expected however, since what we are doing is effec-
tively applying different heuristics in the generation of interfaces.

5 Conclusion and Future Perspectives
This paper describes how Lts language over-approximations can be used to
alleviate the problem of state space explosion in compositional verification. We
have shown how an ODP trader can be generated by semi-composition with a more generic interface with fewer states. Using a reduced interface, we are able to achieve the same number of states for the ODP trader as was achieved with the original (full) interface. As the complexity of the objects composed with the ODP trader increases, the use of LTs reductions which produce language over-approximations makes it possible to generate reasonably sized and effective interfaces.

There are three main directions which, we feel, need to be further explored. The first one is that of coming up with more effective heuristics for LTs reduction. The challenge here is that of being able to design heuristics, which are on the one hand effective in describing the environment as specific as possible and on the other are not computationally intensive to produce. The best solution would be that of generating interfaces on-the-fly. The second direction is that of coming up with alternative ways of combining LTs language over-approximations with refined interfaces. Finally we plan to investigate how the static analysis of the specification of a particular composition of processes can help in the generation of constrains to include as part of the interface.

References


Comparing Title Only and Full Text Indexing to Classify Web Pages into Bookmark Categories

Chris Staff and Charlie Abela
Department of Artificial Intelligence, University of Malta
{chris.staff,charlie.abela}@um.edu.mt

Abstract. Web browser bookmark files are used to retain and organise records of web sites that the user would like to revisit. However, bookmark files tend to be under-utilised, as time and effort is needed to keep them organised. We use two methods to index and automatically classify documents referred to in 80 bookmark files, based on document title-only and full-text indexing, respectively. We evaluate the indexing methods by selecting a bookmark entry to classify from a bookmark file, and recreating the bookmark file so that it contains only entries created before the selected bookmark entry. Classification based on full-text indexing generally outperforms that based on document title only indexing. The ability to recommend the correct category at rank 1 using full-text indexing ranges from 20% to 41%, depending on the number of category members. However, combining the approaches results in an increase to 37% — 59%, but we would need to recommend up to two categories to users. By recommending up to 10 categories, this increases to 58% — 80%.

1 Introduction

Web browser bookmark files are intended to be a repository of web sites that the user would like to revisit. However, bookmark files tend to be passive. Typically, user effort is required to keep a bookmark file organised, and usually, bookmark files become disorganised over time [AB97,ABC98]. Recommendations for assisting with bookmark file organisation include “filing” new bookmark entries [AB97,ABC98]. Although many contemporary web browsers allow users to choose a destination bookmark category for the new bookmark entry, the first category presented to the user is either the top-level one (Mozilla, Firefox, Internet Explorer) or the last category in which a bookmark was created (Safari). In previous work [Bug06,SB07] we presented HyperBK, a bookmark management system that is able to recommend a destination bookmark category (folder). Only a small number of bookmark files were used in the evaluation of HyperBK. Here, we have modified our approach to indexing and classification (HyperBK2), and we have evaluated the new approach with 80 bookmark files. Whereas with HyperBK we selected up to 10 bookmark entries from each bookmark file and attempted to classify them into the original categories, this time we re-create a ‘snapshot’ of the bookmark file containing only those bookmark entries that
existed just prior to the selected web page being bookmarked, which is more realistic. In this way, we do not take future information into account, and we mimic the recommendations made to users at the time the bookmark entry would be created. HyperBK selected the highest weighted 5 terms from bookmarked web pages, combined them into a category description, and then compared them with a similarly selected set of terms from the web page to classify. If a classification was not possible, then other features, such as URLs, etc., were used to determine the recommended category. HyperBK2 uses Term Frequency and a Normalised Document Frequency (see section 3) to weight terms, creates a centroid category representation based on the term weights occurring in the category, and uses the Cosine Similarity Measure to recommend membership of a web page.

In section 2 we discuss similar systems. HyperBK2’s indexing and classification approach is discussed in section 3. The approach to evaluation is described in section 4, and the results are presented and discussed in section 5. Finally, section 6 outlines our future work and conclusion.

2 Background and Similar Systems

Bookmarking is one of the most popular ways in which people store and organise information for later use [Bru04]. However, drawbacks exist, especially when the number of bookmarks increases over time [Bru01]. Numerous tools, called bookmark managers, exist to help support users in creating and maintaining, effectively-reusable bookmarks’ lists. These can be categorised either as “centrally store and browse” [Ben06], such as HyperBK [SB07], Conceptual Navigator [Cna04] and Check&Get [Chg07] or as collaborative, such as Delicious [Del07] and BookmarkTracker [Bmt07]. Bookmark managers need to provide support when it comes to effectively classify newly created bookmarks. Xia et. al.’s approach to bookmark classification classifies a web page by considering information from neighbouring pages in the link graph [Xia06]. Neighbouring pages are referred to as parent, child, sibling and spouse. The adopted method does not rely on the appearance of labelled pages (i.e. pages whose categories are already known) in the neighbourhood of a specific page and this leads to a wider applicability. Evaluations showed that classification increased from around 70% to more than 90%, using pages from the Open Directory Project [Odp06]. Delicious [Del07], which is an online service, allows users to share bookmarks. Categorisation is aided by the use of tags, which users associate with their bookmarks. However there are no explicit category recommendations when a new bookmark is being stored. InLinx [InL03] provides for both recommendations and classification of bookmarks into “globally predefined categories”. Classification is based on the user’s profile and the web-page content. Classification of a new bookmark, according to [Ben06], is a matter of first establishing a similarity in the interests of two users and then finding a mapping between the folder location of a bookmark in the collaborator’s bookmarks’ list and that of the target-user’s bookmark hierarchy. The similarity between two user profiles is computed through the classical cosine vector-similarity. The generation of a
recommendation about the best folder in which to place a new bookmark is dependent on similarity between the collaborating partners, the folder similarity between collaborative partner and target user, and the number of times that a folder was recommended. Recommendations are however based on a computed combined user-similarity (i.e. all users who have recommended a specific folder) and a combined folder similarity (computed recommendation based on all folders the recommended folder was mapped from). A new folder is created when the recommended folder is the target user’s root folder and when the total similarity of the recommended folder falls below a certain threshold.

Classification is usually based on a set of features. However the number of features considered varies between the different classification approaches taken. In the approach suggested by [Jen01] these features for classification are reduced through the use of rough sets. This approach considers the indiscernability between two objects and tries to reduce the number of objects by keeping only one tuple in a minimal set, which is called a reduct.

Robertson’s approach to bookmark organisation and classification focuses on categorisation through visualisation [Rob98] using a technique called Data Mountain, which allows users to place documents in any position on an inclined plane in a 3D virtual environment while using 2D interaction. Data Mountain takes advantage of spatial memory (i.e. the ability to memorise spatial information, such as the geographical layout of things). This technique was compared with the favourites’ mechanism in Internet Explorer. The results showed that it provided more personalisation since it presents a whole view of the bookmarks’ space and the spatial relationships between the pages.

3 Indexing and Classification Approach

We create two indices for each document in a category and the web page to classify, using either the document title or the full-text of the document, and we compare the performance of each in the evaluation section (section 4).

A bookmark file contains references to a number of web pages that a user has bookmarked. During the indexing stage, we remove stop words, stem the remaining terms using the Porter Stemmer [Por97], and calculate the term frequency for each stem (here, the term frequency is a non-normalised term count).

Once the indexing of bookmark entries is complete, we identify the documents that are to be used to create the description of each category. We take the document index of each document \(d_1\) to \(d_N\) in a category and we merge them, calculating a term weight by summing the term frequencies (TF) of each term \(j_1\) to \(j_m\) in each document in the category, and multiplying it by the Normalised Document Frequency, \(\sum_{d=1}^{N} TF_{j_i,d} \times NDF_{j_i}\), where \(N\) is the number of documents in the category. This has the effect of reducing the weight of terms that occur in few documents in the category.

We create a representation of the web page to classify in the same way, although, obviously, the NDF is 1, so the weights of terms are their TF. We then use the Cosine Similarity Measure [SB87] to measure the similarity between the
bookmark entry to place and each category in the bookmark file snapshot. The highest ranking category, if there is one, is recommended.

### 3.1 Processing Steps

HyperBK2 is a series of Python 2.3.5 programs that process bookmark files to access and download bookmark entries; create representations of and data files for categories and bookmark entries in each bookmark file; remove script tags from each downloaded HTML file; determine the order in which bookmark entries are created within the bookmark file and category; create a full-text index (of stemmed words without stopwords, using Gupta’s Python implementation of the Porter Stemmer\(^1\)) of each downloaded HTML file; create evaluation platforms according to given criteria 4; and run and analyse the evaluation platform. Whenever we download a bookmarked web page, we create a full-text index for it, comprising the unique stems of terms, and their frequency. We also keep track of bookmark entries in the same category that may have been created during the same session. For instance, entries created up to 30 minutes apart may be considered to have been created as part of the same session. Jansen and Spink report that web researchers use a session length of anywhere between 5 minutes and 3 hours [JS06]. We use a session duration of at least 30 minutes and at least 3 hours. A group of bookmark entries created in the same session is a ‘set’.

Once we have determined the set and category members, we can create a term weight vector description of each category or set that exists in the bookmark file by merging the indices of bookmark entries of each entry in the set or category into a centroid representation, or average pseudo-document. We also create, on-the-fly, term weight vectors based on the titles of category or set members.

To make a category recommendation we take the full-text index of the bookmark entry to classify and an index derived from the title only of the entry to classify, and we compare these against each bookmark file snapshot category and set vector descriptions using the Cosine Similarity Measure [SB87]. A score of 0 means that no recommendation was made, otherwise the category with the highest score is the recommended category. Even when we use sets, we are interested only in assigning the selected bookmark entry to the correct category.

### 4 Evaluation Approach

We collected 80 real bookmark files. Each bookmark file is built according to the Netscape bookmark file format\(^2\), and stores the date that each bookmark entry was created. We use the ADD-DATE field to re-create the bookmark file as a snapshot of its state just prior to the addition of the bookmark to be classified. The basic method of evaluation for HyperBK2 is to select bookmark entries from a number of bookmark files, according to some criteria, and to measure the

\(^1\) [http://tartarus.org/martin/PorterStemmer/python.txt](http://tartarus.org/martin/PorterStemmer/python.txt)

ability of the indexing and classification methods to recommend their original category. We measure the presence of the target category from ranks 1 to 5.

The criteria we use to select bookmark entries for classification from a bookmark file, to determine the eligibility of the bookmark file snapshot to participate in the particular run, are ENTRY-TO-TAKE, SET, and NO-OF-CATEGORIES. SET is false or true, depending on whether a bookmark entry is to be taken from a category, or whether the entry should be taken from a set of entries created in the same session within a category, respectively. We measure a session over either 30 minutes or 3 hours to determine if there is a significant difference in the ability to classify a web page over a longer session time.

ENTRY-TO-TAKE is the \( n \)th entry in a category (or set) that is selected for classification. We expect a category (or set) to contain \( n-1 \) entries before we select a bookmark entry for classification. If there is a problem with the bookmark entry selected (i.e., it no longer exists, etc.), then we take the next entry in the set or category, if possible. We ran HyperBK2 with values for ENTRY-TO-TAKE of 2, 4, 6, 7, 8, 9, and 11. For example, in the simplest case (ENTRY-TO-TAKE = 2), the second entry created in a category/set would be selected for classification, and a snapshot of that category would contain only one entry.

Finally, NO-OF-CATEGORIES is the number of categories that must exist in a snapshot of a bookmark file for the bookmark file to participate in the evaluation. We imposed a minimum of 5 categories, which would give a random classifier a maximum 20% chance of correctly assigning a selected bookmark entry to its original category. For instance, if the snapshot of the bookmark file contains less than 5 categories, then we will not include that snapshot in the evaluation. We wanted to see if we would bias results in HyperBK2’s favour if we did not impose this minimum, so we removed this constraint for the evaluation platform with the best performing criteria.

In all we ran eighteen evaluation platforms with the following characteristics. All but two of the platforms contained bookmark file snapshots with a minimum of 5 categories. Each run evaluated the algorithms on either categories or sets. Sets were composed mainly of bookmark entries that were created within 30 minutes of each other (two platforms used a session length of 3 hours). The bookmark entry to classify was the 2nd, 4th, 6th, 7th, 8th, 9th, or 11th entry created in the category (or set). If the web page of the bookmark entry no longer exists, or if it was in a format other than HTML or XML, then we selected the next bookmark entry that satisfied the criteria.

The evaluation platform that gave the best results (see section 5) classified the 8th bookmark entry in a category, with a NO-OF-CATEGORIES of 5. To see if the number of categories and the session length had a significant impact on results, we also ran the evaluation platform with a 3 hour session length (maximum of 3 hours between the creation time of bookmark entries in the same category considered to belong to the same set) and no minimum number of categories (so that even if a bookmark file snapshot contained only the category from which the bookmark entry for classification was selected, we still included it in the evaluation). The session length had no significant impact. Removing the mini-


minimum number of categories enabled more bookmark entries to participate in the evaluation (table 2), although there was no significant impact on classification accuracy (tables 3 — 5).

5 Results

In this section, we describe the general properties of the bookmark files that we collected, in terms of the number of categories that they contain and we present and discuss the results of the runs (tables 3 to 4) in tabular format, highlighting the best performances. On average, bookmark files used in the evaluation have

<table>
<thead>
<tr>
<th>No. of categories</th>
<th>No. of bookmark files</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>2–5</td>
<td>28</td>
</tr>
<tr>
<td>6–10</td>
<td>13</td>
</tr>
<tr>
<td>11–20</td>
<td>10</td>
</tr>
<tr>
<td>21–50</td>
<td>9</td>
</tr>
<tr>
<td>51–100</td>
<td>8</td>
</tr>
<tr>
<td>101+</td>
<td>4</td>
</tr>
</tbody>
</table>

23 categories, with a minimum of 1 and a maximum of 229. Table 1 gives the approximate number of categories in each bookmark file used in the evaluation. 8 files (10%) contain only one category. 51 files (63.75%) contain between 2 and 20 categories, and 21 (26.25%) contain more than 20 categories. We conducted

<table>
<thead>
<tr>
<th>Run</th>
<th>1/2/3/4</th>
<th>5/6/7/8</th>
<th>9/10</th>
<th>11/12</th>
<th>13/14</th>
<th>15/16</th>
<th>17/18</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTRY-TO-TAKE</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Session length (mins)</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>180</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>NO-OF-CATEGORIES</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>totalEligibleEntriesSet</td>
<td>1567</td>
<td>626</td>
<td>383</td>
<td>318</td>
<td>259</td>
<td>261</td>
<td>304</td>
</tr>
<tr>
<td>totalEligibleEntriesCat</td>
<td>1373</td>
<td>813</td>
<td>565</td>
<td>470</td>
<td>395</td>
<td>395</td>
<td>470</td>
</tr>
<tr>
<td>inBothTotal</td>
<td>1064</td>
<td>567</td>
<td>372</td>
<td>310</td>
<td>253</td>
<td>255</td>
<td>281</td>
</tr>
<tr>
<td>inCatOnlyTotal</td>
<td>309</td>
<td>246</td>
<td>191</td>
<td>160</td>
<td>142</td>
<td>140</td>
<td>189</td>
</tr>
<tr>
<td>inSetOnlyTotal</td>
<td>503</td>
<td>59</td>
<td>11</td>
<td>8</td>
<td>6</td>
<td>6</td>
<td>23</td>
</tr>
<tr>
<td>Percentage inBoth</td>
<td>57</td>
<td>65</td>
<td>65</td>
<td>65</td>
<td>63</td>
<td>64</td>
<td>57</td>
</tr>
</tbody>
</table>

the evaluation as follows. From each bookmark file, all the bookmark entries that satisfied the criteria (section 4) were extracted, and a snapshot of the bookmark
file was created per selected bookmark entry. The category and set evaluations may have selected different, but possibly overlapping, bookmark entries from the same bookmark file. On average, more bookmark entries were selected by category than by set. 60%±5% were selected by both, and the focus of the evaluation in this paper is on the results of these (table 2). There is no significant increase in the number of bookmarks created within the same category in the same session when the session length is 3 hours (compare Runs 9/10 and 11/12 in table 2), suggesting that bookmarks are created in bursts.

Table 3 shows the percentage of correct category recommendations at rank 1 and the cumulative percentage of correct categories appearing in the recommendations from ranks 2 to 5 for the approach to classification using full-text indexing for both sets and categories. The ranks of correct results returned by each approach are shown in alternating lines to demonstrate that there is a similarity in performance throughout. Basically, it does not seem to matter whether the indexing is based on a set or on a category, because the recommendation made is very nearly the same. Similar behaviour is displayed when a title-only approach to indexing is taken, although the percentages are generally slightly worse (table not shown due to space restrictions). As the performance of the algorithm appears to be independent of sets and categories, we will concentrate on only the full-text and title-only indexing and classification approach based on categories (table 4). We see that from rank 2 onwards, there is an advantage of the full-text indexing approach over the title-only approach (table 4). It also turns out that the title-only approach and the full-text indexing approach are frequently making different recommendations (table 8), even though either approach based on categories or sets make similar recommendations3. Ideally,

3 Full-text indexing on a set and full-text indexing on a category will make the same recommendations R1. Title-only indexing on a set will make the same recommendations R2 as title-only indexing on a category, but R1 and R2 will not be identical.
the top ranking recommendation is the target category. However, the best performance was 44% and worst was 24% (FTC rank 1, both table 3) and 44% and 24% respectively for classification based on sets (FTS rank 1, also table 3). When we merge the results (table 5) we see a significant increase in accuracy — although users would need to be shown a larger number of recommended categories. In future work, we intend to analyse the documents to discover if we are able to predict which approach to use so that we may reduce the number of recommended categories. In tables 6 and 7 we show, for the best performing

Table 4: Comparison of results of full-text indexing (FTC) and title-only indexing (TC) for categories (percent) \[E=ENTRY-TO-TAKE/ S=SESSION/ N=NO-OF-CATEGORIES\]

<table>
<thead>
<tr>
<th>E/S/N</th>
<th>2/30/5</th>
<th>4/30/5</th>
<th>6/30/5</th>
<th>7/30/5</th>
<th>8/30/5</th>
<th>8/180/5</th>
<th>8/30/1</th>
<th>9/30/5</th>
<th>11/30/5</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC rank 1</td>
<td>26</td>
<td>32</td>
<td>33</td>
<td>41</td>
<td>39</td>
<td>38</td>
<td>38</td>
<td>29</td>
<td>40</td>
</tr>
<tr>
<td>FTC rank 1</td>
<td>24</td>
<td>27</td>
<td>38</td>
<td>43</td>
<td>43</td>
<td>44</td>
<td>44</td>
<td>36</td>
<td></td>
</tr>
<tr>
<td>TC rank 2</td>
<td>31</td>
<td>39</td>
<td>49</td>
<td>50</td>
<td>46</td>
<td>45</td>
<td>45</td>
<td>41</td>
<td>48</td>
</tr>
<tr>
<td>FTC rank 2</td>
<td>31</td>
<td>36</td>
<td>48</td>
<td>50</td>
<td>50</td>
<td>53</td>
<td>54</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>TC rank 3</td>
<td>33</td>
<td>42</td>
<td>53</td>
<td>49</td>
<td>48</td>
<td>48</td>
<td>48</td>
<td>44</td>
<td>51</td>
</tr>
<tr>
<td>FTC rank 3</td>
<td>36</td>
<td>44</td>
<td>56</td>
<td>56</td>
<td>57</td>
<td>60</td>
<td>60</td>
<td>55</td>
<td></td>
</tr>
<tr>
<td>TC rank 4</td>
<td>33</td>
<td>44</td>
<td>54</td>
<td>52</td>
<td>51</td>
<td>51</td>
<td>51</td>
<td>45</td>
<td>53</td>
</tr>
<tr>
<td>FTC rank 4</td>
<td>40</td>
<td>48</td>
<td>61</td>
<td>62</td>
<td>59</td>
<td>62</td>
<td>64</td>
<td>59</td>
<td></td>
</tr>
<tr>
<td>TC rank 5</td>
<td>34</td>
<td>44</td>
<td>47</td>
<td>55</td>
<td>52</td>
<td>51</td>
<td>51</td>
<td>45</td>
<td>55</td>
</tr>
<tr>
<td>FTC rank 5</td>
<td>43</td>
<td>52</td>
<td>65</td>
<td>63</td>
<td>64</td>
<td>66</td>
<td>70</td>
<td>63</td>
<td></td>
</tr>
</tbody>
</table>

evaluation platform configuration (taking the 8th entry in a category to classify), the relationship between accuracy from rank 1 to 5 and the number/percentage of categories in the corresponding bookmark file snapshot respectively.

Table 5: Merging recommendations from different approaches gives higher precision/recall (percent) \[E=ENTRY-TO-TAKE/ S=SESSION/ N=NO-OF-CATEGORIES\]

<table>
<thead>
<tr>
<th>E/S/N</th>
<th>2/30/5</th>
<th>4/30/5</th>
<th>6/30/5</th>
<th>7/30/5</th>
<th>8/30/5</th>
<th>8/180/5</th>
<th>8/30/1</th>
<th>9/30/5</th>
<th>11/30/5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rank 1</td>
<td>37</td>
<td>43</td>
<td>52</td>
<td>56</td>
<td>59</td>
<td>59</td>
<td>59</td>
<td>53</td>
<td>53</td>
</tr>
<tr>
<td>Rank 2</td>
<td>46</td>
<td>52</td>
<td>61</td>
<td>67</td>
<td>67</td>
<td>67</td>
<td>69</td>
<td>63</td>
<td>64</td>
</tr>
<tr>
<td>Rank 3</td>
<td>51</td>
<td>59</td>
<td>68</td>
<td>74</td>
<td>74</td>
<td>76</td>
<td>76</td>
<td>69</td>
<td>68</td>
</tr>
<tr>
<td>Rank 4</td>
<td>55</td>
<td>64</td>
<td>73</td>
<td>78</td>
<td>77</td>
<td>77</td>
<td>78</td>
<td>73</td>
<td>72</td>
</tr>
<tr>
<td>Rank 5</td>
<td>58</td>
<td>66</td>
<td>75</td>
<td>79</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>76</td>
<td>76</td>
</tr>
</tbody>
</table>

Discounting bookmark files with only one category, the overwhelming majority of bookmark files (71%) have between 2 and 20 categories (table 1). Accuracy is highest at rank 1, with an accuracy of 65.6% and 53% for bookmark files containing 6–10 and 11–20 categories respectively. With bookmark files that
contain 6–10 and 11–20 categories (32% of submitted bookmark files with more than one category and 33% of bookmark file snapshots), accuracy at rank 5 is 93.7% and 96.5% respectively. This means that for bookmark files with these numbers of categories, the ability to place the target category into the top 5 recommendations is almost complete. Of course, showing the user a maximum of 10 recommended categories when the bookmark file contains only 10 categories appears to be self-defeating, but at least the recommendations would be ordered, with the user’s preferred category ranked 1 about 66% of the time. The worst performing class of bookmark file contains between 51 and 100 categories (68% at rank 5). As part of future work, we intend to measure the consistency (or cohesiveness) of bookmark file categories. This will allow us to measure the average similarity of entries in a bookmark file, to see if there is a relationship between classification accuracy and cohesiveness.

Table 8 gives the performance of title-cat and full-text-cat indexing and recommendation approaches, again broken down by numbers of categories in the bookmark file snapshots. Full-Text indexing recommends the user’s preferred category in a higher rank more frequently than title-only indexing, except, surprisingly, when a bookmark file contains more than 50 categories.
Table 8: Which of Title-Cat (TC) and Full-Text-Cat (FTC) ranks higher?

<table>
<thead>
<tr>
<th>Categories</th>
<th>Equal</th>
<th>TC Higher Rank</th>
<th>FTC Higher Rank</th>
<th>No Recommendation</th>
</tr>
</thead>
<tbody>
<tr>
<td>2–5</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>6–10</td>
<td>7</td>
<td>4</td>
<td>19</td>
<td>2</td>
</tr>
<tr>
<td>11–20</td>
<td>17</td>
<td>9</td>
<td>23</td>
<td>2</td>
</tr>
<tr>
<td>21–50</td>
<td>11</td>
<td>9</td>
<td>30</td>
<td>14</td>
</tr>
<tr>
<td>51–100</td>
<td>7</td>
<td>11</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>101+</td>
<td>18</td>
<td>18</td>
<td>11</td>
<td>13</td>
</tr>
</tbody>
</table>

6 Future Work and Conclusions

We have extended work previously conducted on HyperBK in the area of automatic bookmark classification by comparing indexing and classification methods based on vector-based full-text and title-only representations of documents in a bookmark category. Additionally, we investigate whether there is any significant advantage to grouping bookmark entries within a category by time of creation (to form a set of bookmark entries created within the same session). We conducted several runs in which the bookmark entry to be selected for classification was the 2nd, 4th, 6th, 7th, 8th, 9th, or 11th entry created in a category (or set). We found that there appears to be no advantage to grouping bookmarks into sets of entries created during the same session. Although there was a notable difference in the numbers of participating bookmark entries (generally, less sets exist than other eligible bookmark entries), there was no noticeable improvement in recommendation accuracy. However, there is a significant difference when title-only or full-text indexing is used. Not only do the different approaches make different recommendations, one of which is likelier to be the correct category, but there appears to be a correlation between the method to use and number of categories that exist in a bookmark file.

Other future work includes determining if the approach to use (title-only or full-text indexing and classification) can be predicted from the type of document to be indexed, and whether it is worth defining a cohesiveness function to measure the relative similarity of documents in a category, to determine the likelihood of the category performing well in classification tasks.

We currently need to recommend a maximum of 10 different categories to the user to obtain the highest accuracy at rank 5, assuming that each approach recommended 5 completely different categories. However, on average, there is an overlap of 65% in recommended categories, so probably only around 7 different categories would be displayed. Ideally, the number of recommended categories could be reduced, using a combination of cohesiveness, prediction based on document features, and the correlation between indexing and classification approach and the number of categories in a bookmark file.
References


[Del07] Delicious, http://del.icio.us/


Author Index

Abela, Charlie, 248
Abela, John, 1, 140
Abela, Patrick, 110
Azzopardi, Bjorn, 24
Azzopardi, Carmel, 33
Azzopardi, Joel, 11
Bajada, Jo-Ann, 226
Borg, Claudia, 42, 52
Borg, Jonathan C., 66
Cachia, Ernest, 66
Camilleri, Vanessa, 82
Caruana, Gaetano, 99
Caruana, Steven, 91
Cassar, Alan, 119
Cassar, Gail, 110
Cauchi, Abigail, 130
Colombo, Christian, 140, 150
Cordina, Joseph, 162
Fenech, Jennifer, 130
Fenech, Karl, 130
Fenech, Stephen, 162
Grima, Maria, 175
Lang, Frédéric, 234
Micallef, Luana, 130
Micallef, Nicholas, 187
Montebello, Matthew, 24, 33, 82, 91, 187
Pace, Gordon J., 52, 99, 150, 162, 175, 192, 203, 234
Rizzo, Matthew, 216
Rosner, Michael, 42, 52, 226
Spina, Sandro, 234
Spiteri, Christopher L., 66
Staff, Chris, 248
Tabone, Christian, 203
Vella, Kevin, 119
Vella, Mark, 66