CSAW '06

Computer Science Annual Workshop

2006

Department of Computer Science and A.I.
University of Malta

05 ~ 06 December 2006

Organised by: John Abela, Angelo Dalli, Kristian Guillaumier

CSAW
[2006]

Thanks to our Sponsors

Acknowledgements

We would like to thank all academics, research staff and students who produced quality contributions for this edition of the workshop as well as the previous organisers for guiding us and providing invaluable advice. Special thanks goes to Gordon Pace and Ms. Eileen Darmanin for going out of their way to support us during the planning of the venue and for their feedback.

Finally, we wish to thank all our sponsors for making this event viable – Ascent Software, Bank of Valletta, Crimson Wing, GFI and iXaris.

December 2006

Angelo Dalli,
John Abela,
Kristian Guillaumier

– CSAW '06 Organisers

Contents

MLRS, a Resource Server for the Maltese Language
Mike Rosner et al.

How Did I Find That? Automatically Constructing Queries from Bookmarked Web Pages and Categories
Christopher Staff

Automatic Classification of Web Pages into Bookmark Categories
Christopher Staff

A Brief Comparison of Real-Time Software Design Methods
Tony Spiteri Staines

Automatic Interface Generation for Enumerative Model Checking
Sandro Spina, Gordon Pace

EUMEDGrid: Grid Computing in Malta and the Mediterranean
Kevin Vella et al.

Runtime Validation using Interval Temporal Logic
Karlston D'Emanuele, Gordon Pace

Service Discovery and Composition: PreDiCtS Approach

Charlie Abela
University of Malta
Department of Computer Science and AI
+356 2590 7295
charlie.abela@um.edu.mt

Matthew Montebello
University of Malta
Department of Computer Science and AI
+356 2340 2132
matthew.montebello@um.edu.mt

ABSTRACT
The proliferation of Web Services is fostering the need for service-discovery and composition tools to provide more personalisation during the service retrieval process. In this paper, we describe the motivating details behind PreDiCtS, a framework for personalised service-retrieval. In our approach we consider that similar service composition problems can be tackled in a similar manner by reusing and adapting past composition best practices or templates. The proposed retrieval process uses a mixed- initiative technique based on Conversational Case-Based Reasoning (CCBR), that provides i) for a clearer identification of the user's service requirements and ii) based on these requirements, finds suitable service templates that satisfy the user's goal. We discuss how retrieval can vary through the use of different CCBR algorithms and how adaptation can be performed over the retrieved templates thus providing the personalisation feature in PreDiCtS.

Categories and Subject Descriptors
I.2.6 [**Artificial Intelligence**] Learning

General Terms
Algorithms, Performance, Design, Experimentation.

Keywords
Web Services, Conversational Case-Based Reasoning, Semantic Web

1. INTRODUCTION
Reusability and interoperability are at the core of the Web Services paradigm. This technology promises seamlessly interoperable and reusable Web components that facilitate rapid application development and integration. When referring to composition, this is usually interpreted as the integration of a number of services into a new workflow or process. A number of compositional techniques have been researched ranging from both, manual and semi-automatic solutions through the use of graphical authoring tools, discussed in [24], [22], to automated solutions based on techniques such as AI planning, used in [20], [25] and others.

The problem with most of the composition techniques mentioned above is three fold (i) such approaches attempt to address service composition by composing web services from scratch, ignoring reuse or adaptation of existing compositions or parts of compositions, (ii) it is assumed that the requester knows exactly what he wants and how to obtain it and (iii) composing web services by means of *concrete* service interfaces leads to tightly-coupled compositions in which each service involved in the chain is tied to a Web service instance. Using this approach for service reuse, may lead to changes in the underlying workflow which range from slight modifications of the bindings to whole re-designing of parts of the workflow description. Therefore in our opinion, services should be interpreted at an abstract level to facilitate their independent composition. [13] adds, "*abstract workflows capture a layer of process description that abstracts away from the task and behaviour of concrete workflows*", and this allows for more generalisation and a higher level of reusability. A system can start by considering such abstractly defined workflow knowledge and work towards a concrete binding with actual services that satisfy the workflow.

The reuse of abstract workflows brings with it a set of other issues, such as the way that these workflows are generated, stored and retrieved. Therefore when deciding on which solution to adopt we considered the following motivating points:

- Reusability of compositions has the advantage of not starting from scratch whenever a new functionality is required.
- For effective reusability, a higher level of abstraction has to be considered, which generalises service concepts and is not bound to specific service instances.
- Personalisation of compositions can be achieved by first identifying more clearly the user's needs and then allowing for reuse and adaptation of these past compositions based on these needs.
- Compositions can be bound with actual services thus making them concrete.

In our approach we wanted to put the user (developer or otherwise) in a situation whereby he can reuse existing templates. Infact this approach is similar to that adopted in [21], [25], [11], and [27] which use pre-stored abstract workflow definitions or templates in their composition framework.

This kind of reusability has been widely investigated in work related to Case-Based Reasoning (CBR), which is amenable for storing, reusing and adapting past experience for current problems. Nevertheless CBR restricts the user to define a complete problem definition at the start of the case-retrieval process. Therefore a mixed-initiative technique such as CCBR [6] is more appropriate since it allows for a partial definition of the problem by the user, and makes use of a refinement process to identify more clearly the user's problem state.

In this paper we want to present, the motivation behind, and a prototype of the PreDiCtS framework. Through this framework we allow for i) the encoding and storing of common practices of compositions or templates within cases and ii) for the retrieval, reuse and adaptation of these cases through CCBR.

PreDiCtS' case definition is based on the CCBROnto ontology. This ontology is based on OWL and has been discussed in [3] and [4]. Each case definition is composed of three main components

that capture different knowledge related to a particular service template. The case-context defines information related to the case creator and provides a means through which a case-utility history is maintained. Each case encodes the problem to which it can provide a solution in the form of a set of question-answer pairs (qapairs). It is through this set of qapairs that the retrieval process can present the solution which represents the service workflow definition. Each solution is defined through an OWL-S [19] service definition. This includes both service profile and process descriptions. The latter is important since it defines the actual service workflow information.

Given a new problem or service template request, the PreDiCtS' approach allows first to retrieve a ranked list of past, similar templates which are then ranked and suggested to the requester. Through a dialogue process the requester can decide when to stop this iterative-filtering phase, and whether to reuse or adapt a chosen case.

In a future extension to this work it is envisioned that, given a suitable case, a mapping is attempted between the features found in the chosen template, to actual services found in a service registry. An AI planning component can be used at this stage to handle this mapping from an abstract to a concrete, executable workflow.

The rest of this paper is organized as follows. In Section 2 we will give some brief background information on CCBR and its application in various domains. In Section 3 we will present the architecture of PreDiCtS and discuss implementation details mainly focusing on the case-creator and case-retriever components, making references to a typical travelling scenario. We evaluate the prototype in Section 4 and in Section 5 we discuss future work and extensions. In the final section we provide some concluding results.

2. CONVERSATIONAL CASE-BASED REASONING

Case-Based Reasoning is an artificial intelligence technique that allows for the reuse of past experience to solve new problems. The CBR process requires the user to provide a well-defined problem description from the onset of the process, but users usually cannot define their problem clearly and accurately at this stage. On the other hand, CCBR allows for the problem state to be only partially defined at the start of the retrieval process. Eventually the process allows more detail about the user's needs to be captured by presenting a set of discriminative and ranked questions automatically. Depending on the user's supplied answers, cases are filtered out and incrementally the problem state is refined. With each stage of this problem refinement process, the system presents the most relevant solutions associated to the problem. In this way the user is kept in control of the direction that this problem analysis process is taking while at the same time she is presented with solutions that could solve the initial problem. If no exact solution exists, the most suitable one is presented and the user is allowed to adapt this to fit her new requirements. Nevertheless, this adaptation process necessitates considerable domain knowledge as explained in [18], and is best left for experts.

One issue with CCBR is the number of questions that the system presents to the user at each stage of the case retrieval process. This issue was tackled by [14] which defined qapairs in a taxonomy and by [2] through the use of knowledge-intensive similarity metrics. In PreDiCtS we took into account the possibility that the user opts to use different similarity measuring algorithms for different domains. Infact two approaches are allowed (with the possibility of adding others). One of these approaches is based on the similarity measure defined in [6] and used by [26] to handle workflow reuse. Another similarity measure is based on the taxonomic theory defined in [14]. Through this similarity technique, the abstract relations between qapairs and in particular the sub-class relation are considered to reduce the number of questions that the user is presented in each retrieval cycle.

2.1 Uses of CCBR

CCBR is mostly associated with customer-support systems, though its benefits have been tested in various other fields such as, business process and workflow management, software-component retrieval and in connection with Recommendation systems. In what follows we will consider the above scenarios in more detail.

2.1.1 Business Process and Workflow Management

Weber in her thesis [26] combines CCBR with rules and presents a solution for business process management. The created prototype is called CBRFlow and allows for more flexibility and adaptability in the management of workflows. The adopted hybrid approach takes the best of rule-based and case-based reasoning, though the rule-based component is allowed to have some precedence over the case-based component. Rules are generated from domain knowledge while case-based reasoning is used when no rules are available or updates to a rule exist in the form of cases.

The CCBR component uses the same case-similarity metric as that described by [6]. This similarity is computed by finding the difference between the number of the shared and conflicting observations, and then dividing the result by the total number of observations in a case. A normalisation function is used to set the result within the interval [0, 1].

2.1.2 Software Component Retrieval

In [1] the CCBR technology is used to solve the problem of software component retrieval, especially when the number of components involved is large. The proposed solution is called Conversational Component Retrieval Model or CCRM. A case represents a component and a knowledge-intensive CBR methodology is used to explore the context-based similarities between the user's query and the stored components.

A frame-based knowledge representation and reasoning system called CREEK [10] is used to unify the component-specific cases and the general domain knowledge. A knowledge-intensive similarity calculation is used to determine which knowledge in the knowledge base is relevant to the retrieval process and to calculate the similarity between a new case and the stored cases.

The question-answer interaction during the conversation is motivated by the fact that qapairs are easily understood and that the most informative and discriminating ones are presented to the user during a conversation. For this reason a set of predefined questions together with possible answers for each slot (i.e. for each relation between two concepts) are specified and an

information-gain metric algorithm is used to quantitatively measure the information that each slot can provide.

In our work we intend to resort to such frame structures through the use of OWL ontologies, in particular CCBROnto. We define cases whose solutions are service templates. These templates will be defined through a process definition language, such as OWL-S, though it is possible to use other languages, such as WS-BPEL.

2.1.3 CCBR and Recommendation Systems

[18] presented a web-based CCBR solution which is able to recommend solutions to scientist seeking resources (such as codes and data) related to an Earthquake Simulation Grid provided by the ServoGrid project [23].

A number of grid related ontologies were developed in RDF and these are used to represent case descriptions. Thus a case is considered to be a set of RDF triples. A domain independent CBR engine based on the Indiana University Case-Based Reasoning Framework (IUCBRF) [16] is used.

The implemented prototype uses the RDF ontologies to present questions about the desired resource characteristics and, typically to the CCBR process, which ranks cases based on the chosen answers. During each iteration, the system provides discriminating questions in a ranked order so that the irrelevant cases are incrementally filtered out.

Each case definition contains the problem and solution descriptions together with bookkeeping information such as the time of case creation, the contexts in which the case applies and also source or provenance information. Both the problem and solution are represented by a set of predefined features, where each feature is an RDF triple. During case-base initialisation, all possible <*predicate - predicate value*> pairs are extracted from the ontology and presented as features. The case retrieval mechanism is based on a threshold method which compares the set of features present in both user and case-problem definitions. Cases are ranked based on the number of common features whose values are consistent. Cases with unknown features or having inconsistent feature values are eliminated from the process.

The way in which cases are defined through RDF is consistent with how we envision our own solution. Though in this case, all generated triples are equally considered as possible qapairs. Furthermore, it seems that no reasoning was done on the RDF data, thus no advantage was taken from this when qapairs were presented to the user. In our solution we want to be able to exploit as much as possible the logic behind the concepts and relations within a case description by using an OWL reasoner. For example given that, a question related to some particular concept has already been presented to the user, it is superfluous to present another question whose concept is more generic than the one associated with the previous question.

2.2 Taxonomic CCBR

Taxonomic CCBR (TCCBR) tries to tackle the pervasive issue of expressing case contents and features at different levels of abstraction. The solution is based on the ability to make use of feature taxonomies.

The motivation behind the use of TCCBR is highlighted by three sources of abstraction:

- the different levels of domain expertise between users and developers
- the variations in information availability and the cost of acquiring it
- the variations in decision-making needs

If abstraction is ignored then problems such as unwanted correlation between features, redundancy in the number of questions presented to the user during conversation and inconsistencies in the case representations when new features are added are most likely to occur. TCCBR is defined to include:

- A set of questions which are used for indexing the cases. Each question can be associated with a set of answers.
- A set of taxonomies each one representing a set of qapairs which are related through either an *is-a-type-of* or *is-a-part-of* relation.
- A set of cases each having a problem definition in the form of a set of qapairs and a solution.

Furthermore, in TCCBR two important rules have to be applied to the set of qapairs in a defined case:

i. Only one qapair from a particular taxonomy can be included in each case

ii. The most specific available and applicable qapair is used to represent the case

The process of TCCBR as explained by [14] is divided into three main tasks (the third is optional though):

i. Case retrieval

ii. Conversation

iii. Case Creation

Case retrieval is subdivided into three main steps referred to as searching, matching, ranking and selecting. During this phase, cases are retrieved and ranked based on the questions that the user has chosen to answer. On the other hand the conversation process involves the identification and ranking of the most appropriate questions to present to the user after each iteration. If no suitable solution is found then a new case may be defined by specifying a new set of questions (or reuse existing questions) and a solution for this new problem.

The approach taken in TCCBR is very relevant to our research goal and infact this is one of the retrieval techniques adopted in PreDiCtS. The main theory behind TCCBR is discussed in detail in [14] and though we will make reference to this work we will not explain it here. Nevertheless in what follows we will explain in detail any deviations that we have taken from this original theory.

3. PreDiCtS

The PreDiCtS framework allows for the creation and retrieval of cases (the adaptation process is in the pipeline). The respective components that perform these two tasks are the *CaseCreator* and the *CaseRetrieval* (See Figure 1). PreDiCtS is written in Java and is developed in Eclipse. It uses a MySQL database to store the cases, which are based on CCBROnto, and makes use of both Jena and the OWL-S APIs.

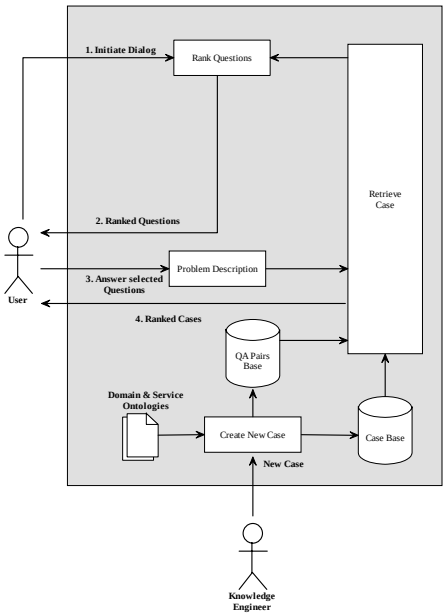

Figure 1: CCBR cycle adopted in PreDiCtS

To explain how PreDiCtS can be used to create and retrieve service templates we will make use of a typical travelling scenario, described in the next section. We will then explain how cases, which represent different problems related to this domain and their respective solutions, are created through the *CaseCreator*. Retrieval is handled by the *CaseRetrieval* component which allows the user to adopt different CCBR algorithms to find the cases with the most suitable service template. Figure 2 represents the main components in our framework.

3.1 Travelling Scenario

The travelling situation that we want to model here is related to an academic who wants to go abroad to attend a conference. The defined cases should represent the problem from an advisor's perspective and present a solution based on this knowledge. An advisor in this situation could have the role of a travelling agent, who is asking his client questions to identify what the latter requires so that he can eventually suggest the best solution.

Goal: User is to attend an **event** on some particular **date** in some particular **location**. A part of a travelling domain ontology is shown in Figure 3.

Looking at the ontology it is noticed that the concept *Person* is associated with three disjoint branches or taxonomies, *Event*, *Accommodation* and *Transport*. Thus the questions should be related to any of these taxonomies.

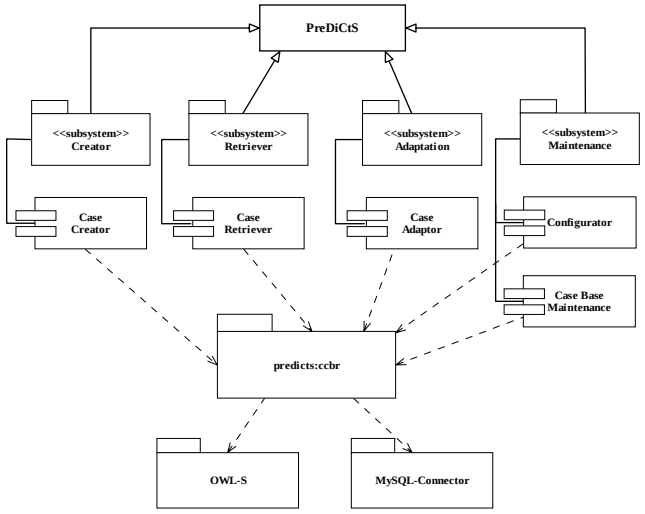

Figure 2: System Component Summary

After having identified the important aspects of the domain, we start by looking at the ontology to identify which typical questions might be asked in this situation. Questions should ideally capture a single aspect of the domain. For example, the most generic questions that are immediately identified are: *Do you want to attend for an Event?*, *Do you want to use Transport?* and *Do you want to reserve an Accommodation?*. Other questions, such as *Do you want to use a Plane?* And *Do you want to stay in a Hotel?* can be considered as being subsumed by the former set. The associated answer types for such questions are typically either a *Yes* or a *No*.

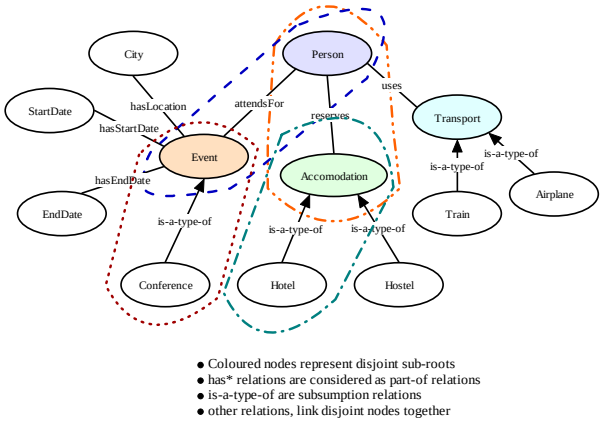

Figure 3: Travelling Domain Ontology

In Table 1 below we have listed some of these questions and have also associated them with a triple from the ontology. Each qapair is assigned a unique QID reference for that particular set. The appropriate link between the set of qapairs and the solution has to be defined by the creator. We adopt the methodology presented by [17] in which domain and task-knowledge are linked together.

Table 1: qapairs set for the Travelling Domain

QID	Description	Triples Set
		<Subject, Predicate, Object>
1	Problem is a Travelling Problem?	<TravellingProblem, subClassOf, Problem>
2	Do you want to attend a Conference?	<AttendConference, subClassOf, TravellingProblem>
3	Do you need transportation?	<Transportation, subClassOf, AttendConference>
4	Do you want to use a plane?	<Airplane, subClassOf, Transportation>
5	Do you want to use a train?	<Train, subClassOf, Transportation>
6	Do you want accommodation?	< Accommodation , subClassOf, AttendConference >
7	Do you want to stay in a hotel?	<Hotel, subClassOf, Accommodation>
8	Do you want to stay in a hostel?	<Hostel, subClassOf, Accommodation>
9	Is Conference registration required?	<Conference, subClassOf, AttendConference>

The domain is used to provide datatype information relevant to the service inputs and outputs. In our case the task knowledge is defined through an OWL-S definition. Thus for example the triple *<Hotel, subClassOf, Accommodation>* will provide, in the solution, a generic place holder for a *Hotel Reservation* service. Other services that might be useful to include in the solution are *Flight Booking*, *Train Reservation*, *Hostel Reservation* and *Conference Registration* services. Figure 4 represents a UML Activity Diagram of a particular solution for this domain. The use of this graphical representation to define a service workflow has also been adopted by [22] and [8].

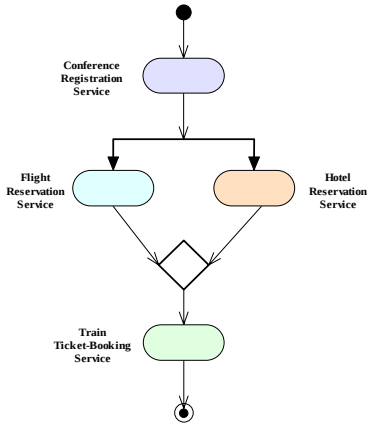

Figure 4: UML Activity Diagram for the Service Workflow

3.2 Case Creation

During case creation the expert user can define and add a new case to the case base. As already explained earlier, in CCBR a case consists of a case description and a problem and solution state. In PreDiCtS though, a case c_i is defined as a tuple:

$c_i = (dsc_i, cxt_i, \{q_1a_3....q_ia_j\}, act_i, hst_i)$ where;

dsc_i is a textual description for the particular case.

cxt_i represents a set of context related features, such as *Role* and *CaseCreator* information based on the *foaf:RDF* ontology definition.

$\{q_1a_3....q_ia_j\}$ is a representation of the problem state by a set of qapairs

act_i denotes the solution which is represented by service composition knowledge stored in an abstract template.

hst_i, is the usage history associated with each case.

Each case is based on the CCBROnto ontology which is described in more detail in [4] and can be found at [9]. The *CaseCreator* UI (see Figure 5) allows the user to add all the necessary information which is then translated into a CCBROnto case-representation in a manner transparent to the user.

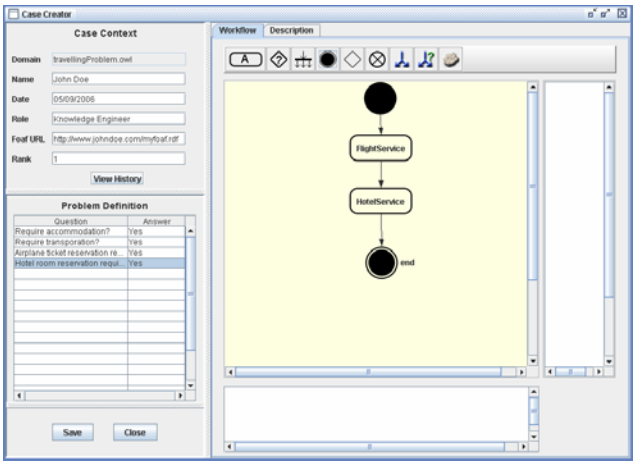

Figure 5: CaseCreator UI in PreDiCtS

The context description *cxt*, is important in an open-world such as the Web since this will be used as a discriminating feature during case retrieval. The action or solution definition *act*, represent the service compositional knowledge and can be defined through any composition language. In PreDiCtS we are using parts of an OWL-S service description, but the framework can be easily extended to work with other service languages such WS-BPEL. hst_i is an other feature which represents the usage-history of each case. This history could provide either positive (i.e. case was found useful) or negative feedback (i.e. implying that aspects of the case were not found ideal by past users) to the case user. This history information is used to generate a reputation value during case retrieval.

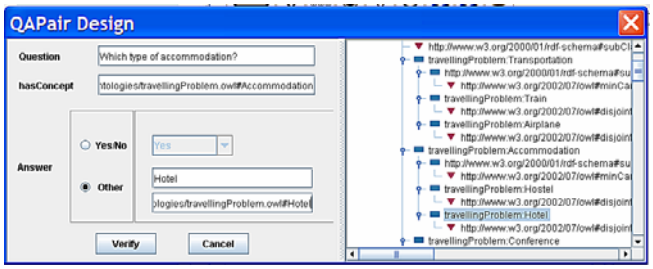

Figure 6: Question-Answer Pair Design Tool

An important aspect to consider when creating a new case is the definition of the problem through a set of qapairs. PreDiCtS' retrieval component uses two approaches to find suitable cases, one of which is based on an adapted version of TCCBR. For this reason a special qapairs-creation tool, shown in Figure 6, is provided that allows the user to easily associate a new qapair definition with domain ontology concepts.

Some adaptations have been made to the TCCBR theory to allow the system to work with ontologies and to be able to handle the open-world aspects required when defining the service template.

3.2.1 Rule 1

Only one qapair from a taxonomy can be included in a case (i.e. there is no abstract relation between concepts relating each qapair

in case). This is similar to TCCBR, unless these concepts associated to these qapairs are specifically defined as disjoint within the taxonomy.

Example:

```
<owl:Class rdf:ID="Hotel">
   <rdfs:subClassOf rdf:resource="#Accommodation"/>
   <owl:disjointWith rdf:resource="#Hostel"/>
</owl:Class>
<owl:Class rdf:ID="Hostel">
   <rdfs:subClassOf rdf:resource="#Accommodation"/>
   <owl:disjointWith rdf:resource="#Hotel"/>
</owl:Class>
```

Given the above situation a case can contain both questions:

Accommodation required is Hostel?

Accommodation required is Hotel ?

In this way the case covers the situation whereby a user might require staying at both a *Hotel* and a *Hostel* which are both subclasses of *Accommodation.*

3.2.2 Rule 2

The most specific available and applicable qapair is used to represent the case. We adapt this rule as is defined in the TCCBR theory. We look at a taxonomy as a dialogue composed of an ordered set of nodes (qapairs). We start by looking at both the domain of discourse and the different services that might be required (in the solution) to solve a particular issue. We extract those classes that are relevant to the problem that we want to model and give them an ordering. This ordering, though abstractly defined through the *subClassOf* relation, does not always imply that one class is in effect a *subClassOf* another, but rather that the question associated with that concept will be asked before or after another one associated with another concept. Therefore given the questions:

Accommodation required is Hotel? and *Do you need Accommodation?*, the former will be preferred over the latter because it is more specific and thus is considered to be closer to the solution.

3.2.3 Rule 3

We consider a qapair to be associated with a unique concept in the taxonomy. Thus for example, the question:

Accommodation required is Hostel? will be associated to the *Hostel* concept while *Do you need accommodation?* is associated to the *Accommodation* concept

We make use of reification to generate more knowledge about each statement. Infact a question will be associated with a reified statement that threats each component of a triple <*subject*, *predicate*, *objec*t>, as a *Resource*.

Example: For the question *Do you need accommodation?*

a reified statement with the following subject, predicate and object resources will be defined:

Subject: *Accommodation*

Predicate: *subClassOf*

Object: *AttendConference*

In this example, *Accomodation* is defined to be a *subClassOf AttendConference*. We envision that this technique will allow us, in the future, to work with other types of abstract relations such as those similar to *is-a-part-of* by considering other properties that associate classes together.

3.2.4 Service Template Creation

The case creator is provided with a visual-composer tool that allows him to easily create a workflow with the generic services that can solve a specific problem. The UML Activity Diagram representation is used to eventually generate an OWL-S Process definition. The Process ontology in OWL-S provides for the definition of a workflow of services and related properties. Since we wanted this description to be as generic as possible, each service definition is conceptually linked to an ontology of service-related concepts. Thus if the user adds a node that represents a *Flight Reservation* service, a generic atomic service definition will be generated whose input and output resources are defined by some external service-related ontology.

```
<process:AtomicProcess rdf:ID="FlightReservationService">
  <process:hasInput>
    <process:Input rdf:about="#FlightReservationInput">
      <process:parameterType
            rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"/>
    </process:Input>
  </process:hasInput>
  <process:hasOutput>
    <process:Output rdf:about="#FlightReservationOutput">
      <process:parameterType
            rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI"/>
    </process:Output>
  </process:hasOutput>
</process:AtomicProcess>
```

In this manner when searching for actual services, these generic placeholders will be bound to actual service inputs and outputs. The workflow also defines information related to the order of execution of the services.

3.3 Case Retrieval

The *CaseRetriever* is responsible for the CCBR retrieval process. It takes as input the choice of similarity measure and problem domain and presents questions for the user to answer. The answered questions will then be used to generate a list of cases based on the similarity measure component.

It is up to the user to decide whether a case from the retrieved set of cases is suitable enough to solve his problem. In the situation where further problem-filtering is required, the user can decide to answer more questions, with the consequence that the list of retrieved cases is also filtered down.

The set of questions presented with every step in this filtering process are generated through a conversation-generation component which takes care of identifying which questions are best suited to be presented to the user in the next step. Different conversation-generation algorithms are available in PreDiCtS, depending on the type of similarity measure chosen initially by the user.

3.3.1 CaseRetriever UI

This is divided into three main components (see Figure 7). The top-most pane consists of two combo boxes; one displays a list of problem-domain ontologies while the other displays the different types of similarity methodologies that the retriever is capable of

using. At present this is limited to two, the *Default* CCBR and the *TCCBR* (Taxonomic CCBR) methodologies. We envision the use of a graph-based retrieval method, *GCCBR*, based on [7], in the near future.

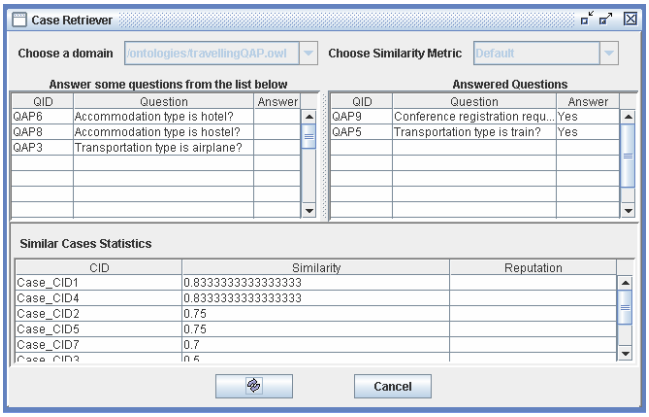

Figure 7: CaseRetriever UI

The middle left and right panes show two tables. The table on the left presents those questions that still require an answer from the requester. These are generated by the conversation-generation algorithm. They act as a filtering and refinement mechanism for the list of retrieved cases. The middle right-hand pane presents a table of already answered questions. The bottom pane also contains a table component, but this one shows the cases that have been retrieved during a CCBR retrieval cycle.

3.3.2 Default CCBR

This similarity measure is based on that presented by [6] and discussed in Section 2. The following similarity function is used:

$$sim_1(Q,C) = \frac{same(Q_{qa},C_{qa}) - diff(Q_{qa},C_{qa})}{|C_{qa}|}$$

where $|C_{qa}|$ represents the number of qapairs in the case problem definition. This function is also adapted by [26] but a normalisation function is also included to keep the similarity value between [0, 1]. The normalisation function is as follows:

$$sim_2(Q,C) = \frac{1}{2} * (sim_1(Q,C) + 1)$$

The technique used to present and rank new questions during a conversation is based on their frequency in the considered cases, though other different techniques could be utilised as defined by [2], [15] and others.

3.3.3 TCCBR

As discussed earlier in Section 2, retrieval in TCCBR is based on two processes, the case-retrieval process and the conversation generation process. We will discuss the steps associated to each one and any adaptations made in PreDiCtS.

The searching step, of the case-retrieval process in the original TCCBR starts by getting the user's textually-defined query and mapping this with the most similar qapair in the specific domain. In PreDiCtS though, the system presents the list of the most generic qapairs for a chosen domain, that is, those that are at the roots of the specific taxonomies.

The qapairs are not directly taxonomised, as explained earlier, but each question is associated with a triple *<subject, predicate, object>* which is defined in a problem related-ontology. This implicitly makes a qapair part of a taxonomy.

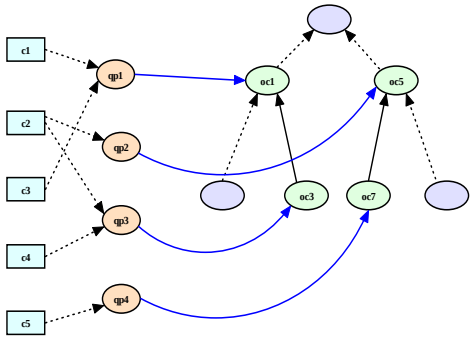

Figure 8: PreDiCtS qapair Taxonomy

Considering the situation in Figure 8 above, we assume that the set of cases, C_Q, and the set of qapairs, QP, are linked to an Ontology O as follows:

case set $C_Q = \{c_1, c_2, c_3, c_4, c_5, \ldots, c_i\}$

qapairs in $QP = \{qp_1, qp_2, qp_3, \ldots, qp_j\}$

ontology $O = \{oc_1, oc_3, oc_5, oc_7, \ldots, oc_k\}$

where, qp_1 can be seen to be present in two cases, c_1 and c_3, and is associated with ontology concept oc_1 (this is assumed to be the *subject* from the associated triple). Similarly qp_3 is associated with ontology concept oc_3 (which is the *subject* in yet another triple) and is present in two cases, c_2 and c_4. In this way qp_1 subsumes qp_3 based on the relation between the ontology concepts oc_1 and oc_3. Therefore during a case retrieval cycle, qp_1 will be asked before qp_3 since it is more generic. In a case c_i there will either be qp_1 or else qp_3, as per Rule 2 above, this provides for a reduction in the redundant questions being presented to the user during case retrieval.

The second step in the case-retrieval process is the matching step and in the original TCCBR this involves matching each qapair element in QP with the qapairs in each of the candidate cases. A ranked list of cases is established based on the following similarity measure $sim(qp_i, p_j)$:

$$sim(qp_i, p_j) = \begin{cases} 1 & \text{if } p_j \subseteq qp_i \\ (n+1-m)/(n+1+m) & \text{if } qp_i \subseteq p_j \\ 0 & \text{otherwise} \end{cases}$$

where,

qp_i is the question-answer pair in the user's query and

p_j is the question-answer pair in a candidate case

n = number of edges between qp_i and the root of the taxonomy

m = number of edges between qp_i and p_j

Having calculated such similarity between qapairs then an aggregate similarity metric is used to calculate the overall similarity between the user query QP and a case problem description, P_k. This aggregate similarity is calculated as follows:

$$Sim\ (QP, P_k) = \frac{\sum_{i \in QP, j \in P_k} sim(qp_i, p_j)}{T}$$

where, T in the original taxonomic theory represents the number of taxonomies, here it represents the number of disjoint branches in the domain ontology, that are associated with the qapairs. Cases are then ranked in a descending order based on this aggregate value.

In PreDiCtS we adopt the same similarity metric except that this similarity is computed on the qapairs' associated concepts rather then on the qapairs themselves. With reference to Figure 8 above, suppose that in the user's problem definition there are two qapairs, qp_1 and qp_4, while in the problem definition of the candidate case there are three, qp_3, qp_2 and qp_5.

Based on the associated concepts oc_1 and oc_3, the qapairs qp_1 and qp_3 are related by a parent-child relation, while the qapairs qp_4 and qp_2, which are associated with the concepts oc_7 and oc_5, are bound by a child-parent relation (see Table 2).

Table 2: qapair/Concept relations

User's Query		Case	
q-a pair	**concept**	**q-a pair**	**concept**
qp_1	oc_1	qp_3	oc_3
qp_4	oc_7	qp_2	oc_5
		qp_5	oc_2

The user's query though, does not contain a qapair in the candidate case that is related with qp_5. Using the adapted similarity metrics defined by TCCBR, we assume the following values:

$$sim\ (qp_1, qp_3) = sim\ (oc_1, oc_3) = 1$$

$$sim\ (qp_4, qp_2) = sim\ (oc_7, oc_5) = 0.5$$

for which the aggregate similarity will be

$$\sum sim\ (oc_i, oc_j) = \frac{1 + 0.5}{3} = 0.5$$

where the number of taxonomies here is 3 since:

a. the concepts oc_1 and oc_5 represent 2 disjoint branches in the ontology and

b. the concept oc_2 has to be considered as a separate disjoint branch.

The last phase of the case-retrieval takes the set of cases that are retrieved and rank-orders them in descending order based on the similarity score obtained from the previous step. Both the TCCBR and the PreDiCtS theories handle this step in a similar manner.

The conversation algorithm in PreDiCtS is the same as that defined by the TCCBR theory. The goal is to present the user with a ranked list of questions derived from the retrieved cases C_R. The process starts by considering all qapair taxonomies applicable to the cases in C_R. The score of each qapair in a taxonomy is based on the similarity scores obtained for the node-related cases. Each node takes the score of all related cases and the similarity of each parent node is the accumulation of the scores of its child nodes. A backward pass algorithm is used to calculate the score of each node. If the user problem definition contains a qapair from the taxonomy then the system selects its child nodes, else the most specific node that subsumes the set of retrieved cases is selected.

The question score is a tuple that includes <*taxonomy score, q-a pairs score*> as in Figure 9.

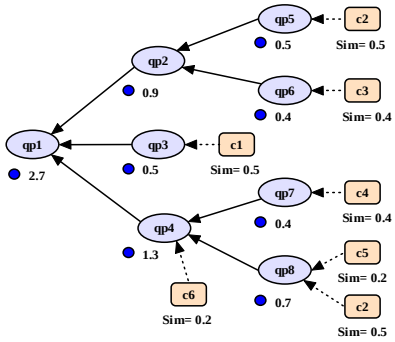

Figure 9:Question-Answer pair scoring

Example: $s(qp_4) = 1.3 = ((Sim_{c6} = 0.2) + (s(qp_7) = 0.4) + s(qp_8) = 0.7))$

4. EVALUATION

In this section we will present the main results of the tests that were carried out to evaluate the case retrieval performance in PreDiCtS (for a more detailed description see [5]). We considered a number of different requests to find suitable cases relevant to the travelling domain.

Table 3: Cases for the Travelling Domain

Case_ID	Solution	Problem
CID1	Conference + Hotel + Train	QID 5, QID6, QID9
CID2	Conference + Hotel + Airplane + Train	QID3, QID5, QID6, QID9
CID3	Conference + Hotel + Airplane	QID3, QID6, QID9
CID4	Conference + Hostel + Train	QID5, QID8, QID9
CID5	Conference + Hostel + Airplane + Train	QID3, QID5, QID8, QID9
CID6	Conference + Hostel + Airplane	QID3, QID8, QID9
CID7	Conference + Hostel + Airplane + Hotel + Train	QID3, QID5, QID6, QID8, QID9
CID8	Conference + Hotel	QID6, QID9
CID9	Conference + Hostel	QID8, QID9

A list of relevant cases (see Table 3) was developed, where each case had a unique CID (case ID). The problem definition consisted of sets of qapairs' ID references where each QID was also unique for the particular domain. The solution represented a set of services that were stored in a service template.

Four problems were identified and the performance of both the *Default CCBR* and the *TCCBR* approaches was considered. The identified problems were:

i. *The person (requester) wants to register for a conference, travel by train and stay in a hotel.*

ii. *The person (requester) wants to attend for a conference, requires accommodation and transportation.*

iii. *The person (requester) wants to travel by airplane for a conference and requires accommodation.*

iv. *The person (requester) wants to attend for a conference, has to travel by airplane and train and stay at a hotel.*

The main differences between the two approaches were in:

1. the number of questions that were presented to the requester by the conversation-generation algorithm, at the end of each retrieval cycle.
2. the accuracy of the similarity values.
3. the effect of leading the requester towards the Most Suitable Case (*MSC*).

The reason behind the first result is attributed to the fact that TCCBR considers the abstract relations between qapairs and thus limits the number of redundant questions to present to the user at each stage. This though, does not come without an initial effort on behalf of the case base designer. Infact time has to be dedicated to create suitable qapairs taxonomies that reflect a particular problem domain and then to associate these to the appropriate cases.

The second and third results both depend on the first one. The former is due to the fact that redundant qapairs are not considered in the case-similarity computation and this gives a more accurate figure at the end. The latter is the end result of the taxonomic aspect when designing the qapairs set. Infact this leading effect to solution-finding in more pronounced in the TCCBR then in the Default CCBR. Though again this is highly dependent on the design aspect of the case base.

5. FUTURE WORK

In this section we will consider how this work can be extended and improved by the inclusion of an adaptation component and feedback mechanism.

5.1 Adaptation

The addition of this component to PreDiCtS completes the CCBR cycle. Adaptation is closely related to the retrieval process, since it can be considered as the personalisation of a retrieved case to suit more effectively the requester's needs.

The main issue that has to be considered is the decomposition of a case into its basic components and then the ability to adapt each component separately. An adapted case will then be added to the case base and tested to ascertain its effect on the case base, after which it might be re-adapted or retained in the case base.

The adaptation of cases can be considered as a personalisation process through which the requester or designer can change aspects of a case to provide a more suitable problem-solution relation.

The possible changes can include:

- the addition and removal of services from the template definition (or solution)
- editing of input/output for particular services, including changes to associated ontological concepts
- changes in the order of execution of services or changes to the control constructs used
- adding or removing qapairs that make up the problem definition. Editing qapairs is a more complex process and will surely have a negative effect on the case base. So great care has to be taken in this case.

The most important aspect of this adaptation component will be the UI that provides the visualisation of all subcomponents of the case requiring adaptation. A mapping from the template definition back to UML activity diagrams is required. This will provide an easy way to adapt the solution. This component will be accessed also from within the retrieval component, where it would be possible to adapt the *MSC* to obtain a higher similarity to the problem at hand.

5.2 Feedback Mechanism

In Section 3 we have mentioned the importance of having a feedback mechanism included in PreDiCtS. It is a process that strives to maximise the usefulness of the case base.

We have identified that such a mechanism can help:

- the requesters to identify how a case has been used, by whom (here we refer to a process which clusters users not the actual person) and whether it has been reputed useful.
- the designer when importing cases from third-parties, since, based on the reputation of cases, then he can decide whether to import or not further cases from this source.
- the designer when performing maintenance on the case base; those cases that have a negative reputation may be removed from the case base.

Such reputation mechanism may be based on user feedback such as that discussed in [27] and in [15]. It is similar to the feedback mechanism of recommendation systems and considers the overall feedback given by case users to generate a reputation score. This score represents the usefulness of a particular case to solve a specific problem. For this reason we have included in CCBROnto a way to structure this information as part of the case history. We have also identified the need to compute a trust value, for the case creator. This trust value will be based on the global reputations of the cases provided by that particular case supplier. If the overall case-reputation is less then a certain threshold then this implies a lower trust level and that source will not be used any more. In a way this is similar to how eBay [12] reputes its sellers and buyers, though PreDiCtS will take action and remove this source from the list of possible case-suppliers.

6. CONCLUSION

In this paper we presented the motivation behind PreDiCtS. The use of the underlying CCBR technique as a pre-process to the service discovery and composition is promising since it provides for inherent personalisation of the service request and thus as a consequence also more personalised compositions. The tests that were performed showed that the design of both the case base and qapairs affects the retrieval process and to some extend, this also depended on the similarity measure. The most important difference between these two similarity measures was infact the number of relevant questions that the taxonomic similarity measure presented vis-à-vis the frequency based similarity measure during the conversation.

It will be interesting to see how PreDiCtS will continue to develop. Meanwhile we hope that the work presented in this paper provides an initial step towards the adoption of such mixed-initiative processes in the personalisation of the discovery and composition of Web services.

7. REFERENCES

[1] Aamodt, A., Gu, M., Tong, X., Component retrieval using conversational case-based reasoning. Proceedings of the ICIIP 2004, International Conference on Intelligent Information Systems. Beijing, China, October 21 - 23, 2004

[2] Aamodt, A., Gu, M., A Knowledge-Intensive Method for Conversational CBR, Proc. ICCBR'05, Chicago, August 2005

[3] Abela, C., Montebello, M., PreDiCtS: A Personalised Service Discovery and Composition Framework, in proceedings of the Semantic Web Personalisation Workshop, SWP 06, Budva Montenegro, 11th-14th June 2006

[4] Abela, C., Montebello, M., CCBR Ontology for Reusable Service Templates, in proceedings of the Demos and Posters session of the 3rd European Semantic Web Conference (ESWC 2006), Budva, Montenegro, 11th - 14th June, 2006

[5] Abela, C., Personalised Service Discovery and Composition Based on Conversational Case-Based Reasoning, MSc thesis, Department of Computer Science and AI, University of Malta, September 2006.

[6] Aha, D.W., Breslow, L.A., Muñoz-Avila, H., Conversational case-based reasoning, Applied Intelligence, 14, 9-32. (2001).

[7] Bernstein, A., Kaufmann, E., Kiefer, C., Bürki, C., SimPack: A Generic Java Library for Similarity Measures in Ontologies, University of Zurich, Department of Informatics, August 2005

[8] Buckle, M., Abela, C., Montebello, M., A BPEL Engine and Editor for the .NET framework, , accepted at the ECOWS 2005 conference, Växjö Sweden, November 2005

[9] CCBROnto, http://www.semantech.org/ontologies/CCBR-Onto.owl

[10] Creek, http://creek.idi.ntnu.no/

[11] Deelman, E., Gil, Y., et al, Mapping Abstract Complex Workflows onto Grid Environments**,** Journal of Grid Computing, Vol. 1, No. 1, pp 9--23, 2003

[12] eBay, http://www.ebay.com

[13] Goderis, A., et al, Seven bottlenecks to workflow reuse and repurposing, 4th Int. Semantic Web Conf., Galway, Ireland, 6-10 Nov. 2005

[14] Gupta, K., Taxonomic Conversational Case-Based Reasoning, Proceedings of the 4th International Conference on Case-Based Reasoning, 2001

[15] Hefke, M., A Framework for the successful Introduction of KM using CBR and the Semantic Web Technologies, I-Know 2004

[16] IUCBRF, Indiana University Case-Based Reasoning Framework http://www.cs.indiana.edu/~sbog-aert/CBR/

[17] Kim, J., Gil, Y., Towards Interactive Composition of Semantic Web Services, In AAAI Spring Symposium on Semantic Web Services, Palo Alto, California, USA, 2004

[18] Leake, D., Aktas M.S., Pierce M., Fox, G.C., A Web based Conversational Case-Based Recommender System for Ontology aided Metadata Discovery. Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04),Pages: 69 - 75

[19] OWL-S, http://www.daml.org/services/owl-s/1.1/

[20] Peer, J., A POP-based Replanning Agent for Automatic Web Service Composition, Second European Semantic Web Conference (ESWC'05), 2005

[21] Rajasekaran, P., et al, Enhancing Web services description and discovery to facilitate composition, First International Workshop, SWSWPC, July 2004

[22] Scicluna, J., Abela, C., Montebello, M., Visual Modelling of OWL-S Services, IADIS International Conference WWW/Internet, Madrid Spain, October 2004

[23] ServoGrid project, http://www.servogrid.org/

[24] Sirin, E., Parsia, B., et al, Filtering and selecting semantic web services with interactive composition techniques, IEEE Intelligent Systems, 19(4): 42-49,2004

[25] Sirin, E., et al, Planning for web service composition using SHOP2, Journal of Web Semantics, 1(4):377-396, 2004

[26] Weber, B., Integration of Workflow Management and Case-Based Reasoning, Supporting Business Processes through an Adaptive Workflow Management System, PhD thesis, University of Innsbruck, 2003

[27] Weber, B., et al, CCBR-Driven Business Process Evolution, Proc. 6th Int. Conf. on Case-Based Reasoning (ICCBR'05), Chicago, August 2005

Learning with Distance

John Abela
Department of Computer Science
Faculty of Science
University of Malta
jabel@cs.um.edu.mt

ABSTRACT

The two main, competing, paradigms in Artificial Intelligence are the *numeric* (vector-space) and the *symbolic* approaches. The debate on which approach is the best for modelling intelligence has been called the 'central debate in AI'. ETS is an inductive learning model that unifies these two, competing, approaches to learning. ETS uses a distance function to define a class and also uses distance to direct the learning process. An ETS algorithm is applied to the Monk's Problems, a set of problems designed to evaluate the performance of modern learning algorithms - whether numeric and symbolic.

Keywords

Machine Learning, ETS, Monk's Problems

1. INTRODUCTION

Evolving Transformation System (ETS) is a new inductive learning model proposed by Goldfarb [3]. The main objective behind this learning model was the unification of the two major directions being pursued in Artificial Intelligence (AI), i.e. the *numeric* (or vector-space) and *symbolic* approaches. In Pattern Recognition (PR), analogously, the two main areas are the *decision-theoretic* and *syntactic/structural* approaches [2]. The debate on which of the two is the best approach to model intelligence has been going on for decades - in fact, it has been called the 'Central Debate' in AI [7]. It was McCulloch and Pitts who proposed simple neural models that manifested adaptive behaviour. Not much later, Newell and Simon proposed the *physical symbol systems* paradigm as a framework for developing intelligent agents. These two approaches more-or-less competed until Minsky and Papert published their now famous critique of the perceptron, exposing its limitations. This shifted attention, and perhaps more importantly funding, towards the symbolic approach until the 1980s when the discovery and the development of the *Error Back Propagation* algorithm together with the work of Rumelhart *et al* reignited interest in the connectionist approach.

One of the main ideas in the ETS model is that the concept of distance plays an important, even critical, role in the definition, specification, and learning of the class. This paper presents the results of applying an ETS learning algorithm to the task of using distance to learn the classes in the *Monk's Problems* [8], a set of problems designed for testing modern learning algorithms. The experiments on the Monk's Problems were carried out as part of the author's Ph.D. programme. A full exposition can be found in the author's Ph.D. thesis [1]. Unless otherwise stated, the definitions, tables and diagrams of this paper are reproduced from this work.

2. THE ETS MODEL

The main idea that characterizes the ETS model is that of using distance, i.e. metric or pre-metric function, for defining a class. Given a domain of discourse O, a class C in this domain can be specified by a non-empty finite subset of C which is called the set of *attractors*, and which we denote by A, and also by a distance function d_C. The set of all objects in O that belong to C is then defined to be:

$$\{o \in O \mid d_C(a, o) < \delta,\, a \in A\}.$$

In other words, the class consists precisely of those objects that are a distance of δ or less from some attractor. We illustrate with a simple example. Suppose we want to describe (i.e. specify) the class (or concept) *Cat*. Let O be the set of all animals, C a finite set of (prototypical) cats, δ as non-negative real number, and d_{Cat} a distance function defined on the set of all animals. Provided that C, δ, and d_{Cat} are chosen appropriately, the set of all cats is then taken to be the set of all animals that are a distance of δ or less from any of the attractors, i.e. the set of prototypical cats. This is depicted below in Figure 1. Here the set C contains just one prototypical cat. All animals that are in the δ-neighbourhood of this cat are classified at cats.

This idea borrows somewhat from the theory of concepts and categories in psychology. The reader is also referred to [6] for a discussion of Eleanor Rosch's theory of concept learning known as *Exemplar Theory*. Objects are classified together if they are, in some way, *similar*. In our example, all the animals that are cats are grouped together since the distance between any cat and the prototype is less than the threshold δ. In other words, an animal is a cat if it is similar

CSAW 2006 Dec 5-6, Kalkara, Malta

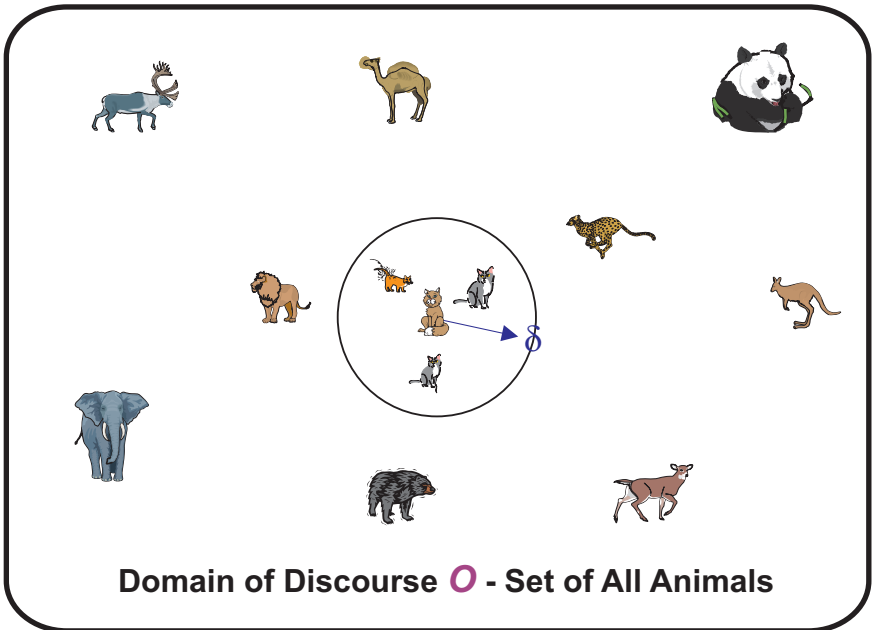

Figure 1: Class Description in the ETS Model.

to the cat prototype. The less distance there is between two animals, the more similar they are - i.e. distance is a measure of dissimilarity.

The ETS model is not just about class description, but also about learning class descriptions of classes from finite samples to obtain an *inductive class description*. Let O be a domain of discourse and let $\mathcal{C}$ be a, possibly infinite, set of related classes in O. Let C be a class in $\mathcal{C}$ 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 $\mathcal{C}$ is infinite, then C^+ may be a subset of infinitely many classes in $\mathcal{C}$. 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 $C' \in \mathcal{C}$ such that C' *approximates* C. This depends, of course, on having a satisfactory definition of what it means for a class to approximate another.

In essence, learning in the ETS model involves finding a distance function that achieves *class separation*, i.e. a distance function such that the distance between objects in C^+ is zero or close to zero while the distance between an object in C^+ and an object in C^- is appropriately greater than zero.

An ETS algorithm achieves this by iteratively modifying a (parametrized) distance function such that the objects in C^+ start moving towards each other while, at the same time, ensuring that the distance from any object in C^+ to any object in C^- is always greater than some given threshold.

3. KERNEL LANGUAGES

Kernel Languages is an interesting subclass of regular languages. A kernel language over a finite alphabet Σ is specified by the pair $\langle K, F\rangle$ where $K \subset \Sigma^*$ is a finite, non-empty, set of strings called the set of *kernels* and $F \subset \Sigma^+$ is a finite, non-empty, and factor-free set of strings called the set of *features*. Informally, the strings in the kernel language specified by $\langle K, F\rangle$ are precisely those strings that can be obtained (generated) by inserting features from F anywhere, in any order, and any number of times, into the kernel strings of K. We only require that;

1. features are not inserted inside other features,
2. no feature is a factor[1] of any other feature, i.e. F is factor-free, and
3. no kernel contains a feature as a factor.

We illustrate with an example. Consider the set of kernels $K = \{bb, bc\}$ and the set of features $F = \{ab, ba\}$. The following strings in $L\langle K, F\rangle$, the language generated from K and F, are obtained by successive insertions of features in the kernels: (kernels letters are shown in bold)

bb	***bc***
*ba**bb***	***bc**ab*
*ba**bb**ab*	*ab**bc**ab*
*ba**b**ab**b**ab*	*ab**b**ba**c**ab*
*ba**b**abba**b**ab*	*ab**b**baba**c**ab*
*abba**b**abba**b**ab*	*baab**b**baba**c**ab*
*abba**b**abba**b**abab*	*baab**b**baba**c**abba*

[1]i.e. a substring

Note that features can be inserted anywhere in a kernel but not inside another feature. We must point out, however, that this does not necessarily mean that a string in $L\langle K, F\rangle$ cannot contain factors such as $aabb$ which can be formed by the insertion of the feature ab inside another occurrence of the same feature. The reason for this is because this feature can also be formed from the features ab and ba as follows: $\boldsymbol{bb} \rightarrow_{ab} ab\ \boldsymbol{bb} \rightarrow_{ab} baab\ \boldsymbol{bb}$ to obtain the string $baabbb$ which, of course, contains $aabb$ as a factor. Testing for membership in a kernel language is achieved either by checking if a given unknown string x can be generated from one of the kernels by a sequence of feature insertions or, alternatively, by (nondeterministically) deleting features from x to obtain a kernel. Note that the latter procedure is equivalent to computing the normal forms of x modulo the special semi-Thue system, R_F, that consists exactly of $|F|$ rules of the form (f, ε), $f \in F$. The set of rewrite rules of R_F is therefore indexed by F. To determine if x belongs to $L\langle K, F\rangle$ we then need only check whether one of the normal forms belongs to K.

There are various types of kernel languages. These include confluent and non-confluent kernel languages, trivial kernel languages, non-congruential kernel languages, and kernel languages with single or multiple kernels. It turns out that kernel languages have a number of real-world applications. The Monk's Problems instances can be encoded as strings from a kernel language. So can the parity problem and other interesting real-world problems [1].

4. THE MONK'S PROBLEMS

In the summer of 1991 at the 2^{nd} European Summer School on Machine Learning held at the Corsendonk Priory in Belgium, a number or researchers proposed a set of problems for testing the various machine learning algorithms that existed at the time. This set of problems was called 'The Monk's Problems' . The idea was that the main machine learning algorithms would be tested and compared using the same dataset.

The Monk's Problems are set in an artificial robot world where a robot can be described by six different attributes as follows (see Figure 2):

x_1 : head_shape	$\in$	{round, square, octagon}
x_2: body_shape	$\in$	{round, square, octagon}
x_3 : is_smiling	$\in$	{yes, no}
x_4: holding	$\in$	{sword, balloon, flag}
x_5: jacket_colour	$\in$	{red, yellow, green, blue}
x_6: has_tie	$\in$	{yes, no}

There were three problems in the set. Each problem was a binary classification task, i.e. that of determining whether or not a robot belongs to one of three classes. Each problem consists of a description of the class and a training set that is a proper subset of the 432 possible robots in the artificial world. The task of the machine learning algorithm is to generalize from the training examples and, if possible, to output a class description of each of the three classes. The three classes were:

1. **Monk1**:
   (head_shape = body_shape) or (jacket_colour = red)
   124 labelled examples were randomly selected from 432 possible robots. No misclassifications.

2. **Monk2**:
   exactly two of the six attributes have their *first* value
   169 labelled examples were randomly selected from 432 possible robots. No misclassifications.

3. **Monk3**:
   (jacket_colour is green and holding sword) or (jacket_colour is not blue and body_shape is not octagon)
   122 labelled examples were randomly selected from 432 possible robots. 5% misclassifications.

The only problem that contained noise was Problem 3. The intention here was to test the performance of the algorithms in the presence of noise. Problem 2 is very similar to parity problems. Problems 1 and 3 are in DNF form and are therefore assumed to be solvable by symbolic learning algorithms such as ID3, AQ, etc. Problem 2 combines attributes in a way which makes it awkward to express in DNF or CNF.

Table 1, reproduced from [8], lists the results obtained by the different learning algorithms on the Monk's Problems datasets. The experiments were performed by leading researchers in Machine Learning, each of whom was an advocate of the algorithm he or she tested and, in many cases, the creator of the algorithm itself.

The algorithms tested included the various decision tree learning algorithms such as ID3 and its variations, mFOIL - a rather interesting inductive learning system that learns Horn clauses using a beam search technique, and various neural networks such as Backpropagation and Cascade Correlation. No algorithm managed to correctly learn all three classes, although some came very close. In spite of the fact that the Monk's Problems are defined in a symbolic rather than a numeric domain, the best performing algorithms were, perhaps surprisingly, the neural networks.

5. THE *VALLETTA* ALGORITHM

The main objective of the *Valletta* algorithm is to investigate the feasibility or otherwise of applying the ETS model to a grammatical inference problem. The aim is to see if and how distance could be used to direct the learning process and also how such an algorithm would perform in the presence of noise. Valletta's learning strategy is based on the observation that the set of features that partially specify an unknown kernel language K must necessarily be a subset of the set of all repeated substrings in C^+ - assuming, of course, that the strings in C^+ were drawn at random from K and that every feature occurs at least twice in C^+. Valletta was designed from the beginning to learn multiple kernel languages and not just single kernel languages. This is because all examples of naturally occurring, i.e. real-world, kernel languages that we came across were all multiple-kernel. It turns out that learning multiple-kernel languages is much more difficult than learning single kernel languages. With

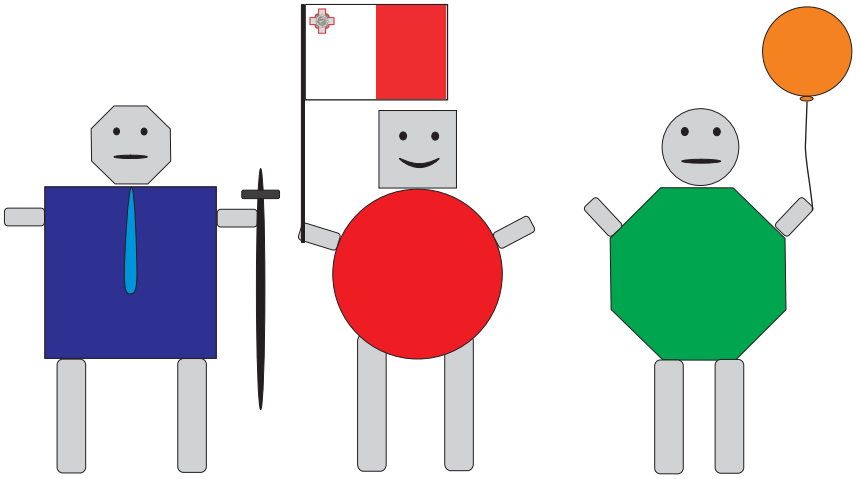

Figure 2: Some of the robots in the Monk's Problems.

Learning Algorithm	#1	#2	#3
AQ17-DCI	**100%**	**100%**	94.2%
AQ17-HCI	**100%**	93.1%	**100%**
AQ17-FCLS		92.6%	97.2%
AQ17-NT			**100%**
AQ17-GA	**100%**	86.8%	**100%**
Assistant Professional	**100%**	81.3%	**100%**
mFoil	**100%**	69.2%	**100%**
ID5R	81.7%	61.8%	
IDL	97.2%	66.2%	
ID5R-Hat	90.3%	65.7%	
TDIDT	75.7%	66.7%	
ID3	98.6%	67.9%	94.4%
ID3, no windowing	83.2%	69.1%	95.6%
ID5R	79.7%	69.2%	95.2%
AQR	95.9%	79.7%	87.0%
CN2	**100%**	69.0%	89.1%
CLASSWEB 0.10	71.8%	64.8%	80.8%
CLASSWEB 0.15	65.7%	61.6%	85.4%
CLASSWEB 0.20	63.0%	57.2%	75.2%
PRISM	86.3%	72.7%	90.3%
ECOWEB leaf prediction	71.8%	67.4%	68.2%
ECOWEB l.p. & information utility	82.7%	71.3%	68.0%
Backpropagation	**100%**	**100%**	93.1%
Backpropagation with weight decay	**100%**	**100%**	97.2%
Cascade Correlation	**100%**	**100%**	97.2%

Table 1: The published results of the Monk's Problems.

single-kernel languages one need only find a set of features. On the other hand, with multiple-kernel languages one also have to find the kernels without knowing beforehand the the number of kernels in the unknown language. Besides the obvious computational complexity this problem also poses an interesting question. Should one find a TS description that minimizes the number of features or the number of kernels? Valletta can be instructed to find TS descriptions that minimize either the number of features or the number of kernels. Valletta has what is called a *variable inductive preference bias*. This means that Valletta allows the user to choose which hypotheses (i.e. TS descriptions) are preferred over others. This is an important advantage over other learning algorithms.

Valletta has two main stages. The *pre-processing stage* searches for all repeated substrings in C^+ and stores them in a repeated substring list R_{C^+}. The *learning stage* then finds a set of features from R_{C^+} that gives class separation, i.e. a set of features that optimizes the function

$$f = \frac{f_1}{\epsilon + f_2},$$

where f_1 is the minimum EvD distance (over all pairs) between C^+ and C^-, f_2 is the average pair-wise *intra-set* EvD distance in C^+, and ϵ is a small positive real constant to avoid divide-by-zero errors. Valletta's learning stage builds a structure, the *search tree*, in which each node represents a feature set. Valletta expands this tree only on the basis of f_2. This means that Valletta's search for the set of features that describes the unknown kernel language K is completely directed by f_2. No other criteria are used to direct the learning process.

Valletta uses a new string-edit distance function called *Evoluntionary Distance* (*EvD*) [1]. EvD is suitable for describing kernel languages since it can detect features inserted inside other features. The idea behind EvD is that, given two strings and a set of features F, the distance between two strings can be taken to be the weighted Levensthein distance (WLD) between the normal forms (modulo R_F) of the two strings [1]. One important advantage of this technique is that normal forms are usually much shorter than the actual strings and this this results in significantly shorter computation times. The main problem is, of course, how to efficiently reduce the strings to their normal form modulo F. This was accomplished using a data structure called a *parse graph*. EvD works by first building the parse graphs for the two strings and then extracting the normal forms from the parse graphs. The EvD procedure then computes the weighted Levensthein distance between the normal forms and the set of kernels that is passed as a parameter.

An explanation of the inner workings of Valletta is beyond the scope of this paper. The reader is refered to [1] for a full exposition.

6. RESULTS AND CONCLUSIONS

The Monk's problems can quite easily be posed as GI [2] problems. In theory, every learning problem can be posed as a GI problem. In particular, each of the Monk's three classes of robots can be represented by a confluent kernel language. For the experiments, a special version of Valletta was developed. This is simply Valletta but with a much narrower inductive preference bias. The new version of Valletta was called Mdina, after Malta's old capital. Mdina considers only trivial kernel languages [1]. Mdina successfully learned problems 1 and 2 but did not learn problem 3. Investigation showed that this was because the training set was not *structurally complete* [4]. The addition of one string made the training set structurally complete and Mdina was then able to learn the class.

Mdina served to show that distance can indeed be used to direct the learning process. The experiments also highlighted the fact that learning algorithms converge to the correct class if the inductive bias of the algorithm is correct. In his technical report Thrun describes the Monk's problems and the results obtained by the various algorithms does not attempt to analyse or explain the results. We feel the whole exercise served more to determine whether each algorithm had the correct inductive bias to learn each of the Monk's problem than to determine the actual learning ability of the various algorithms. Each of the algorithms listed in the report have successfully been used for other learning tasks. We believe that the apparent inability of some of the algorithms to learn the Monk's problems is due more to their type of inductive bias rather than to anything else. Wolpert [10] and 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 [9]. 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 an incorrect inductive bias.

[2] Grammatical Inference

7. REFERENCES

[1] Abela, John. *ETS Learning of Kernel Languages.* Ph.D. Thesis, University of New Brunswick, Canada, 2002.

[2] Bunke, H. and Sanfeliu, A., (eds). *Syntactic and Structural Pattern Recognition - Theory and Applications.* World Scientific series in Computer Science, Vol. 7. 1990.

[3] Goldfarb, Lev. *On the Foundations of Intelligent Processes - 1: An Evolving Model for Pattern Learning.* Pattern Recognition, 23, pp. 595-616, 1990.

[4] Michalski, R., Carbonel, J., Mitchell, T., (eds). *Machine Learning - An Artificial Intelligence Approach.* Morgan Kaufmann Publishers Inc. 1983.

[5] Nigam, Sandeep, *Metric Model Based Generalization and The Generalization Capabilities of Connectionist Models.* Masters Thesis, Faculty of Computer Science, University of New Brunswick, Canada. 1992.

[6] Rosch, Eleanor, H., *On the Internal of Perceptual and Semantic Categories.* in Timothy E. Morre, ed., *Cognitive Development and the Acquisition of Language*, Academic Press. 1973.

[7] J. Stender, and T. Addis (eds). *Symbols vs Neurons.* IOS Press, Amsterdam, 1990.

[8] S. Thrun et al. *The Monk's Problems: A Performance Comparision of Different Learning Algorithms.* Carnegie Mellon University CMU-CS-91-197, December 1991.

[9] Thornton, Chris *There is No Free Lunch but the Starter is Cheap: Generalisation from First Principles* Cognitive and Computing Sciences, University of Sussex, Brighton, UK, 1999.

[10] Wolpert, D., and Macready, W. *No Free Lunch Theorems for Search*, Unpublished MS, 1995.

Performing Fusion of News Reports through the Construction of Conceptual Graphs

Joel Azzopardi
Department of Computer Science and AI
Faculty of Science
University of Malta
jazz018@um.edu.mt

ABSTRACT

As events occur around the world, different reports about them will be posted on various web portals. Different news agencies write their own report based on the information obtained by its reports on site or through its contacts – thus each report may have its own 'unique' information. A person interested in a particular event may read various reports about that event from different sources to get all the available information. In our research, we are attempting to fuse all the different pieces of information found in the different reports about the same event into one report – thus providing the user with one document where he/she can find all the information related to the event in question. We attempt to do this by constructing conceptual graph representations of the different news reports, and then merging those graphs together. To evaluate our system, we are building an operational system which will display on a web portal fused reports on events which are currently in the news. Web users can then grade the system on its effectiveness.

1. INTRODUCTION

If one browses the news sites on the World Wide Web (WWW) such as Reuters [1], or AFP [2], he can find news reports of events which have happened only minutes before the time of reading. Each event that occurs can be found reported by different authors on different news sites. The different reports on the same event usually contain the same back-bone of the main sub-events but different fine details. Therefore, for a user to get the whole picture of that event with all the different details incorporated with it, he/she would have to read various reports on the same event. Unfortunately, most of the information in each report would be present in the other reports as well. Thus to get all the fine details, the user has to endure reading repeated information.

[1]http://today.reuters.com
[2]http://www.afp.com/english/news/?pid=stories

We attempt to address the above issues in our research by constructing logical representations of the different news reports, and then merging those structures which are representing reports on the same event. The merged structure would then contain all the information obtained from the different reports. This merged structure would then be presented back to the user as a 'fused' report for that event with unnecessary repetition of the same information.

The news reports are represented using structures similar to a conceptual graphs, and the construction of these conceptual graphs is performed using surface based approaches only – i.e. we will not be using approaches which require deep semantic analysis of the text or a knowledge base. In our opinion, the use of surface-based approaches is more appropriate to news reports since they are less computationally expensive to implement and execute, and also new names and terms crop up in the news every day – otherwise the knowledge base will have to be updated regularly. Moreover surface-based approaches will enhance the portability of our system across different domains since unlike knowledge bases, they can be more general and not limited to particular domains.

Within our research, we are also attempting to evaluate our approach in terms of helpfulness to the user.

The structure of the remaining part of this report will be as follows: in the next section, we describe related work done or currently being done in the same area as our research. Then in the following section, we will describe the methodology implemented in our system. A description of our intended method of evaluation will follow in the proceeding section, and in the final section we give our conclusions and our plans for future work.

2. LITERATURE REVIEW

The aim of Document Fusion is to produce a fused report which will contain all the relevant information from the different source documents without repetition ([7], [1], [8]). The reasons behind the need for Document Fusion are various, namely:

- Any document/report from any particular source is never fully objective. On the other hand, a document built from multiple sources produces more objectivity ([14]),

- Reports from different sources on the same event may contain different information – in fact reports from different sources may agree with each other, contradict each other or add new information ([8], [7], [1]).
- Information Fusion may also help in tracking the developments to reports over time ([8], [14]).

The steps involved in document fusion are ([8], [1], [7], [13]):

1. The segmentation of each document into different segments,
2. The logical representation of each segment so that each segment may be compared to segments from other documents,
3. The construction of the 'fused' document.

Information Fusion is most commonly applied to textual news reports ([8], [1], [7]). However, we also encountered examples of the application Information Fusion on video news reports ([14]), and also on search engine result listings ([13]).

Information Fusion involves the segmentation of the different documents and then building relationships between segments from different documents. In its definition of the Cross-Document Structure [8] describes relationships which may occur between *paragraphs*, *sentences*, *phrases* and even *individual words*.

These different levels of granularity present different issues. [7] goes to the coarse side of the spectrum and segments the documents into paragraphs. According to [7], the use of paragraph segments simplifies matters since paragraphs are context-independent and hence fused reports built from paragraphs taken from different sources will be more readable. Furthermore, [7] argues that within the context of news reports, paragraph units are not too coarse since the paragraphs within news reports do not usually contain more than 3 sentences.

In direct contrast to the use of paragraph segments in [7], [1] claims that even sentence segments are too coarse and each sentence may contain more than one theme. Thus the construction of the fused report from sentence units may lead to repetition. [1] recommends the break-down of sentences into phrases – this eliminates the repetition, but then brings forward the need to do sentence re-generation to make the fused report readable.

In our opinion, performing fusion with paragraph segments will need relatively less processing than fusion with finer granularities. However, paragraphs are too coarse for fusion, and inevitably if fusion is to be made from paragraph segments repetition will inevitably occur.

On the other hand, breaking down the sentences into phrases will necessitate the need for sentence generation after fusion has been done to ensure readability of the output document.

We attempt to do fusion using sentence-size segments. To avoid the problem of repetition in the case of sentences containing more than one themes, we will give a priority to the shorter sentences to be used in the final 'fused' document since these are the most likely to contain only one theme.

To build the relationships between the segments, the segments are represented using a graph representation ([8], [1]) unless the segments are coarse as in the case of [7]. A particular representation which is of interest to us is the *DSYNT* structure which is described in [1]. The *DSYNT* is a graph structure whereby a node is built for each phrase, and a phrase consists of a verb with 2 nouns.

A representation of concepts and relationships between concepts which is quite similar to the *DSYNT* structures described previously but is more evolved is the *Conceptual Graphs* Representation. Conceptual Graph representation is a representation which is precise but also human readable ([10]). It consists of concept nodes which represent entities, states, attributes and events, and relation nodes which describe the relationships between the different concepts ([10], [4]).

In our research, the ideal representation would be *Conceptual Graphs*. However, the construction of conceptual graphs from raw text would require semantic information and the support of a knowledge base as described in [11]. Since we try to adhere to surface-based approaches, our aim is to build structures resembling conceptual graphs as much possible but using surface-based approaches only.

In the construction of graph representation for text, the concepts are first extracted from the text to form the graph nodes, and then the relationships between the concepts are built ([6], [3], [9], [3]). Concepts can be defined by certain pre-defined phrases or by the extraction of proper names ([6], [3]), or otherwise they can be extracted by the use of heuristic rules ([9], [3]).

In our research, we have used a combination of both approaches listed above used for concept extraction. A Part-Of-Speech tagger is used to tag the text, and then heuristic rules are used to extract the concepts from the text. Over and above that, simple extraction of proper names is also done, and any proper names which have not been inserted as concepts in the previous step, will be inserted now.

We have encountered various approaches to the construction of relationships between different concepts. On one hand, there are approaches which build relationships based on the semantic meaning of the concepts and the words in the text ([6], [11], [12]). A case of interest is that described in [11]), whose system uses a lexicon of word canonical graphs to build all the possible graphs for each sentence. Then, a semantic knowledge base is used to reject those sentence graphs which do not make sense semantically. [12] does not build only relationships according to semantic meanings, but also builds relationships between those words or phrases which have high frequencies of co-occurrence within text windows of pre-defined size.

On the other hand, we have more surface-level approaches which make use of heuristic rules applied on the Part-Of-Speech tags (produced by a parser) corresponding to the different words. Examples include [9] which build *Noun-*

Verb-Noun tuples, and [1] which builds the *DSYNT* structure described previously.

Another surface-level approach which is quite different to those described above is that described in [3]. In this case, there are two type of constructed relationships between concepts. These are:

- **Named Relations** – each such relationship consists of 2 concepts and a relationship name. Such relationships are extracted using heuristic patterns – e.g. the text '*president of*' if used to extract a named relationship between '*George Bush*' and '*United States*' from the following sentence: '*George Bush, president of the United States*'.
- **Unnamed Relations** – each such relationship consists of two concepts and a relationship strength. These relationships are built by finding sets of concepts which have a high co-occurrence rate within the same sentences.

In our research, we are trying to use surface-based approaches as much as possible. Therefore, the use of knowledge bases to construct relationships between concepts would go against our approach and also would prove costly, and probably also limit the domain in which our system can operate. We think that the construction of *DSYNT* structure or *NVN* tuples is quite straightforward since the text would have already been tagged previously for the concept extraction. On the other hand, the approaches described in [3] are very interesting and can be applied as well.

3. METHODOLOGY

Our system performs the construction of fused news reports in the following steps:

1. The downloading of the news reports from different sources via RSS feeds.
2. The clustering the documents according to their content,
3. The construction of conceptual representations for each document,
4. The merging of conceptual representations for documents within the same cluster,
5. The construction of the fused document.

As we mentioned in the previous section, one of the main issues in information fusion is to avoid repetition. In our case, this is handled during the merging of the conceptual structures used for representations.

3.1 Downloading of News Reports

Within this part, the RSS feeds from a number of pre-defined sources are downloaded. For each downloaded RSS record, the system first checks if the corresponding news report has been downloaded already. If it has not yet been downloaded, the news report will be downloaded, filtered from the surrounding HTML code, and stored within an XML file together with the other details specified in the RSS record.

3.2 Document Clustering

The main 'problem' within the task of document clustering is that the number of final document clusters is not known beforehand.

To tackle this problem, we adapted a technique similar to that described in [5]. We first index the documents, remove the stop-words from the indexes and assign term weights to the index terms using *tf-idf* measure. Then the similarity of that documents with each cluster is calculated and if there exists a cluster with a similarity higher than a pre-defined threshold, that document is placed within that cluster.

3.3 Building Conceptual Representations for each Document

The logical representation built within this section is an entity-relation structure whereby we define the entities (the 'objects' within the document) and the relations / actions occurring between these entities. Noun entities form the entities, and verb entities describe the relations between these entities.

The steps followed by the system to construct the conceptual graph are as follows:

1. The contents of the document in question are read and are tagged using a POS Tagger.
2. The noun entities are extracted from the document's contents.
3. The complex noun entities within the document i.e. entities formed from 2 or more 'simple' noun entities are extracted.
4. The verb entities are extracted. These will be the names of the named relations between 1 or 2 entities.
5. The relationships between the noun entities (concepts) are built.
6. The relations are grouped into series of relations which lead from one to other.
7. Co-referring noun entities are grouped together.

The following sub-sections contain a more detailed description of each step.

3.3.1 Tagging the Document's Contents

The document's contents are tagged using Brill's Part of Speech tagger [2].

After Brill's Part of Speech tagger has been employed to tag the text, some corrections are applied to the resulting tagged text. Our system reads a text file which contains a list of tokens each with its own corresponding tag and assigns all occurrences of these tokens within the document's text to that tag.

Finally, the named entities are identified and the tags of the tokens which make up these names are set to be of type 'PROPER NOUN'. Named entity extraction is performed

by identifying those tokens which start with an upper-case letter but do not occur at the start of a sentence. Once those tokens have been identified, the tokens which occur at the start of sentences are matched with the list of names extracted previously, to check whether they also form part of names.

3.3.2 Extracting the Noun Entities

Noun entity extraction is performed by applying the following rules to the tagged text:

```
<Noun-Entity> = [<Determiner>] (<Adjective>)* (<Noun>)
```

In other words, a Noun Entity must consists of at least one noun token, any number of adjective tokens preceding that noun token, and possibly a Determiner at the start of the noun entity object.

In cases where a sequence of adjective tokens is found without a noun token at the end, the last adjective in the sequence is considered to be a noun token. Occurrences of such sequences are due to erroneous tagging of text by the POS tagger.

3.3.3 Building the Complex Noun Entities

Complex noun entities are composed of two or more 'simple' noun entities which are linked together by a preposition, or a conjunction or disjunction. For example if we have the phrase 'the president of Iraq'; within the step described in section 3.2, we identified 'the president' and 'Iraq' as separate noun entities. However, it is quite obvious that 'the president of Iraq' is referring to a single entity. By generalization of this example, we can therefore build complex noun entities using the rule:

```
<Complex-Noun-Entity> = <Simple Noun Entity>
                        <preposition>
                        <Simple Noun Entity>
```

Another feature of complex noun entities, as used by our system, is their composite nature – i.e. a complex noun entity may be built using other complex noun entity objects. In fact, the actual heuristic rule which we used in building complex noun entity objects is as follows:

```
<Complex-Noun-Entity> = <Simple Noun Entity>
                        <preposition>
      (<Simple Noun Entity> | <Complex Noun Entity>)
```

Note from the above rule that we assume complex noun entities to be right-associative. This is illustrated in the following example.

Imagine that we have the phrase: '*The leader of the terrorist organization in Iraq*'. The simple noun entities within this phrase are: '*The leader*', '*the terrorist organization*' and '*Iraq*'. According to the heuristic rule we used to build the complex noun entities, we first build the complex noun entity shown in Figure 1.

Then we build the final complex noun entity shown in Figure 2.

3.3.4 Extracting the Verb Entities

Verb Entities are used as names for the relationships between noun entities. These are extracting by applying the

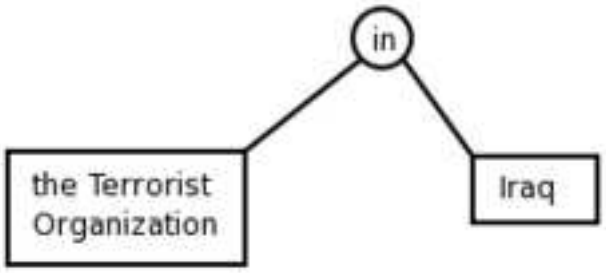

Figure 1: Representation of '*the Terrorist Organization in Iraq*'

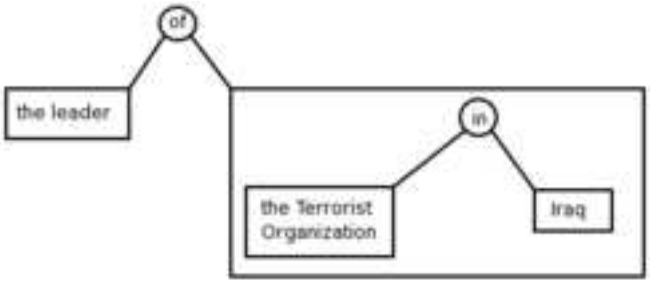

Figure 2: Representation of '*the leader of the Terrorist Organization in Iraq*'

following rule to the tagged text:

```
<Verb-Entity> = (<Adverb>)* [<Auxiliary>] (<Adverb>)*
(<Verb>)*
```

3.3.5 Building the Relationships between the Noun Entities (Concepts)

A Named Relation consists of a verb entity, which provides the name for the relation, and one or two noun entities, between which the relation is defined.

Named Relations may be subdivided into two groups, namely:

- **binary relation** – where the verb entity involved is transitive and the relation is defined between two noun entities. For example, in the phrase '*John kicked Mary*', we have the relation '*kicked*' between '*John*' and '*Mary*',
- **unary relation** – where there is only one noun entity object involved since the verb 'defining' the relation is intransitive. For example, in the phrase '*John died*', we have the 'relation' '*died*' and only the noun entity '*John*' is involved.

Within this sub-section, we are involved in the extraction of these types of entities. To perform this extraction, the system traverses the list of sentence phrases (whose creation is described is section 3.5), and builds the named relations by applying the following heuristic rules in the order given below:

1. `<Named Relation> = <Noun Entity>`

```
                        <Verb Entity>
                        <Noun Entity>

2.  <Named Relation> = <Noun Entity>
                        <Verb Entity>
                        <Phrase Delimiter>

3.  <Named Relation> = <Verb Entity>
                        <Noun Entity>
                        <Phrase Delimiter>
```

If during the construction of the Named Relations, the system finds a temporal entity (a date, or a time string) immediately before or after the entities forming the relations, it attaches this temporal entity to the relation being constructed as an indication of the time/date during which that relation was established.

3.3.6 Identifying Relations Leading from one to another

Within the same sentence phrase, the system may find more than one different named relations. If this is the case, and there is a noun entity which forms part of two relations, those relations are set to 'related' to each other – in the sense that one relation leads off from the other.

For example, consider the sentence, '*The attack hindered the work being done in the country.*' From this sentence, we may extract two named relations – namely '*The attack hindered the work*', and '*the work being done in the country*'. Since there is the noun entity '*the work*' being used in both relations, these two relations are set to be 'related' to one another.

3.3.7 Clustering those noun entities which are referring to the same object

Within a document, different noun entity objects refer to the same real-world object. For example, the noun entities '*the president of the United States*', '*George Bush*', '*the former governor of Texas*' are all referring to the same real-world object – namely George Bush, who is the president of the United States at the time of writing.

Our system attempts to cluster together those noun entities which are 'co-referent' so that we will have as much as possible a one-to-one relation of entities within the conceptual structure constructed to real-life objects. In this way, certain operations, such as the retrieval of all relations which concern a particular object, are greatly facilitated.

This clustering is done in two parts. In the first part, a set of 'significant' tokens is constructed for each noun entity object where 'significant' tokens within a noun entity are those tokens which are the actual nouns. Those noun entities which have equivalent sets of 'significant' tokens are clustered together as co-referent.

The second part of the noun entity clustering utilizes the list on un-named relations whose extraction was described in section 3.5. In this part, the system traverses the list of un-named relations, and for each pair of noun entities, it identifies the two noun entity clusters which contain each noun entity in question, and merges these two clusters together.

3.4 Merging Conceptual Graphs

In the previous section, we described the procedure we used to construct the conceptual graph representation for each document. Now, we need to identify those entities and relations which are common across the different documents of the same cluster.

This *merging* task may be sub-divided into two sub-tasks:

1. The clustering of those noun entity and verb entity objects which are 'synonyms',
2. The clustering of co-referring relations.

3.4.1 Clustering Noun-Entity and Verb-Entity Objects

A similar process is used to cluster co-referring noun-entity and verb-entity objects.

The tokens of each noun-entity and/or verb-entity object are weighted using the *tf-idf* measure based on the inverted index for the document cluster. Those tokens which have a *normalized* weight of 0.5 or greater are considered to be the '*defining*' tokens for that entity object.

When comparing two entity objects, an *intersect* list of defining tokens is extracted, as well as the *difference* list of defining tokens for those two entity objects. The members of the *intersect* list contribute to the equivalence score of those two tokens, whilst the members of the *difference* list hinder this equivalence score. The two entity objects are considered to be co-referring if their equivalence score exceeds a certain threshold.

3.4.2 Clustering the Relations

The approach to the relations' clustering is based on the premise that two relations are clustered if they share at least a *co-referring* verb entity and a *co-referring* noun entity between them, or two *co-referring* noun entities between them.

Two relations are considered to share a *co-referring* verb entity or a *co-referring* noun entity if one of the relations contains a verb (or a noun) entity which forms part of the same verb-entity (or noun-entity) cluster as a verb (or a noun) entity from the other relation.

3.5 Building the Fused Report

Once the previous step has been completed, we end up with a list of conceptual structures representing the concepts and the relations between them. Since these relations have been 'merged' together, we now have a list of 'unique' relations between concepts which represent the different information found in the different reports. The final step involved in the construction of the 'fused' report is to have the system select those sentences which contain the relations represented in conceptual form, and ensure that no relation will be represented in more than one sentence within the final 'fused' document.

4. EVALUATION

Since Document Fusion is a relatively unexplored field, we have not yet encountered any data corpus which provides sample fused documents for clusters of input documents. The information fusion systems described in [13] and [7] use human assessors to evaluate their results. On the other hand, [1] evaluates only similarities found across different documents – this is done by comparing the similar relationships extracted by their system with those extracted by human judges.

To evaluate our approach to Document fusion, we decided to build a News Document Fusion system which will be available on the WWW. This Document Fusion system uses RSS feeds from different sources to download news reports as they are published. The downloaded reports would then be clustered together according to the event they are reporting. Document Fusion would then be applied on the documents within each cluster and the 'fused' document produced will be presented to the user on the WWW site.

To evaluate the Document Fusion system, each fused report on the WWW site will contain links to the original reports, as well as a form where each user can rate that fused report based on the inclusion of all the unique information found in the source reports and the inclusion only once of all the repeating information.

5. CONCLUSION AND FUTURE WORK

Although we do not yet have evaluation results for our system, we are optimistic that our approach is a promising one. The fact that our system is implemented using surface-level approaches only greatly expands the domain across which our system can be operated.

Moreover the representations we use in our approach simplifies the adaptation of our system to other purposes like Topic Tracking and Information Filtering. In fact, in the foreseeable future, we intend to adapt our approach to Topic Tracking and Information Filtering. We intend also to incorporate User Modelling so that a user will be shown a 'fused' report which suits that particular user – i.e. this 'fused' report will contain very few (if any) details on the information that the user already knows, and concentrates on the 'new' details which the user is still to learn about.

6. REFERENCES

[1] R. Barzilay, K. R. McKeown, and M. Elhadad. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 550–557, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[2] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 152–155, Morristown, NJ, USA, 1992. Association for Computational Linguistics.

[3] R. Byrd and Y. Ravin. Identifying and extracting relations in text. In *NLDB 99 – 4th International Conference on Applications of Natural Language to Information Systems*, Klagenfurt, Austria, 1999.

[4] M. Chein and M.-L. Mugnier. Conceptual graphs: fundamental notions. In *Revue d'Intelligence Artificielle, Vol. 6, no. 4, 1992*, pages 365–406, 1992.

[5] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, New York, NY, USA, 1999. ACM Press.

[6] I. Mani and E. Bloedorn. Multi-document summarization by graph search and matching. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Menlo Park, California, USA, 1997. AAAI Press.

[7] C. Monz. Document fusion for comprehensive event description. In *Proceedings of the workshop on Human Language Technology and Knowledge Management*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[8] D. R. Radev. A common theory of information fusion from multiple text sources step one: cross-document structure. In *Proceedings of the 1st SIGdial workshop on Discourse and dialogue*, pages 74–83, Morristown, NJ, USA, 2000. Association for Computational Linguistics.

[9] K. Rajaraman and A.-H. Tan. Knowledge discovery from texts: a concept frame graph approach. In *CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management*, pages 669–671, New York, NY, USA, 2002. ACM Press.

[10] J. F. Sowa. Semantics of conceptual graphs. In *Proceedings of the 17th annual meeting on Association for Computational Linguistics*, pages 39–44, Morristown, NJ, USA, 1979. Association for Computational Linguistics.

[11] J. F. Sowa and E. C. Way. Implementing a semantic interpreter using conceptual graphs. *IBM J. Res. Dev.*, 30(1):57–69, 1986.

[12] N. Stokes and J. Carthy. First story detection using a composite document representation. In *HLT '01: Proceedings of the first international conference on Human language technology research*, pages 1–8, Morristown, NJ, USA, 2001. Association for Computational Linguistics.

[13] T. Tsikrika and M. Lalmas. Merging techniques for performing data fusion on the web. In *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*, pages 127–134, New York, NY, USA, 2001. ACM Press.

[14] Y. Zhai and M. Shah. Tracking news stories across different sources. In *MULTIMEDIA '05: Proceedings of the 13th annual ACM international conference on Multimedia*, pages 2–10, New York, NY, USA, 2005. ACM Press.

Semantically Annotating the Desktop

Towards a Personal Ontology

Jimmy Borg
Department of Computer Science and AI
University of Malta
jbor059@um.edu.mt

Matthew Montebello
Department of Computer Science and AI
University of Malta
matthew.montebello@um.edu.mt

ABSTRACT

The advent of the World-Wide Web brought with it a proliferation of information from e-mail, forums, chat, sites that rapidly led to information overload and a subsequent storage problem and maintenance on users' personal computers. The desktop has become a repository of data that hosts various types of files. The recent massive increase in data has resulted in a continuous attempt to enhance our data organisation techniques and hence to the development of personal information management software.

In this paper we present an overview of data organisation techniques related to personal data management that have been an active research area for decades. We will look at how personal information managers handle different types of files, and abstract these file types into a single user interface. Despite their advanced user interfaces, we argue that traditional personal information managers tend to be very domain specific and lack in user adaptability. To address these limitations we propose a semantic desktop application that exploits the flexibility of semantic web technologies, and introduces the concept of a Personal Ontology to aid in data organisation and can be used by other desktop applications such as information retrieval and intelligent software agents.

1. INTRODUCTION

With the introduction of the PC, computers have moved from single purpose to multipurpose machines. Personal Computers are no longer only used to maintain a database or to run a payroll application. PCs have become an integral part of our daily life. On our PC we watch videos, play audio files, watch TV and store collections of music CDs that we used to place on a shelf. We can take, store and share digital photos. We can chat, write emails, maintain calendars and reminders and store our contact information list. Nowadays books are being stored in electronic format, libraries are becoming online bookstores, magazines and newspapers are being published digitally and huge collections of scientific papers and articles are accessible through the World Wide Web.

This popularity of the PC and the World Wide Web has exposed our machines to a huge amount of new data that needs to be stored, maintained and easily accessed. It is no longer a question of whether we have the information, it has become a question of how we are going to find the required information. Web search engines do the job very well but unfortunately their desktop counterparts are still quite limited. Documents on the desktop are not linked like web pages and thus algorithms such as PageRank cannot be used [3]. Ironically enough, in some cases one may find it more efficient to search for the information on the Web rather than on his or her own personal computer.

In the rest of the paper we will be discussing the organisation and retrieval of personal information. In Section 2 we will define what we mean by personal information, then we will discuss how we can manage personal information and finally we will give an overview of different types of personal information management software. In Section 3 we will proceed by proposing the Semantic Desktop, a system that semantically annotates the data on a user's personal computer and, by using standard Semantic Web languages for information representation, creates a Personal Ontology. We will conclude the paper by discussing some possible environments where the Personal Ontology can be used.

2. PERSONAL INFORMATION

Personal information can be defined as data that a single user stores on his or her personal computer. This information can be of different types and we can produce a never ending list of information that can be classified as personal. To get an idea, such information might include;

- Calendar Entries such as birthdays, anniversaries, appointments, meetings and other significant dates,
- Email Repositories,
- Instant message archives,
- Contact information such as telephone numbers, mobile numbers and postal and email addresses,
- Files of various types such as documents, papers, photos, digital books, video clips and web pages,

CSAW 2006, MCST Bighi, Malta

- Various types of lists such as reminders, notes, bookmarks and RSS/Atom Feeds.

In the rest of this section we will discuss ways of how this personal information can be organised, in other words, personal information management. We will then give an overview of different applications that aid in personal information management, also known as personal information managers.

2.1 Personal Information Management

The area of personal information management has a long history composed of very interesting examples that helped in shaping today's theories. Some even date back to the pre-computer area, such as the famous article "As we may think" [2] by Dr.V. Bush. In his article of 1945, Bush describes his visionary system, *Memex*, as

> *"a device in which an individual stores all his books, records, and communications, and which is mechanized so that it may be consulted with exceeding speed and flexibility. It is an enlarged intimate supplement to his memory."*

Although limited to only analog devices, *Memex* not only was able to store different types of mediums but it allowed searching to be performed in an associative matter, like the human mind. We can say that today we are very close to this kind of system since the computer can store and access many different types of files. Although memory is now becoming much more affordable, it is not feasible and not necessary for all the data to be stored on the same machine. Nowadays the web can be seen as a repository of data that extends our own storage space. Recent approaches to information retrieval and organisation such as [1] are based on this idea.

Despite the fact that our mind thinks in an associative manner, we cannot simply eliminate the traditional indexing and categorisation approach. This approach is the most widely used and over time it has proved itself to be very efficient and effective in many situations. For example one cannot question an indexing approach on a telephone directory or a categorisation approach on a web directory. However we might question other areas that we may take as obvious, such as, "Is a tree organisation the most suitable for a file structure?" and "Is a relational database an appropriate storage method for an email repository?". We will discuss these issues in more detail in Section 3. We will now proceed by looking at some personal information managers that are widely used, and highlight similarities and possible improvements to these tools.

2.2 Personal Information Managers

Personal information managers, or PIMs, are tools that help users to store, maintain stores, search and retrieve personal information. In other words a personal information manager's aim is to aid in the organisation and retrieval of data in a single user perspective. Typical challenges that personal information management software encounter are:

Huge amount of information, As we already discussed in Section 1, personal information managers typically deal with huge amount of information that we use everyday, being either information that we directly access, for example when reading an email, or information that is indirectly accessed by the application, for example when checking for new emails.

of different type and nature, Information does not only consist of different types of files but also databases, archives and online links. For a typical list of personal information that a user may regularly use one can refer to the beginning of this section.

coming from different sources The information may not only be stored on the computer's hard drive but can also reside on network drives, servers, web pages or other online repositories.

As the amount of information that we deal with everyday is increasing, personal information managers are becoming more popular since they can minimise the burden from our memory. One can find applications of different flavours that target different types of users. Some users use a PIM for storage and retrieval of data. Others use it as a communication tool, to send emails, fax and instant massages. Others try to make their lives more organised by keeping important calendar dates, to-do lists and meeting reminders. In general we can categorise information management software as PC based, web based or PDA.

PC based packages are the oldest and typically tend to be the most feature oriented. Most consist of email programs, contact list, organisers and maybe a calendar. A typical and very widely used application is Microsoft Outlook [17], which packages many features under a single user interface. Other applications such as Lotus Notes [18] offer a networked flavour, typically more oriented for the business class. The application also includes an advanced semi-automatic meeting scheduler.

Web based solutions usually take an organisational approach and unlike the PC based applications lack the storage of large data. Typically these applications range from email clients to calendars and schedulers. Web applications offer the advantage of accessibility from any internet enabled machine, not only from the user's personal computer. A typical example of a web based system is the relatively new Google services, which range from an email client, calendar, scheduler and a document editor.

Personal Digital Assistants, or PDAs, are mobile devices designed for being used as personal organisers. Their main merit is mobility, on the other hand, they usually lack in memory and their functionality depends very much on the operator's connectivity. The functionality of PDA software is very similar to web based systems and typically includes a combination of email, calendar, reminder, address book and notes. An interesting, typical feature of PIM software on a PDA is that it can synchronise with other PIM software on a personal computer.

In general, these systems are targeted at different classes of users. The information structure and functions are built upon the targeted user's needs, for example Microsoft Outlook is targeted for a typical home user that needs to access

mail and maybe keep a simple calendar of events. On the other hand Lotus Notes provides features that are more targeted for the business class of users, providing them with a more advanced meeting scheduler, email access over a networked environment and an advanced user profile system. In the next section we will argue that by focusing on the meaning of the data, rather than the user we can build an application that adapts itself according to the user's needs.

3. THE SEMANTIC APPROACH

> *"The dream behind the Web is of a common information space in which we communicate by sharing information. Its universality is essential: the fact that a hypertext link can point to anything, be it personal, local or global, be it draft or highly polished. There was a second part of the dream, too, dependent on the Web being so generally used that it became a realistic mirror (or in fact the primary embodiment) of the ways in which we work and play and socialize. That was that once the state of our interactions was on line, we could then use computers to help us analyse it, make sense of what we are doing, where we individually fit in, and how we can better work together."* -Tim Berners-Lee [4]

That was the initial vision of Tim Berners-Lee, the creator of the World Wide Web as we know it today. However we can note that the second part of his dream is not yet achieved, and this is where we are moving to, the Semantic Web. He defines the Semantic Web as

> *"an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."* -Tim Berners-Lee [5]

In the rest of this section we will discuss how we can apply information management techniques and Semantic Web technologies to the personal computer in order to improve personal information management and collaboration.

3.1 A Semantic Desktop

Semantic desktop applications are quite innovative. The idea evolved from the vision of the Semantic Web itself. The semantic annotation of data on the desktop will allow for the integration of desktop applications with the Semantic Web. The personal computer, as a repository of data, can be seen as a small Web in itself. By annotating the data on the PC we will be placing the first building stone for the Semantic Web. Since the Semantic Web is still in its infancy, current semantic desktop applications are generally built for research purposes. Some of these applications include:
Haystack system at MIT [6],
Gnowsis system at DFKI [10],
D-BIN by SEMEDIA [11],
OpenIris by SRI [12] and
Chandler system by the OSA foundation [13].

A typical semantic desktop application can be split into three main components, namely, the ontology, applications that create and maintain the ontology and applications that make use of the ontology. The foundation of the system is the ontology. This uses standard general purpose languages for information representation, such as RDF and OWL [14]. A typical ontology stores the semantic metadata of all the personal information of the user, thus in this paper we will refer to it as a *Personal Ontology*. The Personal Ontology consists of three levels. At its most basic state, the ontology describes the structure of basic file types. We can call this the *Storage Ontology*. The *Preferences Ontology* is used to store the user's preferences and settings, which are not only used by the semantic desktop application but can be used by all other applications. The third level of the *Personal Ontology* is the *Content Ontology* and is used to describe the content of the files.

The second component of a Semantic Desktop system is a mechanism that modifies the underlying ontology. This will typically consist of several function specific applications, such as an email client, a file browser and a calendar. There are two approaches that one can take when developing a semantic application; the monopolistic approach and the integrative approach. In a monopolistic approach, the semantic desktop application will replace many existing applications and group all the functions into a single user interface. Although this method generally sounds neater, it requires the user to adapt to a new system. An example of such system is Haystack. The integrative approach, adapted by the Gnowsis application, will add extra functionality into the existing applications and make them interact with the ontology. This does not require the user to learn a new system and it can reduce the development effort. However such approach may be limited by the flexibility of the third party applications.

The ontology can be virtually composed of any type of personal information. It is the applications that create and maintain the ontology that limits the extent of the Personal Ontology. Different systems provide different sets of applications, depending on their scope and size of the project. Typical, basic functions that one will find present in almost all systems are emails, calendar events and browser cache. A common challenge in these systems is the annotation of the file system. The user spends considerable time building complex folder classification hierarchies thus it is of vital importance for the semantic desktop application to use this information. However the current operating systems lack the much needed support for file-handling event triggering. [3] proposes a similar application that uses an in-notify enabled Linux kernel while [9] proposes a similar system on Windows. Having the Personal Ontology created and appropriately maintained, it will become a question of how the ontology can be used.

3.2 Using the *Personal Ontology*

We will proceed by identifying possible ways of utilising the Personal Ontology, most of which consist of quite novel research areas. Data organisation is the most obvious utilisation of the ontology, and is the main subject that we have discussed till now. The difference between a semantic desktop application and other personal information managers is that the semantic organiser can change the way of presenting the information to the user according to the information

itself. We can illustrate this by a simple scenario regarding the usage of contact information; in a company the manager will need to know detailed information about a contact such as the name and surname, telephone, fax and mobile numbers and email and postal addresses. On the other hand in personal contact list used only for telephone numbers the user might need to store the name and surname, or maybe a nickname, the telephone number and possibly a mobile number. As discussed in [7], by building the user interface upon the Storage Ontology, the user will be presented with only the required data.

A key element in every user adaptive system is the context information. The Personal Ontology can be an invaluable element for making an application user adaptive. Since the ontology uses standard semantic web languages, it can be accessed by any semantic web application, not just by the Semantic Desktop system. An approach that is quite new to desktop applications is the use of content ontology, partially described in [8]. The idea behind the Content Ontology is to semantically annotate the content of the files, especially documents and emails. The application will then be able to analyse the Content Ontology of different files and suggest possible relations between the files. While the Content Ontology can make the system more adaptive, the Preferences Ontology can make the system more adaptable and share a generalised set of user preferences between several applications.

As P.A. Chirita et al states, in [3], current approaches to desktop search, such as Google Desktop search [15] on Windows or Beagle [16] on Linux, do not include metadata in their system but only perform searching using regular text indexing. This causes such systems to perform poorly when compared to their web counterparts. The key element that makes web search systems very effective is the linking between the elements, which is virtually inexistent on current file systems. The Personal Ontology, especially if the Content Ontology level is implemented efficiently, could fill this gap. The document links can help to apply result ranking techniques [19] in desktop search algorithms.

The Social Semantic Desktop can be seen as a networked collection of Semantic Desktops. The idea is to create an environment where data and metadata can be easily shared between peers. Peers, or agents on personal computers, can collaborate together and form communities to exchange knowledge while reducing the time for users to filter and file the information [20]. The Semantic Desktop is one of the three main components of the Social Semantic Desktop. The Semantic Desktop system, in conjunction with Peer To Peer services, provides a mechanism for users to share their information. The third component, Social Software, maps the social connections between different people into the technical infrastructure.

Other systems use the ontology for more specific purposes. For example IRIS provides a Semantic Desktop interface that builds a desktop ontology which will be used as a learning environment for the CALO Cognitive Assistant project [22]. CALO, [21], is a personal assistant that learns by applying automated machine learning techniques on a user's personal data.

4. CONCLUSION

In this paper we have presented techniques that can be used to semantically annotate personal information on a user's personal computer, thus creating a Semantic Desktop. Having the data semantically annotated using standard Semantic Web languages make it possible for applications to integrate the desktop with the Semantic Web.

To conclude, we can say that the Semantic Desktop system goes beyond the purpose of data organisation. The Personal Ontology can be adopted and used for different purposes ranging from file organisation, to machine learning environments, to the creation of a large semantic network where both users and applications can reason about the shared knowledge.

5. REFERENCES

[1] D. Elsweiler and I. Ruthven and L. Ma, *Considering Human Memory in PIM*, SIGIR 2006 Workshop on Personal Information Management, August 10-11, 2006, Seattle, Washington

[2] Vannevar Bush, *As We May Think*, The Atlantic Monthly, 176(1), p101-108, July 1945

[3] P.A. Chirita and R. Gavriloaie and S. Ghita, *Activity Based Metadata for Semantic Desktop Search*, In Proc. of Second European Semantic Web Conference, ESWC2005, May 21 - June 1, 2005, Heraklion, Crete, Greece

[4] Tim Berners-Lee, *The World Wide Web: A very short personal history*, http://www.w3.org/People/Berners-Lee/ShortHistory.html

[5] Tim Berners-Lee, James Hendler, Ora Lassila, *The Semantic Web*, Scientific American, May 2001

[6] Haystack Project, http://haystack.lcs.mit.edu/

[7] E. Adar and D. Karger and L.Stein, *Haystack: Par-User Information Environments*, Conference on Information and Knowledge Management, 1999

[8] B. Katz and J. Lin and D. Quan, *Natural Language Annotations for the semantic web*, ODBASE, 2002

[9] Leopold Sauermann, *The Gnowsis, Using Semantic Web Technologies to build a Semantic Desktop*, Master's Thesis, TU Vienna, 2003

[10] Gnowsis Project, http://www.gnowsis.org

[11] D-Bin Project, http://www.dbin.org/

[12] IRIS Semantic Desktop Project, http://www.openiris.org/

[13] Chandler Project, http://chandler.osafoundation.org/

[14] Web Ontology Working Group

[15] Google Desktop Search Application, http://desktop.google.com/

[16] Gnome Beagle Desktop Search, http://www.gnome.org/projects/beagle/

[17] Microsoft Outlook, www.microsoft.com/outlook/

[18] Lotus Notes, http://www.lotus.com/notes/

[19] Stefania Costache, *Using Your Desktop as Personal Digital Library*, TCDL Bulletin, 2006

[20] S. Decker and M. Frank, *The Social Semantic Desktop*, DERI Technical Report 2004-05-02, May 2004

[21] CALO Project, http://www.ai.sri.com/software/CALO

[22] A. Cheyer and J. Park and R. Giuli, *IRIS: Integrate. Relate. Infer. Share.*, In Proc. of Fourth Intl. Semantic Web Conference Workshop on the Semantic Desktop, Galway, Ireland, Nov. 2005

Formal Verification of Enterprise Integration Architectures

Dr. Ernest Cachia
University of Malta, Msida, Malta
ernest.cachia@um.edu.mt

Mark Vella
University of Malta, Msida, Malta
mvel0022@um.edu.mt

ABSTRACT

This is a near-finished paper to be presented in an international research conference.

Weak Bisimulation is a process calculus equivalence relation, applied for the verification of communicating concurrent systems [Miln 99]. In this paper we propose the application of Weak Bisimulation for Enterprise Application Integration verification. Formal verification is carried out by taking the system specification and design models of an integrated system and converting them into value passing CCS (Calculus of Communicating Systems) processes. If a Weak Bisimulation relation is found between the two models, then it could be concluded that the EI Architecture is a valid one.

The formal verification of an EI Architecture would give value to an EI project framework, allowing the challenge of cumbersome and complex testing typically faced by EI projects [Khan 05], to be alleviated, and thus increasing the possibility of a successful EI project, delivered on time and within the stipulated budgeted costs.

This paper shows the applicability of value passing CCS (or equivalent) formal notation to model the EI systems characteristics, as well as investigates into the computation complexity of available weak bisimulation algorithms, in order to analyze the applicability of this proposition in real life.

1. Background

In the process of searching for an Enterprise Integration (EI) specific framework to guide an integration team in the strategic implementation of an integrated IT landscape within and beyond the scope of a single enterprise, an extensive research in the following areas of Enterprise Integration was made: -

- Integration and middleware technology [Cumns 02] [Linth 03] – for acquiring the understanding of the technological mechanisms that make systems integration possible.
- Standards [BPMI][Linth 03] [OMG 04a] [OMG 04b] – to be aware of the agreed upon standards in order to work on their lines.
- Scientific foundation [Miln 99] [Press 96] [OMG 04b] – in order to be able to add value to the field of Enterprise Integration based on the concepts of Computer Science and Software Engineering.
- Challenges [Gar 01] [GB & Ruh 04] [Khan 05] [Linth 03] [Lubl & Far 02] [Mav 03] [Sif 01] – in order to locate those Enterprise Integration specific areas that need improvement.
- Methodology and Best Practices (Linthicum 2003) [Ruh et al 00] [Sif 01], [Schm 03] – in order to have the knowledge of existing improvement efforts and possibly build on them.

The industry research, made up mainly of compiled industry reports, articles [BIJ] [IntCons.] and literature [Burl 01] [Cumns 02] [GB & Ruh 04] [Linth 03] [Ruh et al 00], allowed the broad understanding of the current state of EI projects in industry; from the projects business drivers, technologies and methodologies being used, to the factors affecting the success of these projects. On the scientific level, the research investigated which areas of computer science and software engineering could be applied to EI projects, in order to improve the situation of this area. [Miln 99] [Press 96] [OMG 04b]

1.1 Software Engineering Principles

A sound software engineering framework is one that delivers high quality software deliverables on budget and on time. [Ghezz et al 02] [Press 96] [Somm 04] In the case of an EI-specific framework, it is being proposed that in addition this would be a framework that is targeted specifically at EI systems, that achieves the EI-specific goals and qualities, allowing the EI project challenges to be overcome, and thus maximizing the probability of success and avoiding project failures as identified in [Lubl & Far 02].

1.2 EI-specific Framework Value

The proposed value of an EI-specific Project Framework could be better explained in the following scenario: take a software project manager who has managed traditional software projects for some time, but is now faced with the challenge of setting up an EI project plan. He/she should be aware of the fact that managing an EI project, although still a software project, requires a specific management framework to address the specific EI challenges. An EI-specific management framework would be very beneficial, in this particular case, to start building the EI project plan and carrying out all the necessary tasks leading to an effectively built EI system.

1.3 EI Project Challenges, Goals and Qualities

Further to what was presented in [Gar 01], according to [Lubl & Far 02] and [Herr 04] the main challenges causing failure in EI Projects include:

- Lack of standard methodologies – so far only industry best practices and EI product specific methodologies were found.
- Lack of proper business process definitions – in fact many business models today exist only in the head of

departmental managers and at times these also conflict with the understanding of their colleagues.

- Lack of business units co-operation – In several cases, business units only communicate to put the blame on each other, and compete fiercely for company budgets. On the other hand, EI projects require full business unit co-operation.
- Implementation is more complex than expected – An EI implementation usually consists of several implementation technologies, packages from different vendors, multiple platforms and an unexpected number of interface links.
- Relying too much on integration technology for project success – Middleware technology in fact is only an enabler of integration implementation but far from being a complete software engineering tool.
- Lack of thorough testing - this is so, given the newly introduced integration level and the enterprise wide scope of such systems.
- Lacking the proper integration team roles – EI projects, given their novel nature, are only seen by the IT department as just another IT project, and fail to re-organize the IT roles before the start of these projects.

[Ruh et al 00] compiled a list of goals and qualities that should be reached/exhibited by an EI-specific project framework. These are as follows: -

Goals

- Ensure that the EI architecture and developed applications satisfy business needs
- Describe how to manage the EI process
- Describe how to work with legacy systems and packaged solutions to integrate them
- Provide guidance on technology selection and standardization
- Ensure that the methodology promotes reuse

Qualities

- Align IT with the enterprise business strategy
- Build on a solid enterprise architecture
- Leverage legacy and commercial software
- Focus on security

In addition to these goals and qualities, the main strategic business value of EI today is that of being an enabler of Business Process Management [McGov 01] [McGov 03] [Krish 04]. This point was taken into consideration and a decision was taken to focus on the Business Process Integration [GB and Ruh 04] type of integration, where the main focal points of integration are the business process and not the applications. Here, Application Integration is only a consequence of joining up the business processes, but not the main driver.

These goals and qualities along with the EI project challenges form the basis for the reasoning underlying an EI-specific framework.

1.4 Framework Building Blocks

The foundation of this framework is made up of building blocks from the fields of Computer Science, Software Engineering and Business Management. These building blocks are: -

Value passing CCS (Calculus of Communicating Systems) or equivalent Process Calculus – from the field of Computer Science that allows the mathematical modeling of communicating concurrent and mobile systems. [Miln 99] CCS allows the formal specification and reasoning of communicating concurrent systems. Its applicability to the EI domain is shown in section 3 of this paper.

Unified Modeling Language (UML) – a software engineering tool allowing the modeling of software specification and design. [OMG 2004b] UML was chosen due to the wide adoption in the software engineering world and its OMG standard status.

Business Process Management (BPM) – a business management discipline born out from Business Re-engineering that advocates end to end business process modeling, automation and their continuous monitoring and optimization. [Burl 01] This building block was made part of the project in order to be able address the goal of EI systems to be an enable to Business Process Management programmes.

This paper concerns only the application of the first building block: value passing CCS. More specifically, Weak Bisimulation, that is a binary relation on CCS processes [Miln 99], is being proposed as a possible tool for the verification of EI architectures.

1.5 Scope

The whole process required for completing the formal verification of an EI architecture is illustrated in figure 1, where the business process model defines the system specification and the EI Architecture is the system design that is verified against the business process model.

This paper shows the applicability of process calculus such as value passing CCS to the domain of EI, present a treatment of Weak Bisimulation and the possibilities this offers as a formal verification tool, and place the verification step within the context of an EI-specific framework. The formal specification and design, as well as an in-depth look at the verification process will be treated in subsequent papers.

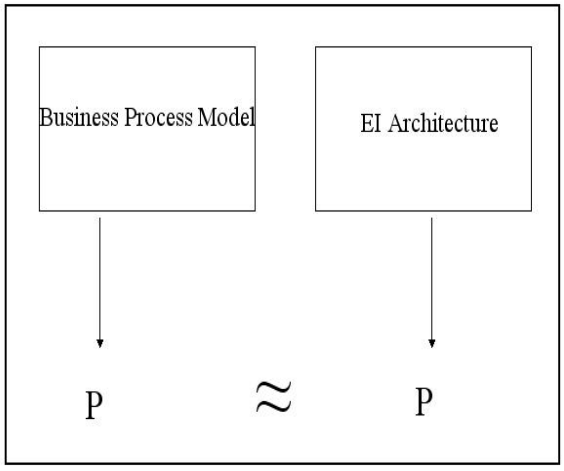

Figure 1 – EI Architecture verification process

2. Formal EI Architecture Verification Proposition

Weak Bisimulation (≈) is a process binary equivalence relation based on the equivalence of just the observable reactions between 2 processes. In other words, as long as the second process can match each observable reaction sequence of the first process and vice versa, the 2 processes are regarded as weak bisimilar, irrespective of their internal reactions. Thus Weak Bisimulation is also known as observation equivalence, with the processes in question regarded as black boxes

[Miln 99] shows how the weak equivalence relation (≈) could be used to prove that a particular system structure implements correctly a particular system definition. More specifically he showed the application of Weak Bisimulation as follows: ***System ≈ Specification.***

Weak bisimulation is the chosen process equivalence relation for the fact that a system specification does not have as yet an internal structure and as a consequence, minimal internal reactions. Thus, this reasoning rules out both Structural Congruence (≡) and Strong Bisimulation (~). Given that Weak Equivalence (≈) is insensitive to both internal reactions and structure [Miln 99], it fits well the need of proving the correctness of the implementation of a system against its specification.

Applied to the domain of Enterprise Integration, or more specifically to the chosen Business Process Integration type of integration, this observation equivalence relation is being proposed to be applied as follows: -

Business Process Model ≈ EI Architecture

The Business Process Model defines the required business process flows to be automated by the underlying system, whilst the EI Architecture is the design of the EI system implementing the business flows. By mapping these models into value passing CCS processes, the equivalence between the Business Process Model and the EI architecture, could be formally verified by finding a Weak Bisimulation relation between these two processes.

3. EI systems characteristics

From the initial research underpinning this paper, the following architectural characteristics of EI systems stood out: -

- Concurrent Systems [Cumns 02] [Linth 03]
- Business Process Modelling [GB and Ruh 04] [McGov01]
- Mobility [Smit and Fin 03]
- Security [Ruh 00]

The next four sub-sections introduce these characteristics and show how value passing CSS fairs, in modeling these characteristics.

3.1 Concurrent Systems

In EI architectures, concurrency is exhibited by the several applications, services and middleware executing in parallel, whilst messaging is the communication link between them. In Enterprise Integration, messaging is carried out by several middleware technologies that link applications together. [Linth 03] categorizes the middleware technology available today as follows: -

- Remote Procedure Calls – this type of middleware allows a software process to make synchronous calls to remote processes. E.g. Java Remote Method Invocation (RMI) [JavaRMI]
- Message Oriented Middleware – this type of middleware is a queuing software that allows software processes to write and read messages to and from a queue. Communication between processes is asynchronous with guaranteed delivery. E.g. MQSeries. [IBMMQ]
- Distributed Object Transactions – this is a middleware infrastructure allowing the exposure of business logic making up applications, supported by a transactional platform. E.g. Component Object Model (COM) [Microsoft.com/com.] and Common Object Request Broker (CORBA) (CORBA]
- Database Oriented Middleware – this kind of middleware provides software processes with access to database servers. E.g. Open Database Connectivity (ODBC) [IODBC]
- Transaction Oriented Middleware – this type of middleware provides co-ordination of information movement and method sharing within the scope of a transaction. These are mainly to link legacy procedural applications to the transactional enterprise level. E.g. Tuxedo [Bea]

As defined by [Miln 99], value passing CCS – a formal way of modeling concurrent communicating systems, where variables are allowed along communication channels - is able to model concurrency and messaging, in the following ways: -

Concurrency

Being an extension of the CCS (Calculus of Communicating Systems) [Miln 99] value passing CCS supports the modeling of concurrent processes with the construct $P ::= (P1 \mid P2)$, where processes *P1* and *P2* are parallel composed together.

Messaging

The following value passing CCS constructs support the modeling of messaging as represented in Business Process Models (Figure 2) and EI Architectures (Figure 3). Figure 4 shows how these are represented in value passing CCS.

Input channel - $x(y).P$, means input a name on channel *x* by substituting with place holder *y*, and use the input in process *P*. In the EI scenario, *x* can represent a listening port such as a web service. The name y represents the place holder for an incoming message. Continuing on the example of a web service implementation, this incoming message can be an input XML (eXtendible Markup Language) document to the web service (a tagged data document), whose schema is referenced in the web service WSDL (Web Service Definition Language), which defined the interface of the particular web service.

Output Channel - $\bar{x}\langle y\rangle.P$ means output the name *y* on the channel named *x*, and then do *P*. In previous the web service analogy, this represents the consumption of the web

service, where an XML document y is sent from the client application along channel x. In this case *x* is the TCP/IP based connection using the SOAP protocol.

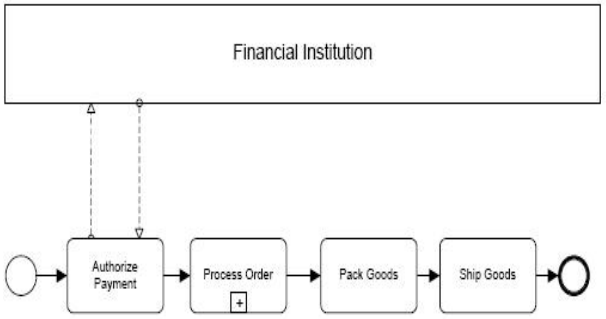

Figure 2 – Business Process Modeling Notation – Sending a message

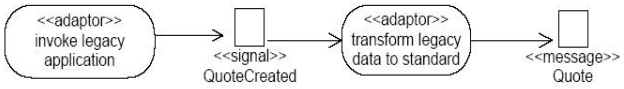

Figure 3 – Messaging in UML Activity Profile for EAI (Enterprise Application Integration) [OMG 04a]

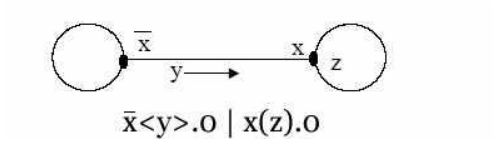

Figure 4 - Messaging in value passing CCS

In EI, messaging can be either ***synchronous*** for example in the case of a web service call or a Remote Procedure Call (RPC), or ***asynchronous***, as in the case of message queues. It is possible to model both these types of messaging using value-passing CCS as follows.

Synchronous Messaging

$P = new\ x\ (\ P1 \mid P2\ )$

$P1 = \bar{x}\langle y\rangle.x(z).P1'$

$P2 = x(a).\tau.\ \bar{x}\langle b\rangle.P2$

Using the web service analogy in the above simplified process; an application process *P1* consumes web service *P2*. *P1* calls *P2* along channel *x* and passes the XML document *y* as an input. When the call is made, *P1 -> P1''* transition occurs, where *P1'' = (z)x.P1'*. In the *P1''* state, the calling process is kept blocking waiting on the same channel for web service *P2* to return. Once a message is returned back along channel *x* the calling process goes into state *P'* executing the rest of the process.

From the server's perspective *P2* listens indefinitely on channel *x*. When an XML document arrives along channel *x*, the web service executes its internal logic, represented by the tau (τ) symbol, and returns an output document XML back on channel *x* to *P1* on completion. The web service then resumes in state *P2* listening indefinitely for the next call.

Asynchronous Messaging

$P = new\ x\ y\ (\ P1 \mid P2 \mid P3\ )$

$P1 = \bar{x}\langle y\rangle.P1$

$P2 = x(w).\tau.\bar{y}\langle w\rangle.P2$

$P3 = y(z).\tau.P3$

Process *P2* handles continuous asynchronous communication between processes *P1* and *P3*. Even though the communication between *P2* and the other processes is synchronous in terms of readiness of processes to be able to communicate between each other, process *P2* provides the mechanism for asynchronous messaging between *P1* and *P3*. In the above-simplified example, *P2* models a messaging queue-like structure, allowing application *P1* to asynchronously call *P3* without blocking, even in cases when *P3* is not ready to communicate.

3.2 Business Process Modeling

Business Process Modeling involves the modeling and documentation of the business process flows within an enterprise and is a main element of Business Process Management [Burl 01] [BPMI] [Smit & Fin 03]. EI systems are expected to serve as an infrastructure to these modeled processes, possibly by means of a Business Process Management System. [McGov 01]

CCS abstracts the notion of a process and is not specific to any particular software process living in a computer memory. Thus, it can be argued that CCS could also be suitable for modeling processes in the business sense. In the business context, we have business processes, embodying workflows, running in parallel communicating between each other inter-departmental and business to business (B2B) messages. In [Smt & Fin 03], Smith and Fingar elaborate in full detail of how Pi Calculus, an extension of CCS incorporating mobility, perfectly suites the modelling requirements of business processes.

As a matter fact, Process Calculi are already being applied to BPM for other reasons. The Business Process Modelling Notation (BPMN) adopted by the Business Process Management Initiative (BPMI) [BPMI] is fully based on Pi Calculus foundations; as is the Business Process Execution Language For Web Services (BPEL4WS) by Microsoft Corporation (Microsoft.com) and IBM [IBM] and the BPML (Business Process Modelling Language) by the BPMI.

3.3 Mobility

From personal experience in enterprise integration projects, the communication links between the nodes of an EI architecture (applications, services and middleware) are not of fixed nature but rather of a dynamic mobile nature, where communication channels are created, moved and destroyed. For example we have the scenarios where communication channels between two processes are created as in the discovery of a web service by UDDI (Universal Discovery Description and Integration) protocol [UDDI]. There are also situations of channel proliferation where for example an application server or middleware becomes unavailable. There

are also situations of pure mobility where a communication channel is relocated in the process space as in the scenario where a client application is instructed to start communication with an alternate server for example, for performance reasons or during a seamless, no down time, new application roll out.

Mobility in the context of business processes is reflected in the continuous change in business rules as a result of a Business Process Management Programme [Smith and Fingar, 2003]. An example is where in a move to eliminate bureaucracy an electronic components manufacturing company consolidates approvals of component designs in one department. In this case a link between the engineering department and the first auditing department in line, moves to a link with the newly created consolidated department

Pi Calculus supports the modeling of process mobility by definition; in fact Pi Calculus was invented by extending the CCS to support mobility. Mobility is modeled in Pi Calculus by allowing names representing communicating channels to be themselves passed as messages along channels. [Miln 99]

Even though Pi Calculus would be required to model mobility, for the time being only value passing CCS is being considered in order to make the proposition of this verification tool clearer. Moreover, the researched Business Process Modeling Notation [BPMI] does not include the concept of mobility within the graphical notation; thus making value-passing CCS sufficient for formally representing business models built using this notation for the time being.

3.4 Security

Now that the applications have been 'opened up' in order to communicate possibly with the whole world, the issues of security breaches increase [Ruh, et al, 2000]. An improper security infrastructure can invalidate an otherwise well built integrated architecture. In fact the role of secure messaging increases immensely in such architectures. The importance of security in EI projects has already been identified in the Secure Application Integration Methodology by [Ruh et al 00].

The notion of restriction in CSS denoted by *(new x) P* means that *x* is for the exclusive use of *P*. A more specific example is *P = new x (P1 | P2)* which means that *P1* and *P2* communicate via a private channel x even when placed in the context of other concurrent processes having a channel with the same name [Miln 99]. This construct enables us to model secure communication channels in integrated architectures.

Having said that, this does not mean that by using the restriction construct, it means that model is definitely secure, it only specifies that the indicated channel of communication should be secured. A typical case where a secure channel does not imply complete overall security is shown in the following system: -

System = new x (P1 | P2) | P3

$PI = \overline{x}\langle outval \rangle$

$P2 = x(inval).\overline{y}\langle inval \rangle$

P3 = y(inval)

Where even if *P1* and *P2* communicate via a secure channel, over which *P3* can never interfere, it is still up to the implementation of P1 and *P2* to ensure that that any sensitive data is not passed on to *P3*. In the above example, *P2* is sending the data received over the secure channel *x* to *P3*.

Still, having the individual processes formally defined, these can be individually formally verified in terms of system security.

4. Enterprise Architecture Verification

It is being proposed that CCS based verification occurs within an EI framework as follows.

Once the Business Process Model diagram (the system specification), and the EI architecture (system design) are produced, these are converted to CCS processes. This way the model passes from a semi-formal representation of the system to a formal one. If a Weak Bisimulation relation were found between the two processes, the EI architecture would be considered equivalent in behavior to the Business Process Model, and thus validated.

This verification step is carried out during the framework stage where the EI Architecture is completed and the lower level steps of design and implementation are about to start (Figure 5). Having the architecture validated at such an early stage maximises the success of the EI system by alleviating the challenge of EI testing as discussed in [Khan 05] and [Lubl & Far 02]

5. Practical applicability

In order for the proposed EI Architecture formal verification tool to be practical and usable in real world applications, the verification process must be automated, and the algorithm automating the process should do this in an efficient manner.

[Baier & Herm 99] explain that Weak Bisimulation can be decided in a time complexity of $O(n^{2.3})$, where n is the number of states, using a technique called the partitioning/splitter technique. Thus, using this algorithm, it is possible to efficiently decide a Weak Bisimulation between two processes given that: -

1. We keep the number of states finite
2. Possibly minimizing the number of states

The first condition can only be adhered to by not using variables from infinite domains (the algorithm does not work on symbolic labeled transition systems). This might sound too restrictive at first, but if it is kept in mind that at this stage the system is simply being described rather than specified in terms computations, it can be viewed from a different aspect. For example, consider a task that receives an integer, and decides on two alternative flow branches based on the value of the integer received. In the system specification and design models, one simply needs to pass a boolean value based on the size of the integer. In this case, instead of using a variable from an infinite domain, one can use a variable from a finite one. The latter can be achieved by assuming synchronous parallelism as described in [Baier & Herm 99], and thus

avoiding a state size explosion when a system is made up of several parallel composed processes.

An alternate approach would be that of using a Weak Bisimulation algorithm that functions on symbolic transition systems, allowing the use of variables from an infinite domain. [Dovier et al.] present an efficient Bisimulation algorithm stating that is possible to port to the symbolic case.

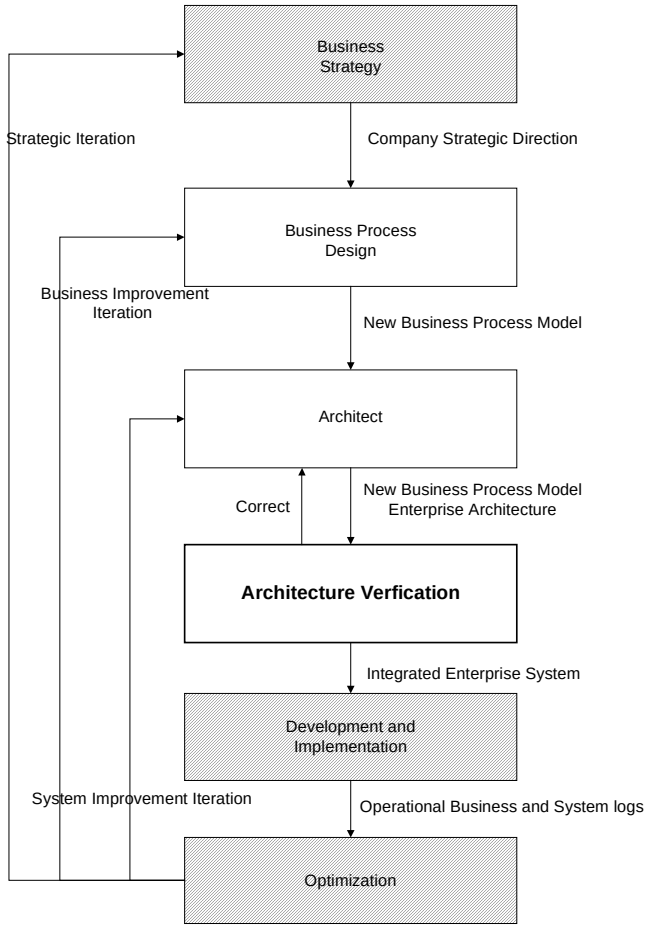

Figure 5 – Architecture Verification as part of an EI framework

6. Conclusions

This paper attempts to show that the application of value passing CCS and Weak Bisimulation as a means of validating an EI Architecture against the Business Process Model is possible by the treatment of the mapping of Business Process Models and EI Architectures to value passing CCS processes. CCS is able to model the main features of EI systems; these being Concurrency, Business Process Modeling, and Security. On the other hand the Weak Bisimulation relation is an equivalence relation that is insensitive to internal structure and reaction. This enables the checking for an equivalence relation between the Business Process Model and the EI Architecture. Whilst Pi Calculus would be required to model the remaining EI characteristic, mobility, this is not considered for the time being since mainly since no Business Process Modeling Notation researched so far addresses mobility, and also for reasons of better focusing on the usage of the proposed formal verification system in real life.

The formal verification of the EI Architecture is proposed to give value to an EI-specific project framework, allowing the challenge of complex testing typically faced by EI projects, to be overcome by allowing the formal verification of the architecture, before development and testing starts.

Finally the paper also proposed a sound foundation for a way forward for a practical application of the proposition, by mechanically automating the verification process.

7. References

[Baier & Herm 99] Baier, C. and Hermanns, G. (1999) Weak Bisimulation for Fully Probabilistic Processes

[Bea] BEA Tuxedo. *http://www.bea.com/framework.jsp?CNT=index.htm&FP=/content/products/tux* http://www.bea.com

[BPMI] BPMI.ORG The Business Process Management Initiative Homepage http*://www.bpmi.org*

[Braun 05] Braunstein, J. (2005). Integration Audit Methodology: A Primer on Assessing Integration. In *EAI Journal* Feb' 2005.

[BIJ] Business Integration Journal Online. *http://www.bijonline.com*

[BPMI] Business Process Management Initiative. (2004). Business Process Modelling Notation v1.0. *http://www.bpmn.org*

[Burl 01] Burlton, R. (2001). Business Process Management: Profiting From Process. SAMS

[Corba] CORBA.ORG. The OMG's CORBA Website. *http://www.corba.org*

[Cumns 02] Cummins, F.A. (2002). Enterprise Integration: An Architecture for Enterprise Application and Systems Integration. Wiley

[Dovier et al.] Dovier A., Piazza, C. Policriti A. An efficient algorithm for computing bisimulation equivalence.

[Erk & Pen 98] Eriksson, H.E. and Penker, M. (1998). UML Toolkit, Wiley Computer Publishing.

[Gar 01] Garimella, K. Ph.D. (2001). Integration Challenges in Mergers and Acquisitions. In *EAI Journal Aug 01.*

[GB & Ruh 04] Gold-Bernstein, B. and Ruh, W. (2004). Enterprise Integration: The essential guide to integration solutions, Addison-Wesley.

[Ghezz et al 02] Ghezzi, C. et al. (2002). Fundamentals of Software Engineering (2md edition). Prentice Hall.

[Herr 04] Herrera, J. (2004). Avoiding Common EAI Implementation Missteps, LogicCurve.

[IBM] IBM.COM IBM Homepage http*://www*.ibm.com

[IBMMQ]IBM WebSphere MQ. http*://www*.ibm.com

[IODC] IODBC.ORG. Platform Independent ODBC. *http://www.iodbc.org*

[IntCons.] Integration Consortium *http://www.wwintegration.com*

[JavaRMI] java.sun.com/products/jdk/rmi. Java Remote Method Invocation (Java RMI) http://www.sun.com

[Khan 05] Khanna, R. (2005). Top Challenges in Integration Projects. Wipro Technologies White Paper.

[Krish 04] Krishnan, M. (2004). The EAI Paradigm Shift, WIPRO Technologies White Paper.

[Linth 03] Linthicum, D. S. (2003). Next Generation Application Integration: From Simple Information to Web Services. Addison Wesley.
[Lubl & Far 02] Lublinsky, B. and Farrel M. Jr. (2002). Top 10 Reasons Why EAI Fails. In *EAI Journal.Dec' 02*
[MSE] MSE *http://www.magicsoftware.com* Magic Software Enterprises
[Mav 03] Maverick, G. (2003). EAI Project Management. In *EAI Journal Nov' 03.*
[McGov 01] McGoveran, D. (2001). BPMS Concepts, Enterprise Integrity. In *EAI Journal Jan '01*
[McGov 03] McGoveran, F. (2003). Managing Business Process for EAI, In *Business Integration Journal Sep '03.*
[Microsoft] Micosoft.com Microsoft Corporation Homepage *http://www*.microsoft.com
[Micr COM]Microsoft.com/com. Component Object Model Technologies *http://www.microsoft.com*
[Miln 92] Milner, R. (1992) Mathematical Structures in Computer Science, Vol. 2, pp. 119-141
[Miln 99] Milner, R. (1999) Communicating and mobile systems: the Pi-calculus. Cambridge
University Press.
[OMG 04a] Object Management Group (2004). UML for Enterprise Application Integration,v1.0. OMG Formal Specification. *http://www.omg.org*
[OMG 04b] Object Management Group (2004). UML Superstructure Specification,v2.0. OMG Formal Specification. *http://www.omg.org*
[OMG 04c] Object Management Group (2004). UML Flow Composition Model v1.0. OMG Formal Specification. *http://www.omg.org*
[OMG 05a] Object Management Group (2005). UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms. OMG Formal Specification. *http://www.omg.org*
[OMG 05b] Object Management Group (2005b). UML Profile for Schedulability, Performance, and Time, v1.1. OMG Formal Specification. *http://www.omg.org*
[Press 96] Pressman, R.S. (1996). Software Engineering: A Practioner's Approach. 4th Edition. McGraw-Hill.
[Roch 04] Roch, E. (2004). A Software Development Methodology for EAI. In *EAI Journal Sept '04*.
[Ruh et al 00] Ruh, A. W., et al; (2000). Enterprise Application Integration: A Wiley Tech Brief. Wiley
[Sang 96] Sangiorgi, D. (1996). A theory of bisimulation for the π-calculus. Acta Informatica, Volume 33 , Issue 1.
[Schm 03] Schmidt, J. (2003). EAI Lifestyle Evaluation. The Software Ecologist Column, In *EAI Journal Apr' 03*.
[Sif 01] Sifter, C.J. (2001). Integration Project Management 101. In *EAI Journal March' 01.*
[Smt & Fin 03] Smith, H. and Fingar P. (2003). Business Process Management: The Third Wave. Meghan-Kiffer Press.
[Stribna 98] Stribrna, J.(1998) Decidability and complexity of equivalences for simple process algebras.
[Somm 04] Sommerville, I. (2004) Software Engineering (7th Edition). Addison Wesley.
[UDDI] UDDI.ORG The Universal Description, Discovery and Integration (UDDI) protocol homepage *http://www.uddi.org*
[V der Aalst 04] Van der Aalst, W. et al. (2004). Workflow Patterns
http://tmitwww.tm.tue.nl/research/patterns/patterns.htm
http://tmitwww.tm.tue.nl
[Whit 05] White S.A. (2005). Process Modelling Notations and Workflow Patterns. IBM corp

An Ontology of Security Threats to Web Applications

Dr. Ernest Cachia, Mr. Mark Micallef
Software Engineering Process Improvement Research Group
Department of Computer Science and Artificial Intelligence
University of Malta
ernest.cachia@um.edu.mt, mmica01@um.edu.mt

Abstract

As the use of the internet for commercial purposes continues to grow, so do the number of security threats which attempt to disrupt online systems[1][8][9]. A number of these threats are in fact unintended[11]. For example, a careless employee might drop a cup of coffee onto essential equipment. However, when compared to the brick and mortar world, the internet offers would-be attackers a more anonymous environment in which to operate. Also, the free availability of hacking tools makes it possible even for the curious teenager to carry out dangerous attacks[3]. Despite this ever-present threat however, it is all too often the case that security is dealt with (if at all) after a web application has been developed[2]. This is mainly due to our software development heritage whereby companies prefer to focus on the functionality of new systems because that provides an immediate return on investment.

As a precursor to proposing an framework for building security into web applications, this paper presents an ontology of threat to web applications. The thinking behind this is that much the same as in the military world, one needs to have as much intelligence about the enemy as possible, the same can be argued in the case of online security threats. Such an ontology would enable stake holder in online applications to take less of a reactive stance but instead be more proactive by being aware what's out there.

Keywords: *Security, Software Quality Assurance, Web Applications, E-Commerce*

1. Introduction

W. Edwards Deming stated several years ago that "the quality of a product is directly related to the quality of the process used to create it"[4]. Although Deming's work involved production work during World War II, his statement holds true today when dealing with complex software systems. As part of a wider body of work dealing with the rapid development of high-quality e-commerce systems[6], the authors of this paper identified the five most important quality attributes in e-commerce systems of which the most important was deemed to be security[5].

Despite the importance of security, it is still often the case that high-profile breaches surface in the news with many more going unannounced[3][8][9]. It appears that development companies focus mostly on functionality when developing a system since this is perceived to provide an immediate return on investment. However, a study amongst 350 online shoppers, by the authors of this paper revealed that 86% of potential customers would not shop at a site if they were not confident in its security capabilities[5]. Also, 36% of online shoppers consider security considerations as the primary reason for choosing a brick and mortar store over an online equivalent[5]. This leads to the observation that web application developers are not really delivering a product of good quality when they concentrate on functionality, but rather they deliver one of perceived quality. The lack of focus on security throughout a web application's development life cycle often leads to a vulnerable first release of that application[2]. Considering potential customers' security awareness, this may prove to be a costly mistake.

Through research in psychology, it has been established that humans are able to handle large quantities of information much more efficiently if they are able to classify it into a manageable number of categories[15]. In light of this, the authors of this paper argue that managing thousands of security threats would be far easier if each threat could be seen within the context of a category of similar threats. Once a threat has been placed in this context, it is far easier to manage it by applying the same treatment that would be applied to similar threat albeit slightly tailored to any particular characteristics of the individual threat.

In this paper, it is being argued that there is a need for

a security ontology of online security threats. The authors also go on to propose such an ontology.

2. Overview of the proposed ontology

At its highest level, the ontology consists of 6 components as shown in figure 1.

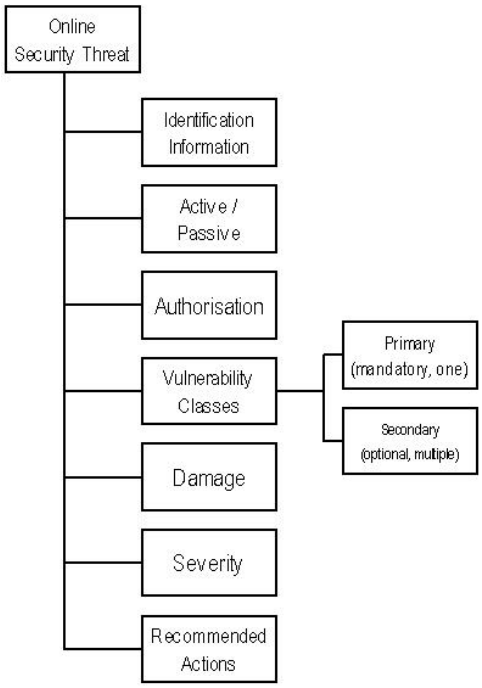

Figure 1. A high-level overview of the proposed Ontology

Each of these components is discussed in the following sections.

2.1 Identification Information

Every security threat needs to be identified. The *identification information* component of the proposed ontology provides for this. It is being proposed that threats be given a unique identifier (for use in database storage of information), a name, and a freetext description. This is depicted in figure .

2.2 Active/Passive Threat

The second component of the proposed ontology seeks to indicate whether a threat is *active* or *passive*. An active threat is in essence an intentional threat by someone with malicious intentions. This may include a hacker intentionally trying to gain access to credit card information, a virus trying to corrupt your operating environment, and so on. On the other hand, a passive threat is unintentional. That is to say it happened by accident. Instances of unintentional threats may include dropping a coffee mug over a server, maintenance personnel disabling power lines to essential equipment, and so on.

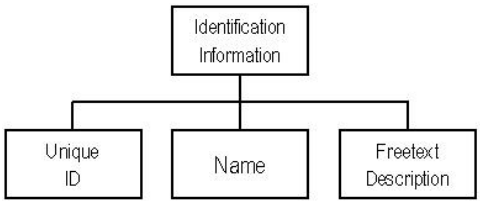

Figure 2. The *Identification Information* Component

2.3 Authorisation

The authorisation component seeks to identify what type of user would carry out a particular attack. Three possibilities exist and these are shown in figure **??**. Firstly, a user could be *authorised*. This means that a user has been delegated rights in the environment, on a system, or on a network and chooses to abuse of these rights in carrying out an attack. The second possibility is that of an *unauthorised* users gaining access to the environment through some weakness and thus causing havoc on the system. Finally, there is also the possibility of an *impersonation*. An impersonator could be a user, hacker or even a computer program which manages to gain access by taking on the identity or rights of another authorised user.

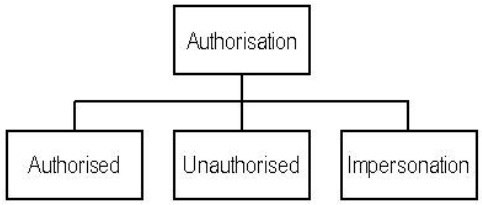

Figure 3. The *Authorisation* Component

2.4 Vulnerability Classes

The term *vulnerability class* refers to one of five classes which have been defined in this ontology as broadly covering all possible vulnerabilities which a security threat can exploit. shown in the figure 4. As a minimum, a threat will have a *primary vulnerability class* assigned to it. This refers to the vulnerability which is most exploited by the threat. Should a threat exploit more than one vulnerability, there is the option to define additional *secondary vulnerability classes*.

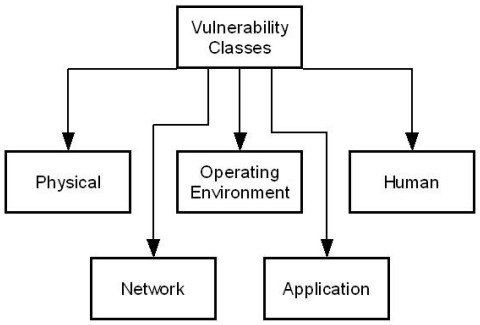

Figure 4. Types Security Threats

A web application may be compromised by threats which exploit vulnerabilities relating to any one of the categories shown in figure 4. All too often, one may be lulled into a false sense of security by emphasising security against threats on one of the above-mentioned levels. For example, one could feel that using secure communications[12] is enough to ensure a particular application's security. However, secure communications only assure privacy at the network level and a private web conversation with a hacker will not deter him/her from exploiting vulnerabilities of a different nature. The following is a brief discussion of the threat categories (sometimes referred to as levels) shown in figure 4.

At the ***physical level***, one must ensure that the locations where hardware (be it for deployment or development purposes) resides are physically secured[16]. Any physical intrusion into such a location may result in interruption of service, stolen data, and so on. In such cases, intruders may not necessarily be outsiders but could also be insiders (e.g. disgruntled employees).

Another category of security threats involves exploiting vulnerabilities at the ***network level***. Such threats attempt to manipulate shortcomings in communication protocols to disrupt service and/or gain access to private information. Appropriate actions must be taken to prevent this from happening[11].

The ***operating environment*** where a web application is hosted is also vulnerable to attacks. Vulnerabilities may be present in any components ranging from the operating system [17] to any other web-enabled component such as web server software, mail server software, and so on[18]. Again, one can easily imagine a scenario where an application is highly secured by developers, only to be breached by attacking its operating system instead of the application itself.

The category of ***application level*** threats refers to threats which exploit vulnerabilities within the web application itself. SQL injection attacks, cross-site scripting and authentication related attacks are typical examples of threats which can be found at this level[14][19]. It is beyond the scope of this paper to explore the technical merits of such attacks.

Finally, one should discuss vulnerabilities at the ***human level*** of security. Such vulnerabilities involve trusted human beings knowingly or unknowingly enabling outsiders to breach a web application and access restricted data[20]. Numerous incidents have been reported whereby a technically secure web application was breached because an insider was tricked into revealing information such as their username and password (a technique commonly known as social engineering). An effective security policy must ensure that trusted users are in a position to repel any attempts made to use them as a weak point within the application's security structure.

2.5 Damage

The *damage* component of the proposed ontology seeks to identify what sort of damage would be caused by the threat should it succeed in materialising. The following possibilities have been extracted from a US government report on information security risks[7]. Should new types of damage surface in future, these should be included in the ontology. A security threat can cause damage in one or more of the following categories:

- Disclosure of information
- Modification of information
- Destruction of information
- Degredation of service
- Denial of service
- Website defacement

2.6 Severity

The *severity* component gives an indication of the amount of harm and/or fallout which would result from the threat if it materialises. It is proposed that this information be categorised as *low*, *medium* or *high*. A *low* severity level indicates that the consequences of a materialisation of this threat would have little repercussion on the web applications ability to keep functioning and on its reputation. An example of this might involve a relatively harmless adware program managing to install itself on a server. A threat with *medium* severity is one that causes a degradation in service without completely disabling the site or one that causes damage or loss which can be recovered. Examples might include minor website defacements, or loss of data which is regularly backed up and can thus be retrieved. Finally, a *high* severity level indicates threats which severely impinge on a web application's stability and/or its users' privacy and security. Examples include hackers gaining access to clients' personal information, viruses corrupting or erasing data in such a way that it cannot be retrieved, an so on.

2.7 Recommended Actions

Finally, one should discuss the *recommended actions* component of the proposed ontology. This component has three sub-components as shown in figure 5.

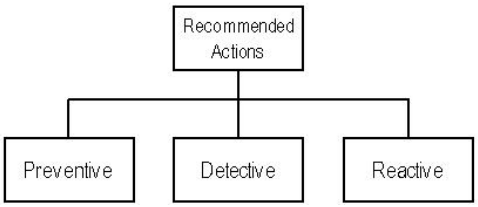

Figure 5. The *Recommended Actions* Component

The *preventive* sub-component refers to actions which can be taken to prevent the treat from materialising. A list of generic actions such as *deploy firewall solution*, *validate all application inputs*, etc is defined but due to length restrictions cannot be included in this paper. Since preventive measures may sometimes fail, the ontology also seeks to define a list of *detective* actions. These actions/mechanisms will be put in place to detect the occurrence of a security threat. Finally, the proposed ontology defines a number of possible *reactive* actions which are to be taken in the case of a materialisation of a particular security threat. Again, a number of generic actions have been defined but cannot be included here.

3. Conclusion and Future Work

It is believed that the ontology proposed here will prove to be very useful once a sufficient number of threats have been defined and place within it. Future work in this area will start off with this very task. However, the final aim of this line of work within the SEPI research group is to develop a multi-tier, multi-role security framework which will be developed. This framework will provide a process based way of securing web applications against security threats which are defined within the framework proposed in this paper.

4. References

[1] Glisson W. B., Welland, R., "Web Development Evolution: The Assimilation of Web Engineering Security", Proceedings of the 3rd Latin American Web Congress, IEEE Press, 2005.

[2] Gaur N., "Assessing the Security of Your Web Applications", Linux Journal, April 2000

[3] Morrisdale P.A., "The Six Dumbest Ideas in Computer Security", http://www.ranum.com/security/computer_security/editorials/dumb/

[4] Rakitin S. R., "Software Verification and Validation: A practitioner's Guide", Boston: Artech House, 1997

[5] Cachia E., Micallef M., "Towards Effectively Appraising Online Stores", Proceeding of the Software Engineering Knowledge Engineering (SEKE) Conference, 2004

[6] Cachia E., Micallef M., "Towards a RAD Framework for E-Commerce Systems", International Conference on Web Information Systems and Technologies, 2006

[7] United States General Accounting Office, "Information Security Risk Assessment - Practices of Leading Organisations", United States Government, 1999

[8] Deloitte, "2005 Global Security Survay", Deloitte Touche Tohmatsu, 2005

[9] Gordon L. A., Loeb M. P., et al, "2005 CSI/FBI Computer Crim Survey", Computer Security Institute, 2005

[10] Hersherb J., et al "Software quality and the Capability Maturity Model", Communications of the ACM, June 1997

[11] Mackey D., "Web Security for Network and System Administrators", Thomson Course Technology, 2003

[12] Freier A. O., Karlton P., Kocher P. C., "The SSL Protocol Version 3.0", Netscape, 1996

[13] Schneider B., "Secrets & Lies: Digital Security in a Networked World", John Wiley & Sons Inc, 2000

[14] "The Open Web Application Security Project", http://www.owasp.org/

[15] Antherton J. S. Learning and Teaching: Assimilation and Accommodation http://www.learningandteaching.info/learning/assimacc.htm ,2005

[16] Diroff T.E., "The protection of computer facilities and equipment: physical security", ACM SIGMIS Database, ACM, 1978

[17] Blakely R., "Hackers Uncover Biggest Microsoft Vulnerability", TimeOnline, http://business.timesonline.co.uk/article/0,,9075-1968021,00.html, 2006

[18] Nagel B., "Microsoft Releases Out-of-Cycle Patch for VML Flaw", http://www.redmondmag.com/news/article.asp?EditorialsID=782 2006

[19] Jovanovic N., Christopher K., Engin K., "Precise Alias Analysis for Static Detection of Web Application Vulnerabilities", Proceedings of the ACM SIGPLAN Workshop on Programming Languages and Analysis for Security, 2006

[20] Orgill G. et al, “The Urgency for Effective User Privacy-education to Counter Social Engineering Attacks on Secure Computer Systems”, Proceedings of the 5th conference on Information technology education , ACM Press, 2004

Forecasting Software for Chaotic Deterministic Systems using Dynamical Systems Theory

Michel Camilleri, B.Sc. M.Sc.
University of Malta
00356-25907295
michel.camilleri@um.edu.mt1st

ABSTRACT

Forecasting is a very important tool in all aspects of human activity ranging from agriculture to medicine and from welfare to high finance. Certain processes, especially those arising from natural phenomena or human activity may be intrinsically **deterministic** but exhibit **chaotic** behaviour. This makes them difficult to model and forecast. A set of nonlinear time series statistical techniques based on **dynamical systems theory**[Kantz & Schreiber 97] may be used to make forecasts on such processes by exploiting relationships found in the measurements without having to create an explicit model. These techniques are also wholly dependent on computer automation because of the relatively higher volumes of data and computations. Although developments in computer systems and the increased availability of large volumes of data have increased the potential use of these techniques, there is still a relative lacuna in software supporting these techniques. A **software application** [Camilleri 06**]** was developed to promote the use of these techniques and help fill this lacuna through an approach which is different from the few other existing applications. The system was validated by detailed testing on **sunspot activity data** [SIDC 05] and compared favourably to those obtained in separate published studies [Tong 90] [Kulkarni et al 98] using auto regression techniques and artificial neural networks.

Categories and Subject Descriptors

G.3 **[PROBABILITY AND STATISTICS]** Nonparametric Statistics, Statistical Software, Time Series Analysis

H.4.2 [**TYPES OF (INFORMATION) SYSTEMS**] Decision Support

I.5.3 **[CLUSTERING]** Similarity Measures

J.2 [**PHYSICAL SCIENCES AND ENGINEERING**] Mathematics and Statistics

General Terms

Algorithms, Management, Measurement,

Keywords

Forecasting, Nonlinear , Chaos, Determinate Systems, Dynamical Systems Theory, Phase Space Embedding, Delay Vectors,

1. STATISTICAL METHODS FOR FORECASTING

Forecasting is an important tool in the hands of decision takers. Forecasting methods are usually based on determining a model of visible part of the phenomenon as it varies over time or a model of the underlying processes which contribute to the phenomenon. This model or set of models, once they are gradually refined until they can be considered as representative of the process under study. They can then be used to project the process into the future and generate forecasts.

Many phenomena are dependent on processes which are determinate in nature and lend themselves to such modeling and forecasts can be generated with varying degrees of success. The degree of accuracy depends on the predictability inherent in the underlying processes, the analytic techniques applied and the technology available. Statistical methods have long supplied various forecasting methods for both linear and non-linear processes. Several statistical techniques create models which represent the process in all the time steps e.g. autoregression, or different models for different phases of its evolution e.g. threshold autoregression. [Tong 90]

1.1 Chaotic Deterministic Systems

However certain nonlinear phenomena exhibit behaviour which is too chaotic for modelling and forecasting. These phenomena may be inherently deterministic but due to a large number of variables or noise in measurement they appear chaotic.

1.2 Dynamical Systems Theory

A set of techniques have been developed from the theory of dynamical systems [Kantz & Schreiber 97] have indicated potential in solving some of these data sets. The theory behind

these techniques is that although a chaotic process evolves over time in a manner which makes modelling difficult, it is still predictable in the initial time steps. It is possible to exploit this window of time to generate useful forecasts using data from prior time steps. It has also been shown that the way such a process will evolve beyond the current time step is dependent on the current state and a set of states prior to current one. In a cyclic process there may be several previous occurences where the process was developing in the same manner as the current point in time. The basis of the forecast is in finding these similar prior evolutions and using the data about how they evolved in order to create a 'projection' into the future from the current time point.

1.3 Almost Model-Free Techniques

A subset of the techniques based on dynamical systems theory do away with the need to create a model of the process [Kantz & Schreiber 97]. Instead, the database of actual measurements is used as a basis for measuring predictability and for forecasting. This requires a higher order of computation and computing power. It also requires the determination of a set of parameters which define the number of measurements needed to compare past evolutions to the current state. A set of measurements related in time to the current point in time is identified and called a delay vector. The **delay vector** is compared to the database of past evolutions using one of a number of similarity techniques in order to find similar instances

1.4 Phase Space Embedding

The process of finding similar evolutions in the past is made faster by translating the whole database of measurements into phase space i.e. the change occurring to the process is determined at each time step. Once this has been done the search for evolutions of the process similar to the curent point in time is rendered much faster. This is called finding **neighbours** in phase space.

1.5 Finding the optimal dimension number and time delay

The construction of the delay vector is dependant on the number of elements or **dimensions** spaced by a time distance called the **time delay**. The dimension number and the time delay have a determining effect on the finding of neighbours in phase space (similar vectors in the database) and ultimately on the accuracy of the forecast. A number of techniques can be used to help find the **optimal values** for these parameters.

2. FORECASTING APPLICATION USING AMFPSE

Although developments in computer systems and the increased availability of large volumes of data have increased the potential use of these techniques, there is still too little software supporting these techniques[Hegger et al. 99] [Belaire-Franch 02]. A software framework was developed [Camilleri 06] with the aim of

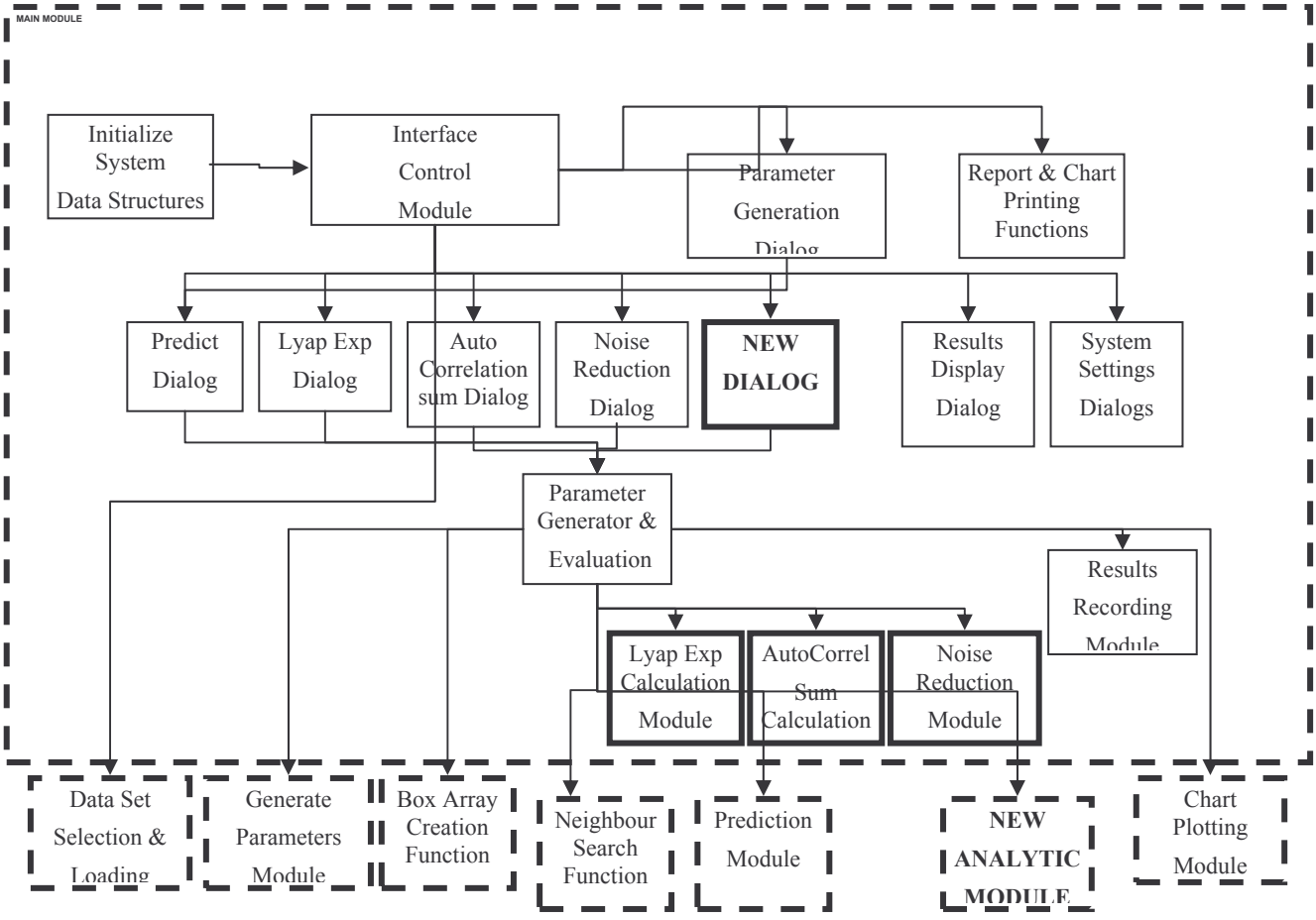

Figure-1 Component Diagram of Main Module (detail), Internal and External Modules

implementing a subset of these techniques in a re-usable set of modules, with a graphical interface and other features useful to the analyst and the potential software developer. The software can be used as a standalone package, providing the data analyst with the tools required to run a forecasting session. The software may be extended with additional analytic modules. The potential also exists to embed the software (as a whole or in part) in other applications e.g. a demand planning system, in order to enhance the forecasting capability.

The software framework consists of a **number of modules** for the common tasks required for almost model-free forecasting using phase space embedding (AMFPSE). It incorporates the various analytic functions as well as a number of support **modules** into a **single application** that works as an integrated system to support the analytic process. A central or **main module** defines and initializes the main data structures required for the various operations. This main module interacts with the various modules by passing information via parameters as actual values or pointers to shared data structures. The other modules include the selection and loading of the data set into memory, transformation into phase space, searching for neighbours in phase space, forecasting, plotting of forecast data.

3. FORECASTING USING THE SOFTWARE

This software was tested [Camilleri 06] on a number of large volume data sets representing diverse phenomena showing nonlinear behavior. The data sets varied in predictability, and applicability of the software.

3.1 Using Software to Forecast Sunspot activity

The data set of greatest interest was the one which represented the amount of sunspot activity measured over a period of 250 years Figure -2. It is known that this phenomenon exhibits a 10-11 year cycle, however the amplitudes of the various cycles varies.

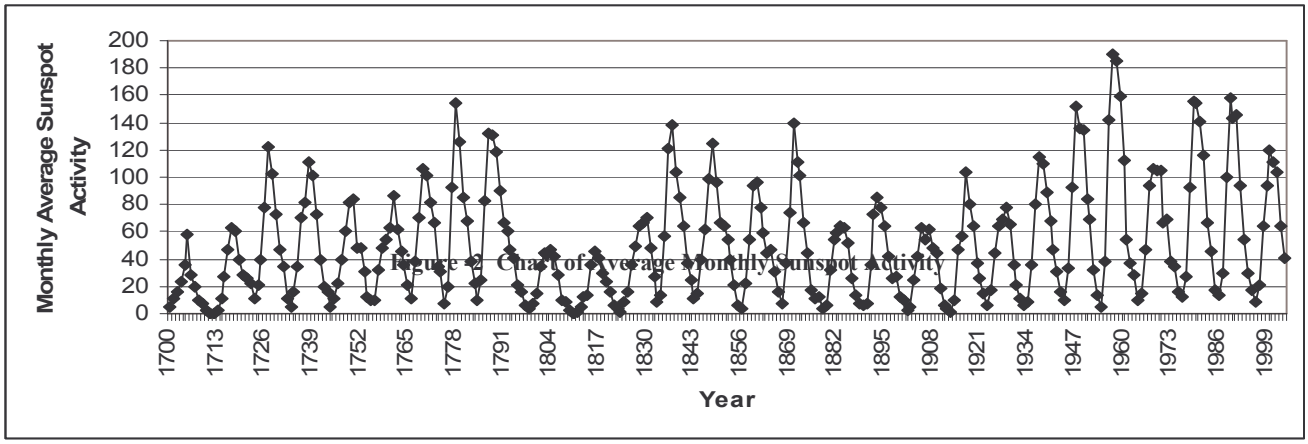

Figure -2 Chart of Average Monthly Sunspot Activity

A set of forecasts using the software framework were carried out on monthly average sunspot activity data for two periods 1980-1987 and 1997-2004. The results were compared with the results available in two separate published texts namely [Tong 90] and [Kulkarni et al. 98].

[1][Tong 90], documents a set of forecasts for 1980 – 1987 using two different techniques i.e. linear autoregression (AR) by Subba Rao T. and Gabr M.M. 1984 and Self Exciting Threshold Autoregression (SETAR) by Tong H. and Lim K.S. 1981. The results of another forecast using a combined technique (AR and SETAR) by Tong et al. and another using SETAR by Ghaddar T.M. and Tong H. 1981 were also available in the same text. The second study [Kulkarni et al. 98], describes a set of forecasts for 1997-2004 based on artificial neural networks.

3.2 Years of Data used for the training set

The years of sunspot data used in the studies described in [Tong 90] varied because they were carried out at different points in time. Ideally one would compare results using the same basis years each time. However this would have entailed several repeat analyses. Such extensive testing was beyond the scope of this study. The validation of the software framework using these published data sets was not an attempt to outdo these results but to show that the software is adaptable to different domains and can achieve results which are at least comparable to those obtained using other techniques. The years used for the training sets in the second study [Kulkarni et al. 98] were the same as those used by the author with the software framework.

3.3 Optimization of Parameters using 'hidden' sets

In order to simulate real-life conditions, the forecasts using the software framework did not include the years being forecast in the training data during the parameter optimization process. Moreover a further set of hidden data was used to optimize the parameter set. This meant that the sunspot data set was separated into three sets each time:

- the training set used for optimizing the parameters
- the hidden set used for optimizing the parameters
- the 'future' set used for measuring the accuracy of the final forecast.

The initial training set for the 1980-1987 forecast was 1850-1974 and the parameters were optimized using 1975-1979 as the hidden set. This means that there was no information from the forecast years 1980-1987 which could bias the 'optimization' process. The initial training set for the 1997-2004 forecast was 1850-1991

and the parameters were optimized using 1992-1996 as the hidden set. The final forecast for 1997-2004 using the optimal or near optimal parameters used 1850-1996 in the training set.

3.4 Accuracy of final forecast

The final forecast for 1980-87 used the optimal or near optimal parameters obtained by training on the 1974-1979 hidden set. This time the 1850-1979 data was used in the training set. The forecast set was compared to the 1980-1987 data and the prediction error was calculated using RMSE and MSE. Similarily the final forecast for 1997-2004 used the optimal or near optimal parameters obtained by training on the 1992-1996 hidden set. The 1850-1996 was used in the training set. The forecast set was compared to the 1997-2004 data and the prediction error was calculated using RMSE and MSE.

3.5 Comparison of results of sunspot activity analyses

The following are the results of the various studies compared with those obtained by the software framework [Camilleri 06]. The prediction error is measured in both RMSE and MSE as the two sets of studies used either RMSE or MSE. In order to make it easier for referral to these studies both measures are included. Table -1 compares the results of the 1980-1987 forecasts in [Tong 90] with those obtained using the software framework.

Table -1 Comparison with sunspot predictions using autoregression – adapted from [Tong 90]

Study	Technique	Years for optimization	MSE	RMSE
Subba Rao T. and Gabr M.M. 1984	Linear Autoregression	1700-1920	222.4	14.9
Tong H. and Lim K.S. 1981	Self Exciting Threshold AutoRegressive	1700-1920	253.9	15.9
Tong et al	COMBINED (AR + SETAR)		237.9	15.4
This study	AMFPSE	1850-1974	227.2	15.1
Ghaddar T.M. and Tong H. 1981	SETAR 1980	1700-1979	85.5	9.2
	following modification (with bias correction)		50.5	7.1
	(without bias correction)		55.5	7.4

Here, the framework compares well with the studies based on the older data sets i.e. 1700-1920 but is outdone by the more recent studies using data up to 1979. However a look at Figure-3[1] shows how close the forecast set is to the actual set.

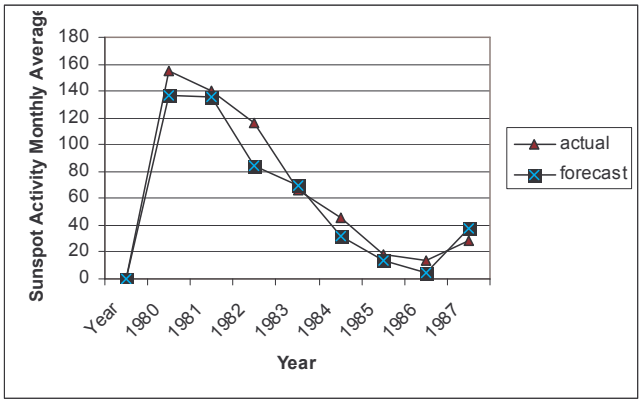

Figure-3 Forecast of Sunspot Numbers for 1980 – 1987

The software framework [Camilleri 06] was applied to forecast the 1997-2004 period in order to compare with the results obtained in the [Kulkarni et al. 98] study using the artificial neural network. This time the process of optimization gave five parameter sets with optimal performance when trained on the 'hidden' set. When used individually for the actual forecast their performance varied.

Presuming that the future values are not known the author's solution was to average the forecasts obtained [Camilleri 06]. The result of the averaged forecasts (see Table-2) was in fact better than all the other results i.e. it yielded a RMSE of 9.0.

Table-2 Comparison with Neural Network [Kulkarni et al. 98]

Study	Technique	Years used for Fitting	MSE	RMSE
Kulkarni et Al 1998	Artificial Neural Network	1850-1997	323.35	17.98
This Study	AMFPSE	1850-1997	81.5	9.0

Figure-4 [1]shows visually that the behaviour of the actual sunspot activity is very closely reproduced.by the software framework. The result may have improved over the previous study possibly due to:

- the averaging of forecasts of the optimal parameter sets
- the inclusion of more years data
- the fact that predictiability improves over certain parts of the cycle.

[1] Chart was obtained using a spreadsheet application

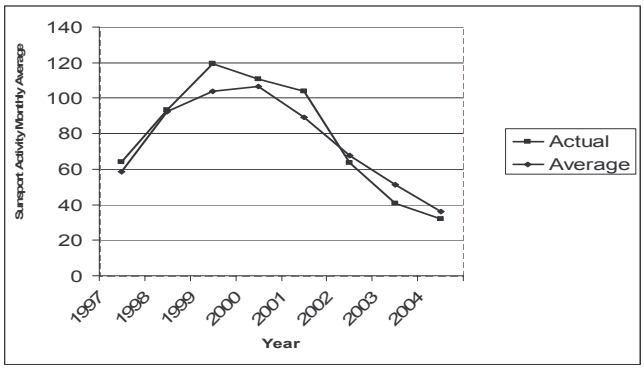

Figure-4 Forecast of Sunspot Activity 1997-2004

The RMSE and MSE figures both give an indication of how close the forecasts are, however the charts demonstrate this much more clearly.

3.6 Forecast over a more extended period (25 years)

The same parameters and data set used for the 1980-1987 forecasts were then used to forecast a longer span of 25 years i.e. from 1980 – 2004. This yielded an MSE of 336 and a RMSE of 18.3 (see below and Appendix 18 for the list of forecast values). The graph shown in Figure-5[2] (obtained from the software framework) shows clearly how a fair degree of accuracy is maintained even over a more extended period. The 25 year subset shows clear differences from cycle to cycle which adds to the merit of these techniques.

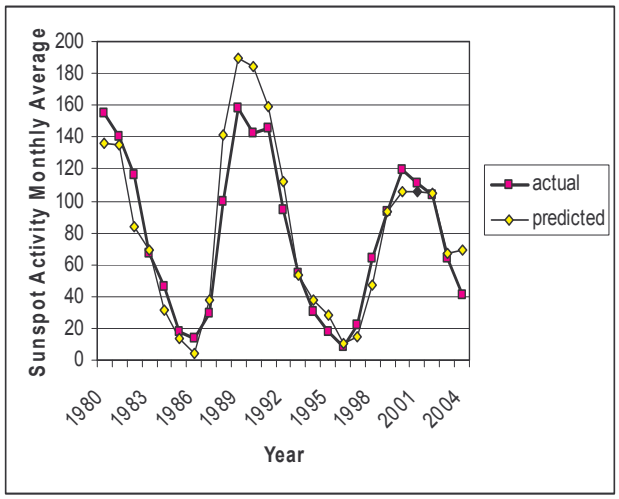

Figure 5 Sunspot numbers forecast for 1980-2004 compared with actual

[2] Chart was obtained using a spreadsheet application

4. CONCLUSION

The results obtained compared favourably with the best results obtained in the studies using other techniques. This demonstrated that these techniques and the software developed could be applied to create accurate forecasts on a time series data with an underlying deterministic nature but exhibiting chaotic behaviour. One of the key factors was the number of cycles over which the process was sampled (22 in this case).

5. ACKNOWLEDGMENTS

My thanks go to Prof. Holger Kantz and Mr. Thomas Schreiber for the algorithms and background and R.A.M. Van der Linden and the SIDC team for the sunspot data

6. REFERENCES

[Belaire-Franch 02] Belaire-Franch, Jorge and Contreras Dulce, Recurrence Plots in Nonlinear Times Series Analysis: Free Software, *Journal of Statistical Software* 7 Issue 9, 2002

[Camilleri 06] Camilleri M. *A Software Framework for Forecasting using Almost Model-free Techniques with Phase Space Embedding (AMFPSE) in Non Linear Time Series Analysis, MSc. Thesis, University of Malta 2006*

[Hegger et al. 99] Hegger R., Kantz H., and Schreiber T., Practical Implementation of nonlinear time series methods: The TISEAN package, CHAOS 9, 413-435 ,*Winter School on Nonlinear Time Series Analysis (TISEAN 98) Dresden Feb 1998 (1999)*

[Kantz & Schreiber 97] Kantz H. and Schreiber T., "Nonlinear Time Series Analysis*", Cambridge University Press, Cambridge, (1997)*

[Kantz 01] Kantz Holger, Time Series Analysis in Reconstructed State Spaces, *Stochastics and Dynamics, Vol 1, No. 1 (2001), 85 111, World Scientific Publishing Company, 2001*

[Kulkarni et al. 97] Kulkarni D.R., Pandya A.S. and Parikh J.C., Dynamic Predictions from Time Series Data – An Artificial Neural Network Approach , *Int. Journ. Mod. Phys.*C8,1345(1997).

[Kulkarni et al. 98] Kulkarni D. R., A. S. Pandya, and J. C. Parikh, Modeling and Predicting Sunspot Activity – State Space Reconstruction + Artificial Neural Network Methods *GeoPhys Lett. 25, 457,1998*

[SIDC 05] SIDC, RWC Belgium, World Data Center for the Sunspot Index, Royal Observatory of Belgium, 1850-2005

[Tong 90] Tong Howell, Non-Linear Time Series – A Dynamical System Approach, *Oxford University Press, Oxford UK*, ISBN 0-19-852300-9

Of Web Trust and Policies: A suggested framework to enhance Internet Security

Steven Caruana
Department of Computer Science
Univeristy of Malta
scar009@um.edu.mt

Dr. Matthew Montebello
Department of Computer Science
Univeristy of Malta
mmont@cs.um.edu.mt

ABSTRACT

We tend to trust people, software or anything else around us through experience or through a recommendation from a trusted source. Web voyeurs have similarly envisaged the notion of software entities roaming over the World-Wide Web capable of trusting other similar entities. Ideally, Web agents would be able to distinguish and differentiate between sites, services and resources over the Internet that are reliable and worth the confidence accredited to them. In this paper we present several trust and policy frameworks built within the evolving agent mark-up languages in an attempt to encapsulate the Web in a new dimension of trust. Furthermore, we propose a novel mechanism that exploits existent policies, that govern servers and provide them with credentials responsible for their authentication, by extending the existent web site structure.

Keywords

Agents, Semantic Web, Trust, Security, Policies

1. INTRODUCTION

Trust is one word which is important to the greater part of humanity for whatever the scope, reason or context one applies it. Most often business, industry, academics and researchers place information and the divulgation of information as the centre point of all of their activities. This means that people for whom the evolution of means of communication and dissemination of information plays a central role for their operations and accomplishments, trust gains more significance. With the onset of more complex and demanding e-services in various fields, the concerns of security has become a reason for which many a brow will furrow. In a recent talk during a World-Wide Web Consortium [16] meeting in London, Sir Tim Berners Lee pointed out that People learn to trust through experience and through recommendation, and argues that a Web of trust would be not only safe for people to use and access but also for software which needs reliability and dependability when servicing user requests. The notion of having trustworthy web services such as active networks, mobile agents, etc. which thrive upon entire distributed networks of information and computation resources, is becoming more challenging as requests are becoming more complex and less trivial. The rest of the paper is organised as follows: In Section 2 we review how trust over the Web is currently being achieved through the use of a number of policy frameworks. In Section 3 we propose a model for building trust over the Semantic Web through the use of entities and adapting policies to govern those entities. The model described in Section 3 aims to empower users with creating their own policy frameworks, merging them with distributed authenticated systems which are essential components for achieving trust and ensuring security over the web. The last section of the paper describes the future implementation of the application and the extent of research needed in the field of trust and security over the World Wide Web.

2. TRUST OVER THE SEMANTIC WEB

The current web has no notion of entities or relationships between entities. Users browse the Internet finding information in the same document format, style and layout as they are used to handling. On the other hand, software agents whose role is to handle information and service requests by the users, have no means of parsing information from the traditional layout. For these agents to be able to crawl the web and service user requests, there needs to be defined a service-oriented architectural structure to represent data on the internet and the relations that different clusters of information display. The Semantic Web is a possible solution for this problem because it deals with the representation and aggregation of relationships between resources on the web.

One of the more important factors that tend to dissuade the community from the adoption of Semantic Web Applications is the notion of trust. At this present stage in the development of the Semantic Web we have produced languages (DAML + OIL, OWL[18] etc.) which are capable of representing information on the internet. Despite the richness of these languages we are still not able to describe relations like "I want to trust only papers written by professors who are still registered with a university and these papers must have been published less than five years ago", or "Do not give information to users unless they can clear their identity as members of the university I am registered with".

2.1 Introducing Trust

Research projects use the term trust to signify different meanings. The following is one such definition:

> "Trust is a method of dealing with uncertainty; when dealing with independent agents, institutions or providers of resources (including knowledge), one trusts them if one accepts their characterisation of what they will do. Trust can be a moral notion (X trusts Y to act in Xs interests), or not (X trusts Y to perform some task T)." [12]

If one were to apply the above statement to the Semantic Web it would mean that a user (human or agent) has the capacity to trust other users, but that not every user can be trusted. In this paper we are defining trust as being either of two issues. The first is the issue of authenticating a person as being whom they claim to be. Trust can also be defined as being a measure of the degree of trust that a user attributes to another user in the context of a particular domain.

Most of these problems are not new to the internet. Projects such as the Microsoft .Net Passport[11] and OpenID[13] have provided solutions for possible distributed authentication infrastructures. On the other hand projects such as Foaf[6] have provided us with a means of adding our credentials in a machine readable format across the web.

2.2 Trust and Policy Frameworks

When the Semantic Web was first introduced in [1] one of the ideas which was put forward was that of having a web of correlated information from which software agents could acquire knowledge which they would eventually use to help the user for every day life service requests. In turn research projects have been making progress in trying to find solutions for various issues that this vision put forward. As a result of this, trust frameworks were developed by research groups and are now being used by developers researching the field of semantics in web applications.

The following section will give a brief description of few such projects.

2.2.1 Ponder

Ponder[3] is a distributed policy management system that was developed a few years back. This system was designed to allow users across the web to specify information (policies) about resources which would otherwise be impossible to define and enforce.

These specifications were recorded using what the authors referred to as policies. Ponder supports a number of policy specifications, including *Authorisation* policies, *Obligation* policies, *Refrain* policies, *Delegation* policies, *Composite* policies and *Meta-policies*. Each of these policies can have *constraints* applied to them. These policies are then applied to specific domains for which Ponder provides the authentication and trust reasoning engine.

2.2.2 P3P

P3P[17] is a project organised by the W3C[16] which is trying to provide a specification for an extension to the current web. This initiative aims at enabling web sites to store information about privacy policies which should be applied to them. This information is meant to be analysed and processed by agents that can use this information in the background on the user's behalf, while he/she is browsing other web pages. Even though this information is meant to be used by agent software, this standard also caters for human users by presenting information in a reader-friendly way. Having agents process privacy preferences on behalf of the user will conveniently take the load off the user, relieving him/her of having to acknowledge every privacy policy for each individual page.

Numerious implementations of this standard have already been presented and can be found at the W3C web site[16].

2.2.3 KAoS

KAoS[2] is an open agent system that offers possible solutions to some of the problems that adversely affect agent architectures. In [15] the author writes about how KAoS policies can now be defined in OWL and gives a good description of how KAoS proposes to use a policy ontology termed the KAoS Policy Ontology (KPO). The KPO is then used as a base ontology to define ontologies that can represent statements like:

> It is forbidden for employees of a company X or employees of a company Y to apply for this package.

The system that employs this policy can then use it to determine who is eligible to apply for the package in question and who is not. At runtime the system will then interact with users and once each user registers with the services, the site will be able to formulate who are employees of companies X and Y and determine what they can apply for.

KAoS can be said to be more agent oriented than others because it was designed with agents in mind rather than web users. It aims to provide agents (KAoS agents) with a roaming space intended for authenticating users when required and providing other services which the user might benefit from.

2.2.4 Rei

Rei[7] is another attempt to create a policy language. It provides an ontology which is used to define the policies in its engine. In the Me-Centric project[7], the author proposes a system whereby the use of policies defines a global perspective which is made up of domains and sub domains that can overlap each other. Similar to other projects, this project aims at making this web accessible not only to users but also to software agents.

The first problem that Rei tries to tackle is the fact that users who are not very technical find it hard to understand most policy languages. Therefore Rei is based on First-Order Logic (FOL), which is not only easy for users to read but is also simple to translate to RDF or DAML + OIL. This

means that FOL language can be translated into a semantic language representation.

This system also provides a set of ontologies which are not domain specific so that whoever is writing his/her own policies can use these ready made concepts to create varied operations and functions such as setting of permissions, obligations, speech acts etc. It might be the case that a single domain might require more specific objects to be defined (such as person, readFile, deleteUser etc.) and properties associated with them (name, age, comany, email), which is why Rei also supports the extension and definitions of its policy ontologies by the user.

The Me-Centric policy server stores all the policies that determine how entities are to be treated, while its domain server contains the information pertaining to various domains. The policy server can retrieve the policies of different domains and then use the domain server to map the domain specific names to the policies. When a user requests a speech act, the policy engine will determine the logic of this speech act and then will add the information it gathers to its knowledge base. These speech acts can be categorised in four different policies, *delegate*, *revoke*, *cancel* and *request*.

2.2.5 Rein

Rein[8] is a project that tries to extend on Rei by adding support for N3 logic reasoning. The target audience of Rein is the internet in general and it proposes a few twists to the structures seen so far. All the policy languages reviewed so far have defined their own structures (policy languages) in which policies should be defined (policy files, KAoS ontology, Rei Onotlogy etc.).

This methodology proposes the use of Rei ontologies to define the policies, but unlike other projects, it does not place strict rules to enforce which language is used to define these policies. It promotes the use of different policy languages and refers to policies written in these languages as Meta-Policies. It also takes advantage of various features of the Semantic Web, such as allowing an entity to be categorised by another entity, according to its definition. In such an eventuality the Rein engine will go to the desired URIs and collect the necessary information to complete the definition of the entity.

[9] gives the reader a good idea of how Rein can be used to deploy a domain with varied implementations of Rein and a variety of policies residing on each node. It also gives the reader a good insight as to how Rein ontology is defined and how the Rein engine works. [10] also discusses issues related to the implementation of Rein and adds a few more examples which provide a useful insight as to how a network using Rein policies should be constructed.

3. FROM WEB OF TRUST TO A SEMANTIC WEB OF TRUST

When one browses the web, he/she will encounter a number of pages which encapsulate certain unique features. Some sites propose mechanisms for the sharing and annotation of information (e.g. Flickr[5], Riya[14] etc.) or even community voting which can determine the trust level of a resource or user (e.g. ebay[4] etc.). In this paper we are proposing the construction of a mechanism which extends the current website structure by providing a direct feed to the user's machine providing a diagrammatic representation of the policies that govern the currently viewed website. It is also a means of informing the client's machine about how to provide its user's credentials to the servers that are responsible for authentication. We are also proposing the extension of a typical web server allowing it to make use of a policy language such as Rein to define policies that will govern the specified domain and to provide a means of allowing the server to use distributed authentication.

Below is a diagram of how such a network would be setup and of how the user is expected to interact with the system.

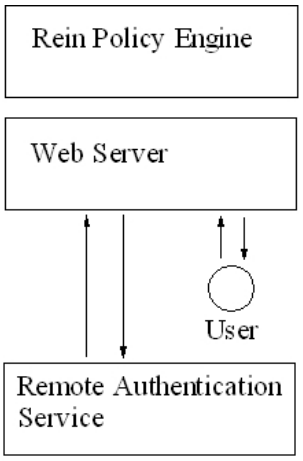

Figure 1: Network Setup

In the diagram above one can see all the major entities that partake in this system. At the back-end there is a server that will await requests from the client. Once the server receives a request it can make use of remote servers to authenticate the client and send an adequate response in return.

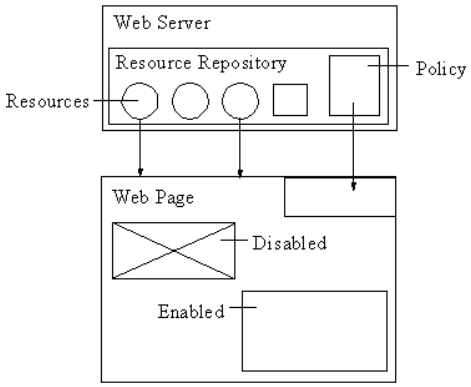

Figure 2: Document transfer

In Figure 2 the browser will display only items for which the user has credentials. Items for which no credentials are given will be automatically blocked. It also shows how the user can send his credentials to provide authentication for the blocked items.

This setup is quite often adopted by those who develop distributed authentication systems. What distinguishes this setup and makes it unique is the way the client and server communication is handled. When the client first requests information from a page the server will return and display the requested data. Along with this, the user will also receive a 'feed' that will inform the browser that the user was not authorised to view certain components because he/she did not fulfil the necessary policy requirements. The user will then be prompted by the browser to allow the use of credentials which would have already been inputted into the system in order to fulfil these requirements. Once this structure is in place, the system will know that this user can be trusted and that his/her credentials can be reused at a later stage if required.

The above proposition could offer a solution in terms of the server which would otherwise have to decide whether the user can be trusted or not. However this still does not solve the problem of how the user could gain the trust of other users viewing pages submitted by him/her. As a solution to this problem, we are proposing that the servers store the credentials within the policy feed attributed to this page, and that agents on the client-side will be able to process this information and make use of rules that the user would have defined beforehand to determine whether a site could be trustworthy or not.

Below is a typical example of how such a system could be used:

A user might log into his/her account on the University of Malta web server. His/her agent submits his credentials to the web server and this in turn keeps track of them. The user then decides to upload a paper about his main area of research. This paper is then forwarded to a board who is assigned to approve it. Once the document is approved, the user's agent returns to the site and updates the credentials which are related to the paper and the policies governing it, thus including the fact that the document was accepted and published.

To extend this example let us consider the added circumstance during which, three other uses are roaming the net looking for papers in this same research area. Once the page is accessed by all three search agents, the privacy field embedded in the page will interoperate with these agents. User agent 'A' has a policy of not showing unpublished papers, whilst 'B' and 'C' lack this policy. When 'A' identifies that this paper has not yet been published it refuses to view the paper and informs the user that an unpublished paper was refused. User agents 'B' and 'C' try to access the paper but the paper requires that to view the paper, the users *must* be on the auditing board. User agent 'B' had been informed a priori that its owner would be reviewing this paper, and sends over the necessary credentials. On the other hand user agent 'C' lacks this credential and sends a message to the user's browser to ask the user to submit the necessary credentials if s/he intends to view this page. User 'C' then inputs an encrypted link to his board credentials and updates his public profile which his agent can then use to re submit the credentials required to view this site.

3.1 Of entities and policies

In this paper we are proposing that web pages and the Internet should have a means of publishing or of enforcing a set of policies. In the example provided in section 3 we can notice that there are various entities and each have their own policies.

The following is a table listing the policies applied in the example found in Section 3

Table 1: Table of entity policies

Entity	Policy
Web Page	Allow all
Paper	If user can be authenticated as being on the submission board
	(After paper approved) Allow all
User Agent A	Do not download publications which have not been approved
User Agent B	Allow all
User Agent C	Allow all

The following is a table listing the credentials found in the example found in Section 3

Table 2: Table of entity credentials

Entity	Credentials
Web Page	Credentials that point to author
Paper	Credentials that point to author
User Agent A	No Credentials in this example
User Agent B	Credentials to prove he/she is a board member
User Agent C	No Credentials
	(After inputting credentials) Credentials to prove he/she is a board member

3.2 Empowering users

An important factor which is to be considered is that it will be hard for such a system to be adopted unless the users are able to migrate their current security structures and perform maintenance on them more efficiently. To assist users in overcoming this problem we are proposing that a browser extension for web authors be developed and that this be used for both testing and updating of the policy files. This extension should provide two different modes of use which are *browse mode* and *edit mode*.

When the user is viewing the page in browse mode the browser should be searching for policy feeds that web pages might be broadcasting and offer the necessary credentials to them accordingly, whilst still offering the user a chance to

replace these credentials with more specialised ones. If the browser cannot find the necessary credentials in the user's profile, then a means of asking the user to input his authentication details should be provided.

In edit mode the browser should visualise the policy feed that the user is constructing, in a logical and intuitive structure. Making it easy to construct and view policy files is of great importance because it will be only the minority of users who are Semantic Web aware and capable of constructing complex policy strucutres manually. A visual aid, such as a graphical user interface, would help bridge this problem by providing the less technical users with an adequate mechanism enabling them avoid dealing with the back-end system whilst still being able to declare their own policies.

Using the structure discussed we would be empowering the less technical users (such as blog users or wiki eidtors) to declare their own policies and help them to implement a Semantic Web Policy Structure to limit access to the resources they publish on the internet. As for the users who are meant to access these web resources, they can use the simple interface provided by their browser to fulfill the requirements that the authors would have set and gain access to the services being broadcasted.

3.3 Distributed Authentication

Distributed authentication research projects have been undergoing development for a number of years. As a by-product of these research groups, products like OpenID[13], Microsoft .Net Passport[11] were developed.

Both of the projects mentioned above aim at offering the user a single sign-on structure. The .Net Passport framework provides a number of Microsoft-owned servers on which each user has an account. Using this infrastructure Microsoft are pushing forward the idea that a user should have the facility for logging into a website (e.g. logging into hotmail) and not needing to log into any other page including ones in separate domains. OpenID tries to push forward a similar concept by offering an infrastructure where the user need only have an account on one domain and should use this account to authenticate himself accross the web.

These systems aim at providing solutions to the problems of authentication and trust. However not one of them provides a solution which could solve both of these problems. The .Net Passport does not offer a real solution for trust problems because even though each user should be assigned a single user login on the internet, it promotes no means of understanding the user's profile. For example if a user has a .Net Passport he/she can login using that passport and roam about on the internet using this credential, but if the user is publishing a paper to a website and this user has already submitted a number of publications before, the .Net Passport has no means of relating these publications.

OpenID on the other hand offers a structure that is not only a single login system but provides an infrastructure that can be used to identify a user as being a member of a certain domain for example, if one decides to submit a blog post on a blog hosted at www.myBlog.com, but his/her current blogging account is found at www.hisBlog.com, using OpenID the user can be allowed to log into hisBlog.com account and post a comment on someone else's www.myBlog.com account. As the system logs the user and authenticates his credentials it is also giving access to the site's security gateway. OpenID does not let you carry forth information about your account to use as credentials for trust algorithms. For example to make a distinction between two classes in a domain (e.g. a profressor of science and a student of science) using OpenID, the only workaround to represent this differnce is for the two classes to be assigned different domains to log into (e.g. student login student@student.cs.um.edu.mt, professor login professor@professor.cs.um.edu.mt)). Although this work around does provide a rudimentary means of attributing trust, it can become very hard to manage, and will only allow the representation of very basic differences between the classes of users and more complex relations (such as number of publications, issued or amount of time dedicated to students a week). If the user were to write the policy files manually with no visual tool they would be next to impossible to keep track of. Policies would need to be amended every time a change in a user's credentials occurs.

In this paper we are proposing a structure that can leverage the power of distributed login mechanisms with an added extension which integrates a trust framework into it. This would allow a user not only to authenticate himself/herself as being registered to a specific domain (like OpenID) but also to carry forward his/her credentials which can give him/her credibility over the web. Using Semantic Web technologies credentials can now be represented in a format that can be reasoned upon (such as rdf) and this can then be used as input for a trust engine (such as Rein) giving as a result the trustworthiness of a website or user.

4. FUTURE WORK

The application being described in this paper is currently still being developed. Further extensions to the mechanisms and structures mentioned above are still being refined.

In this paper we have also emphasised the concept that a ubiquitous security framework might not be the best way around providing security mechanisms. However more research is needed to devise a framework which could bridge the transfer of user credentials across domains using a predefined security standard, whilst still allowing the separate domains to make use their own means of authentication mechanisms.

Finally an idea which will need to be looked into further, is that of providing support for Semantic Web Services and Semantic Web Applications. As the web continues to progress, web services and web enabled applications are becoming ever more vital for the web and its users. Extending such a framework to support the use of policy feeds could provide these applications with a subsystem that caters for trust and authentication.

5. REFERENCES

[1] T. Berners-Lee, J. A. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[2] J. Bradshaw, S. Dutfield, P. Benoit, and J. Wooley.

Software Agents, chapter KAoS: Towards industrial strength open agent architecture, pages 375–418. MIT Press, 1997.

[3] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. Ponder: A Language for Specifying Security and Management Policies for Distributed Systems. Technical report, 2000.

[4] ebay. *http://www.ebay.com*.

[5] Flickr. *http://www.flicker.com*.

[6] FOAF. *http://www.foaf-project.org/*.

[7] L. Kagal. Rei: A policy language for the me-centric project. Hp labs technical report, hpl-2002-270, HP Labs, 2002.

[8] L. Kagal and T. Berners-Lee. Where policies meet rules in the semantic web. Technical report, MIT, 2005.

[9] L. Kagal, T. Berners-Lee, D. Connolly, and D. Weitzner. Self-describing delegation networks for the web. *IEEE Workshop on Policy for Distributed Systems and Networks (IEEE Policy)*, June 5 - 7 2006.

[10] L. Kagal, T. Berners-Lee, D. Connolly, and D. Weitzner. Using semantic web technologies for policy management on the web. *21st National Conference on Artificial Intelligence (AAAI)*, July 16 - 20 2006.

[11] Microsoft. *http://www.passport.net*.

[12] K. O'Hara, H. Alani, Y. Kalfoglou, , and N. Shadbolt. Trust strategies for the semantic web. In *Proceedings of the Trust, Security and Reputation workshop at the ISWC04, Hiroshima, Japan*, Nov. 2004.

[13] OpenID. *http://openid.net/*.

[14] Riya. *http://www.rija.com*.

[15] A. Uszok, J. M. Bradshaw, M. Johnson, and R. Jeffers. Kaos policy management for semantic web services. *IEEE INTELLIGENTSYSTEMS*, 284(5):32–41, July-August 2004.

[16] W3C. *http://www.w3.org/*.

[17] W3C. *http://www.w3.org/P3P*.

[18] W3C. *OWL Web Ontology Language 1.0 Reference*. http://www.w3.org/TR/2002/WD-owl-ref-20020729, July 2002.

Functional HDLs: A Historical Overview

Joseph Cordina
Dept. of Computer Science and A.I.
New Computing Building
University of Malta, Malta
joseph.cordina@um.edu.mt

Gordon J. Pace
Dept. of Computer Science and A.I.
New Computing Building
University of Malta, Malta
gordon.pace@um.edu.mt

ABSTRACT

When designing hardware systems, a variety of models and languages are available whose aim is to manage complexity by allowing specification of such systems at different abstraction levels. Languages such as Verilog and VHDL where designed with simulation in mind rather than synthesis and lack features such as parametrised complex circuit definitions, a must for the design of generic complex systems. A more modern approach is the use of functional languages for hardware description that take advantage of the inherent abstraction in this paradigm, resulting in a more concise and manageable description of the system. This paper gives an overview of different functional language implementations for hardware description, highlighting their historical significance in terms of their capabilities and design approach. We will compare and contrast different ways that certain features, such as circuit sharing, have been implemented in these.

Keywords

Hardware Description Languages, Language Embedding, Functional Languages

1. INTRODUCTION

When an electrical engineer needs to specify a complex system, he or she certainly does not specify the system by listing every single gate and connecting them. The usual techniques of modularity, re-use and abstraction are applied to improve both development time and also the quality of the final system. After specifying the system using some structural description, the engineer may want to perform actions on the circuit described:

- **Simulate the circuit.** At the very least the designer of the system would want to run a simulation of the circuit to observe its behaviour. This is also useful for testing the eventual circuit, where test cases are simulated on the circuit description and on the circuit proper.
- **Verify the circuit's behaviour.** Ideally one should be able to run some model checking techniques to verify that the described circuit obeys from specification. Having this facility increases the confidence of the circuit and also helps in discovering certain obscure bugs early on prior to the expensive hardware realization process.
- **Gather knowledge about the circuit.** These are metrics on the circuit such as number of components or expected signal propagation delay. More complex things which are desirable are behavioural description of the circuit, and an analysis of non-functional aspects.
- **Netlist generation.** Eventually the circuit will need to be placed in hardware, and thus it is crucial that the original circuit description is translated to the gate level together with an unambiguous description of how the gates connect to each other (also known as the *netlist*). The more abstract the specification language and the more automatic the low level description generation, the easier it is for the systems designer. It is also assumed that the original semantic behaviour of the system is maintained when the netlist is generated.
- **Circuit Transformation.** It is also commonly desirable for the designer to be able to make changes to the original circuit specification or to the netlist. This could include changes in functionality or tweaking the circuit to decrease the number of required components. Changes could be the result of bug-fixing or to tailor the system for re-use. Automated circuit transformation would be highly desirable.

The first step to allow the above is to generate some language that is able to describe the final target system. Very popular languages are VHDL and Verilog [1]. These languages offer a variety of features such as allowing the user to specify the circuit using a structural description of the circuit, alternatively using a behavioural description of a circuit, efficiently simulating the target system, synthesizing the high-level description to hardware and being general enough to describe any hardware system. One thing that is immediately obvious when looking at VHDL and Verilog compilers it that a lot of work has been done on optimising

them for simulation. In turn this led to the situation that synthesis becomes a very difficult process and it is not uncommon that one specifies a system that behaves differently after synthesis [4]. One can still use the behavioural description for testing, yet it obviously is not an ideal scenario. In addition, it soon became apparent that certain circuits are very difficult to specify in either VHDL or Verilog. Earlier versions of VHDL did not allow one to define complex circuits which vary according to the definition's parameter (for example a circuit which is made up of a number of components, the number of which is defined as a input parameter inside the circuit's definition). More modern versions allow a very limited subset of these[1]. In brief, while Verilog and VHDL are very successful tools, they do not lend themselves easily to higher-order descriptions when giving the structural description, thus limiting the amount of abstraction one can have. These higher-order descriptions are usually known as *connection patterns* and are functions that take other circuits as input parameters and whose outputs are generic patterns of the input circuits, for example rows, trees or arrays of circuits.

What we discuss in this document is an approach that has proved to be very successful in describing circuits at multiple levels of abstraction, while being able to maintain easy and automatic synthesis and simulation. We concentrate mainly on the structural description of circuits, even though we mention briefly some recent research work in terms of automatic synthesis of behavioural descriptions. Also we limit ourselves to describing synchronous systems, due to the fact that combinational circuits with feedback can only converge to a specific fixed output when taking into account low level propagation delay, something which one normally tries to abstract away from.

2. FUNCTIONAL HDLS

We will now discuss how the functional language paradigm has been been used to allow the specification of hardware systems. This has been done using one of two approaches: the development of new functional languages or the use of existing languages and embedding these with hardware description capabilities.

2.1 Early Work

Following Backus' Turing Award Paper in 1978, it was immediately obvious that descriptions of programs in functional languages tend to be concise and since abstraction is implicit in the model, it is an ideal platform for describing complex systems. In 1984, Sheeran [17] developed a variant of FP, μFP, and used it to describe the tally circuit. What is apparent is that the definition was much more understandable and concise than the same description in VHDL or Verilog (languages that had still to be created). One of the reasons is that the circuit was defined recursively, thus lending itself perfectly to the functional style of programming. In addition, such a concise description makes it easier to debug and to modify.

In μFP, circuits are defined in terms of built in connection patterns, i.e. functions that accept circuit descriptions as input parameters and connect these descriptions together depending on some other input parameters. The successor of μFP was Ruby [18]. In Ruby, while maintaining the concept of combinators, it looks at circuits as relations on streams. In other words, inputs and outputs are seen as a stream of data and circuits as the functions transforming them. This gives the advantage that components such as a delay can be easily specified.

[1]This is achieved through the `generic` keyword but is still not *generic* enough!

2.2 Functional Embedded Languages

Early attempts to create functional hardware description languages (HDLs) concentrated on the creation of new languages that make use of the functional paradigm. Yet this entails the construction of compilers and interpreters for each new language and also does present the problem of deciding upon the syntax and semantics of each new in-built feature.

An alternative approach, is to embed a hardware description language in a generic host language. This allows one to make use of all the features of the host language, thus taking advantage of all the packages available for the host language, including compilers and debuggers. Another large advantage is that the written circuit descriptions are themselves objects within the host language and can thus be manipulated, inspected and transformed. To embed the new language, one usually creates new libraries to allow the description of hardware in the host language.

One the earliest attempts to create an embedded HDL was HDRE [12]. This was implemented in Daisy, a non-strict functional language. In HDRE, wires are treated as a stream of values using lists. One can then easily simulate the circuit by defining circuits as transformation functions on these lists. One can also synthesize the circuit's definitions by having alternate values within the list and defining the circuits themselves as functions on lists of generic types. Depending on the type of values within the lists passed to the functions, one can evaluate (i.e. simulate) the circuit or one can generate a circuit description(i.e. netlist). This approach is called *shallow embedding*. A typical implementation within a Haskell-like language would be as follows:

```
type Signal = [Bit]

inv::Signal -> Signal
inv = map bit_invert

bit_invert::Bit -> Bit
bit_invert True = False
bit_invert False = True
```

Having a lazy programming language, one can also talk about infinite lists which allows any complex circuit to be specified, such as delay circuits whose delay is an input parameter. The biggest disadvantage with shallow embedding is that since circuits are programs within the language, they cannot be inspected and thus something simple like generating the number of gates in the target netlist is impossible.

2.3 Data Types and Deep Embedding

Instead of using lists of values, one can alternatively define circuit descriptions as values in a recursive data type. Then one can write functions that take these values and manipulate them. Thus the description given above would look as follows:

```
data Signal = Inv Signal | Low | High | ...

inv :: Signal -> Signal
inv = Inv
```

We can also create functions that are able to evaluate a given circuit description, thus effectively simulate it. Additionally one can write a function to generate the symbolic interpretation of the circuit, thus resulting in the netlist (symbolic evaluation).

This approach has been taken by the majority of functional HDLs today. One the early implementations to make use of this approach was Hydra [14], a functional HDL implemented in Haskell and now used as a teaching language. In Hydra one still has to annotate the circuit descriptions when needing to generate the netlists. A follow up to this language is Lava [4], a language with a large suite of features including automatic synthesis of circuits in VHDL, specification of generic connection patterns, automatic verification of properties through the use of observers and an adequate model checking tool. Another embedded HDL within Haskell is Hawk [7]. Hawk's main target is for modeling and simulating microprocessor architectures. Architectures can be described at the behavioural level. As Claessen notes in [4], it is very difficult to generate netlists from such a high level of abstraction. In Hawk, high-level components are used as elementary objects within the language and thus its very difficult to simplify these automatically to their gate-level counterparts. Another deeply-embedded HDL, this time within ACL2 is DE2 [9]. This language is mainly targeted towards rigorous hierarchical description and hierarchical verification of finite-state machines. Another variant is the modeling of streams within Daisy [10], a descendant of LISP, which can be used to model communication between self-timed communicating processes.

2.4 Haskell and Embedding

As one can see, Haskell is being used extensively for the use of embedding hardware description languages. One of the main reason for this is purely historical, in that the people working on embedded HDLs have been working closely with the Haskell development team. Yet arguably Haskell has got a very strong type system and is well renown for its elegance and clarity of syntax and semantics.

Other languages have also been used to embed within them HDLs, some with more success than others. There are several groups working on different languages yet one that seems very promising is the construction of an HDL within *reFLect*, a functional language based on ML [11]. Interestingly within such a language one can make use of shallow embedding and then use the reflection capabilities[2] to manipulate the circuit functions, thus arbitrarily changing from shallow to deep embedding as required.

Deciding to embed within a host language does have its disadvantages, primarily that one has to live with limitations of a generic language that was not designed primarily as an HDL. Within Haskell, one cannot define types that can also be given the allowed size of the type. These are known as *sized types*. Certain circuits makes certain assumptions on their input and output types and thus it would be desirable to be able to talk about type sizes within Haskell. Additionally as noted in [4], it would be helpful if Haskell was able to distinguish easily[3] between parameters to circuit descriptions, at the very least between the inputs and outputs signals.

3. OUTPUT SHARING AND FEEDBACK

It is obvious that one inherits a lot of advantages when making use of a functional language for hardware description. One major feature is *referential transparency* whereby any expression will always yield the same result for the same arguments, a property which is assumed in hardware and subsequently components are seen as functional elements. In functional languages, referential transparency is a result of lambda beta reduction, whereby evaluation becomes a simple exercise of argument replacement. Unfortunately this hides away context and does not allow us to refer to internal components form other parts of the program.

Consider Figure 1 that depicts two typical circuits made up of several components. When describing the first in a Haskell-like Language, one would use:

```
circuit_i::Signal -> Signal -> Signal
circuit_i in1 in2 =
  let interm = f in1 in2
  in g interm interm
```

While this definition might look strange, we know it will behave correctly precisely because of referential transparency whereby the evaluation of *g* is applied to two separate evaluations of *interm*. Yet this evaluation strategy leads us to the realization that circuit (ii) cannot possibly be described since f would have to be evaluated twice. Implementing the second circuit as follows, clearly highlights the resulting similarity between the first and second circuits.

```
circuit_ii::Signal->Signal->(Signal,Signal)
circuit_ii in1 in2 =
   let interm = f in1 in2
   in (g1 interm, g2 interm)
```

While this has no effect on its behavioural semantics (thus its simulation), it does have a huge effect when this circuit is realised in hardware. Since the above naive description of this circuit will result in two separate and identical implementations of *f*, this will result in a larger number of components than is strictly required.

[2] *reFLect* is able to reason and manipulate about the programs written within it, using quote and unquote operators

[3] Note that this is possible in Haskell as shown in Wired, yet the techniques used are not straightforward

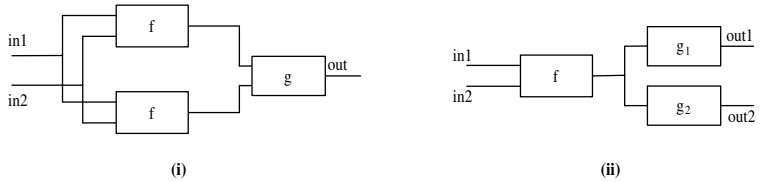

Figure 1: Typical circuits. (i) makes use of two identical circuits while (ii) shares the output of one single circuit.

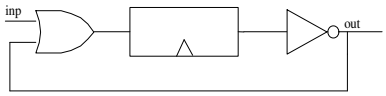

Figure 2: Circuit containing feedback. This circuit will alternate from true to false with every clock cycle, initiated by a true input.

The situation worsens when one has feedback loops, a common occurrence in real circuits. Consider Figure 2 where we have a typical circuit that inverts its output per cycle[4]. It is made up of an OR gate, a latch (a component that delays its input by one clock cycle) and an inverter. At first glance, one would implement such a circuit using the following code:

```
circuit::Signal
circuit inp =
   let out = inv (latch (or (inp,out) ))
   in out
```

Unfortunately when trying to generate the netlist of this circuit, a functional language compiler would try to expand the *out* argument in the second line, whose evaluation depends on the value of *out* again within the OR gate. This cyclic dependency has no terminating condition, and thus would be expanded until there is no more working stack space.

Such a terminating condition can only be found if the HDL in its execution trace can recognise components that it has already visited. Such a clause would also facilitate wire sharing as demonstrated in the beginning of this section. Note that through the use of shallow embedding and functional laziness, one can solve very nicely these problems, yet as noted before, shallow embedding does not allow us to analyse the circuit description within the program.

3.1 Wire Forking

One solution we can envisage is the use of a circuit that explicitly represents the forking of a particular output wire (see Figure 1(ii)). The semantics of this *fork* circuit is that an input wire is connected to this circuit that outputs two or more wires containing a copy of the input wire. This circuit can then be translated in the netlist generation to the diagram shown in the figure. This approach has several drawbacks, most importantly that the use will have to explicitly make use of this *fork* circuit to assemble the circuits. In addition, we cannot envisage how this can solve the problem of feedback loops.

[4]While such a circuit usually requires a synchronisation input, this has been omitted to simplify the circuit.

3.2 Explicit Naming

A better solution is to give a component an explicit name. This name will then be used when generating the description of the circuit. By storing the names of symbolically evaluated components and not evaluating already seen components, one can avoid having infinite recursion loops. A naive implementation would simply keep a list of names of evaluated components, and then traverse this list for every component that has to be evaluated. When implemented within a lazy functional language, making use of certain techniques that delay evaluation to the last possible moment can speed up evaluation considerably [8].

This approach was implemented in Hydra and proposed by O'Donnell in [13]. The code for the inverter circuit would now look as follows, where the inverter has been been given an explicit name.

```
circuit::Signal
circuit inp =
   let out = inv Name1 (latch (or (inp,out) ))
   in out
```

This code would not generate an infinite description since *Name1* will only be evaluated once. Another advantage of this approach is that the explicit name can be carried on all the way to the netlist generation, thus having a reference to the top-level design, aiding in debugging. Yet a major drawback is that it relies on the user to keep track of component names. While this might be a trivial task, it does tend to increase the possibility of error. One alternate approach is to make use of the *fork* approach mentioned above and one just explicitly names the fork components, thus reducing the management overhead required.

3.3 Monadic State

In traditional imperative languages, one makes use of variables to store data that will be needed by subsequent computations. In functional languages, one makes use of state monads to store some data through some computation [20]. In the first version of Lava [16], monads were used to hide away the details of components that have already been evaluated and need to be re-used. Using this method, a circuit identifier can be created automatically. While this approach does away with the user having to specify the names for components, it does require the user to use special operators and ways of programming that tend to quickly make the

code unreadable. In addition, to express feedback loops one needs to make use of some very nasty looking definitions.

3.4 Non-updateable references

Another approach proposed by Claessen and used in the latest version of Lava [6] is the use of non-updateable references resulting in *observable sharing*. Here the problem of sharing of wires is solved through the use of references, very similar to pointers in C. By allowing references to circuits and then evaluating reference equality, sharing can be easily implemented transparently from the user. In addition, both circuits shown in Figure 1 can be easily specified. Thus from the user's point of view, the only significant change is the types of the arguments to circuits.

The introduction of references in a functional language means that referential transparency is not upheld and one could also end up with side-effects. In Lava, the impact is limited by enforcing the references to be read-only. Underneath the hood, observable sharing is achieved without resorting to changes in the compiler by taking advantage of compilers that automatically evaluate an expression only once when this is repeated[5].

4. RECENT DEVELOPMENTS

One major drawback of functional embedded languages is that due to their abstraction mechanism, they are unable to describe *non-functional aspects*. In other words, while they are able to describe the components of the circuits (that contribute to the function of the circuits), an engineer might want to describe the components' eventual placement, how they are connected in terms of distance of wiring, etc. As importantly, before burning to hardware, the wires and the area configuration needs to be analysed since it has a direct impact on the cost of production. The reason why the description of non-functional aspects is difficult in functional HDLs is that it would require descriptions that cannot be evaluated, thus breaking the abstraction levels. Wired [2] is an extension to Lava where the placement of wires in the final circuit can be specified within the language. A series of operators are provided, with which one can connect different components together and specify their relative placement. One can also analyse the eventual space requirements and also power consumption. This approach has also been applied by Taha [19] where in their implementation, two specifications for the generation of the circuit are given. One specification talks about the circuit itself and the other specifies domain-specific optimizations targeted at the circuit generated. This approach avoids the transformation of circuits after they have been generated. Note that when using VHDL or Verilog, one specifies these aspects by using a different specification language than is used to describe the circuit. Yet this just adds another layer which the use has to manually correlate. As of yet, the problem of the complete specification of these aspects in a functional way has not been solved.

In our discussion, we have been mainly concerned with functional HDLs that can describe circuits by using a structural language, normally embedded within a functional language. In 2002, Claessen and Pace [5], showed a technique that allows the specification of a circuit using a behavioural description language. By embedding the behavioural language syntax using data types (very much like HDLs are embedded), one can specify the semantics of the language syntax in terms of other circuits (thus defining its semantics). Furthermore, by making use of the netlist generation mechanism within the HDL, one can easily also automatically synthesize the behavioural description. By making use of previously mentioned techniques, one can also verify properties about the behavioural program. In 2005, Claessen and Pace [15], also showed how one can verify properties about the compilation process itself. At time of writing, the authors of this paper are currently working on implementing an industry standard language, Esterel[3], into Lava allowing complex behavioural programs to be specified while also guaranteeing the semantics during compilation.

[5]This implies that the solution is not entirely portable. A naive compiler might not be able to implement observable sharing correctly.

5. CONCLUSION

Hardware systems are ever increasing in complexity, and are placing large demands on the hardware designer, both in terms of development time and complexity management. To solve these problems, the standard approach is to use different levels of abstraction and then map each layer to the one underneath. This was the approach taken with VHDL and Verilog, even though the mapping between some levels was not automatic and rarely dependable. Another approach is the use of hardware description languages that make use of functional programming paradigm, viewing circuits as maps between the inputs to the outputs. The main advantage of this paradigm is that abstraction is an implicit concept.

In this paper we have seen an overview of several functional HDLs in terms of their historical development. According to our opinion, the most advanced functional HDL to date is Lava with all its supporting libraries, automatic synthesis of circuits and automatic verification capabilities. We have also investigated a small selection of hardware characteristics that tend to be incompatible with the functional way of programming, namely circuits containing feedback and non-functional specification of circuits. We saw how these two characteristics have been recently tackled. We envisage several other approaches will arise in the near future since a lot of ongoing work is being done to solve these problems.

6. REFERENCES

[1] P. J. Ashenden. *The Designer's Guide to VHDL*. Morgan Kaufmann Publishers, 1996.

[2] C. Axelsson and Sheeran. Wired: Wire-aware circuit design. In *Charme 2005, LNCS 3725*. Springer, 2005.

[3] G. Berry. The Foundations of Esterel. In *Proof, Language and Interaction: Essays in Honour of Robin Milner*. MIT Press, 1998.

[4] K. Claessen. Embedded languages for describing and verifying hardware, April 2001. Dept. of Computer Science and Engineering, Chalmers University of Technology. Ph.D. thesis.

[5] K. Claessen and G. J. Pace. An embedded language framework for hardware compilation. In *Designing Correct Circuits '02, Grenoble, France*, 2002.

[6] K. Claessen and D. Sands. Observable sharing for functional circuit description. In *Proc. of Asian Computer Science Conference (ASIAN)*, Lecture Notes in Computer Science. Springer Verlag, 1999.

[7] B. Cook, J. Launchbury, and J. Matthews. Specifying superscalar microprocessors in HAWK. In *Formal Techniques for Hardware and Hardware-Like Systems*. Marstrand, Sweden, 1998.

[8] HaWiki. Typing the knot, cyclic data structures. Available at http://www.haskell.org/hawiki/TyingTheKnot.

[9] W. A. Hunt, Jr. and E. Reeber. Formalization of the DE2 Language. In *The Proceedings of the 13th Conference on Correct Hardware Design and Verification Methods (CHARME 2005)*, No. 3725, pages 20–34. Springer-Verlag, 2005.

[10] S. D. Johnson and E. Jeschke. Modeling with streams in daisy/the schemengine project. In M. Sheeran and T. Melham, editors, *Designing Correct Circuits (DCC'02)*. ETAPS 2002, 2002. Proceedings of the Workshop on Designing Correct Circuits, held on 6–7 April 2002 in Grenoble, France.

[11] T. Melham and J. O'Leary. A functional HDL in ReFLect. In *Designing Correct Circuits*, Mar. 2006.

[12] J. O'Donnell. Hardware description with recursive equations. In *IFIP 8th Internation Symposium on computer Hardware Description Languages and their Applications*, pages 363–382. North-Holland, 1987.

[13] J. O'Donnell. Generating netlists from executable circuit specifications in a pure functional language. In *Functional Programming Glasgow*, pages 178–194. Springer-Verlag Workshops in Computing, 1993.

[14] J. O'Donnell. From transistors to computer architecture: Teaching functional circuit specification in hydra. In *Functional Programming Languages in Education*, Volume 1125 of Lectures Notes in Computer Science. Springer Verlag, 1996.

[15] G. J. Pace and K. Claessen. Verifying hardware compilers. In *Computer Science Annual Workshop 2005 (CSAW'05)*. University of Malta, Sept. 2005.

[16] M. S. Per Bjesse, Koen Claessen and S. Singh. Lava - hardware design in haskell. In *International Conference on Functional Programming*. ACM SigPlan, September 1998.

[17] M. Sheeran. μfp, An Algebraic VLSI Design Language. In *LISP and Functional Programming*, pages 104–112. ACM, 1984.

[18] M. Sheeran. Describing Hardware Algorithms in Ruby. In *Functional Programming, Glasgow 1989*. Springer Workshops in Computing, 1990.

[19] W. Taha. Two-level languages and circuit design and synthesis. In *Designing Correct Circuits*, Mar. 2006.

[20] S. Thompson. *Haskell, The Craft of Functional Programming*. Pearson Assison-Wesley, 2nd edition, 1999.

System for Spatio-Temporal Analysis of Online News and Blogs

Angelo Dalli
University of Malta

angelo.dalli@um.edu.mt

ABSTRACT
Previous work on spatio-temporal analysis of news items and other documents has largely focused on broad categorization of small text collections by region or country. A system for large-scale spatio-temporal analysis of online news media and blogs is presented, together with an analysis of global news media coverage over a nine year period. We demonstrate the benefits of using a hierarchical geospatial database to disambiguate between geographical named entities, and provide results for an extremely fine-grained analysis of news items. Aggregate maps of media attention for particular places around the world are compared with geographical and socio-economic data. Our analysis suggests that GDP per capita is the best indicator for media attention.

Categories and Subject Descriptors
H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; J.4 [**Social and Behavioral Sciences**]: Economics and Sociology;

General Terms
Algorithms, Theory, Performance, Design, Economics

Keywords
Geolocation, disambiguation of geographical named entities, media attention, news, blogs, social behavior, spatio-temporal

1. INTRODUCTION
Online news and, more recently, blogs, are increasingly becoming one of the most popular destinations for Internet users, slowly increasing their influence to levels approaching those of traditional media [1]. Media attention and popular attention shifts continuously as new events happen in the world. Generally, media attention influences popular attention, although the reverse also occurs to a lesser degree. Attention in this context can be conveniently defined as the number of documents on a given subject, which is the same definition used by Zuckerman in his seminal paper on Media Attention profiles [5].

Existing news and blogs classification systems such as Google News and Blog Pulse usually focus on topic classification or keyword frequency tracking over time [6,7]. Previous work on georeferencing texts with a geospatial aware NER system has addressed the issues of spatial grounding of geographical entities and geographical name disambiguation [9,10,11,12,13] utilizing input from text analysis or Internet IP and DNS data [14]. However, almost none of these systems have comprehensively accounted for the temporal aspect associated with geographical named entities. This work (partly supported by EPSRC grant EP/C536762/1 and a small grant from Linguamine) presents a system, cpGeo, which analyses mentions of different geographical locations over time in news texts and blogs, creating real-time maps of shifting attention profiles that are convenient for highlighting current hot spots and determining what places capture the most attention in the world over time. Our interest is to find a set of indicators that are indicative of the level of attention enjoyed by different places in the world, and hence, the people living at those places.

2. ANALYSIS SYSTEM
The spatio-temporal analysis system, called cpGeo, is made up of five components (a high-performance web crawler, distributed storage, knowledge extraction, geospatial processor and data mining system) that allow download and analyze millions of items in a highly scaleable manner and generate summary reports.

cpGeo currently downloads between 18,000 to 21,000 news items every day from around 6,000 sources. Our level of coverage is rapidly approaching 100% of all online news published on the Internet everyday. In order to have adequate coverage of news from earlier years, we supplemented the news through the LDC English Gigaword corpus [14]. Blog entries are also being downloaded at a rate of around 156,000 blog items a day from around 90,000 authors. The cpGeo knowledge extraction subsystem performs various tasks related to basic text document processing and knowledge extraction. Documents are indexed and processed through a custom-built multiple document summarisation system that automatically detects document duplicates and merges highly similar documents together. Named entities and related events are identified and extracted to a temporal database. The geospatial processor identifies and disambiguates references to geographical locations around the world, and can produce graphical GIS-like presentations of its output results. The clustering and data mining system, utilizes a multivariate cluster analysis technique with an integrated rule learning model [18,19,20] and a small database of world knowledge that is used to interpret results correctly.

3. GEOSPATIAL PROCESSING
The cpGeo geospatial processor has three main components namely, the Multilingual Geospatial Database, a Multilingual Geographical Named Entity Recognizer (GNER) and a Disambiguation Module. The database has entries in 139 different languages and 3 main hierarchical levels covering 251 countries, 4,815 administrative regions and 7,574,966 individual place names and features. The database allows us to take into account the element of time and the fact that place names sometimes change over time.

Figure 1 shows the spatial database coverage, with shaded regions representing recognized geographical locations in the world. Some regions have lower coverage density (this is for

example apparent for India, which has high population density but lower coverage in the spatial database).

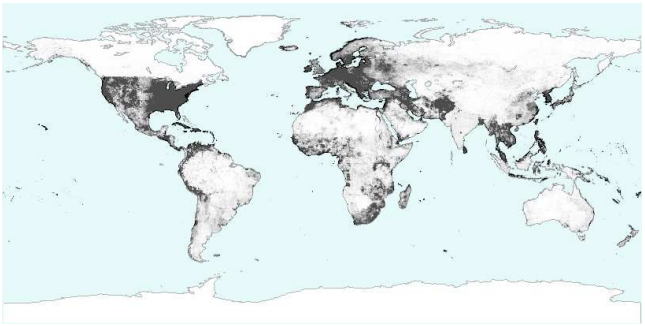

Figure 1. Spatial Database Coverage.

The geospatial database also contains additional information such as the latitude and longitude using the WGS84 geodetic reference system, feature type (populated place, street, etc.), relative importance, and aliases. The GNER uses the feature type information to determine the reliability of entries in the geospatial database, making it possible to identify, for example that "Lascaux", "Lascaux Cave", "Cave of Lascaux", and "La Grotte de Lascaux" refer to the same location. The GNER also has a geographic anaphora resolver, enabling it to know, for example, that "Bay Area" is referring to "San Francisco Bay Area".

Surprisingly, there are many duplicated place names in the world (around 10% to 25% of all place names have some duplicate elsewhere, depending upon the region or country). The GNER uses a mixture of heuristics and statistics to successfully disambiguate between duplicates. Geographical proximity and relative importance of other place names mentioned in the same context are considered in the disambiguation process.

The GNER also uses the knowledge extraction system to determine whether ambiguous names should be classified as person names or place names (e.g. to determine whether "Washington" is referring to the city, state or surname), enabling it to successfully resolve ambiguities in over 98% of cases. cpGeo achieved an F-measure of 0.9965 compared to 0.904 for the Perseus system [13].

4. EVALUATION AND RESULTS

We have evaluated the cpGeo system on our main news items database spanning from 1994 to 2005 (a total of 4,197 days). On average, every day had mentions of around 500 unique location names with 16,500 mentions of geographical named entities. Figure 2 shows the output of the system for 1 January 2000. Generally, when viewed on a global scale, the map changes slowly, although spikes and changes occur rapidly on local scales. The cpGeo system also keeps aggregate statistics of all place names mentioned together with their frequency, thus building up a map that indicates the regions in the world that are receiving the most media attention (as shown in Figure 3). In the United States it is apparent that North-East states receive more attention than other states, with the exception of California. In Europe, the UK and Belgium also receive more attention, while in Asia, Japan gets mentioned most frequently (with China catching up).

The aggregate maps can be useful in predicting the background level of attention that a particular region usually receives, providing better means of identifying spikes and anomalies instead of using simple threshold or rate increase methods. Aggregate maps represent a probability density function for the amount of news coverage likely to occur at any particular location in the world.

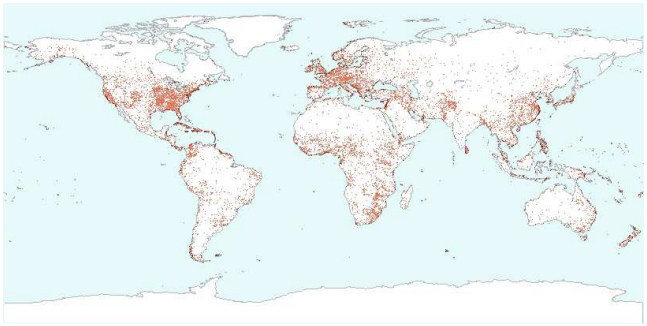

Figure 2. System Output for 1 January 2000.

Our results show that the top 80 mentioned place names consistently dominate the daily global news, generating more than 50% of all mentions on average. The top 3 daily place names generate around 11% of all mentions in the news.

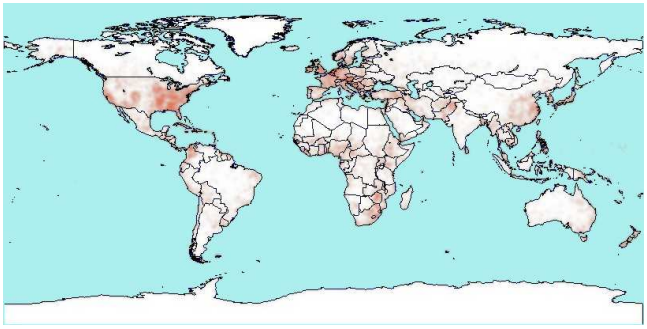

Figure 3. World Average News Media Attention.

We have also evaluated the cpGeo system on a small geographical scale using a four year collection of news about the smallest EU member state of Malta. Based on this evaluation we have determined that the cpGeo system can produce accurate results at a global resolution of around 3m x 3m.

Various statistical indicators were examined in an attempt to find correlations between statistical indicators and the cpGeo media attention ratings, with Number of Unresolved International Disputes, GDP Per Capita, and Number of Internet Users being the top three indicators that correlate with media coverage with GDP per capita being the most significant indicator of media attention.

The clustering system also produced 26 distinct clusters of countries based on these indicators, showing that for poorer countries, the secondary determining indicator for media attention is their number of disputes, while for richer countries the secondary determining factor is the number of Internet users. Thus, poorer countries are often in the news whenever they are involved in some armed conflict or dispute, and are most likely to be portrayed in a negative fashion.

The cpGeo system results also show that certain countries are abnormally represented in the media and blogs. The top 5 most over-represented countries in the world (with respect to their population levels) are the Holy See (Vatican), Monaco, Liechtenstein, Iceland and Luxembourg while the bottom 5 most under-represented countries are India, China, Brazil, Indonesia and Pakistan. There is a huge disparity between the top and bottom countries, for example, each Liechtenstein citizen gets the same average media coverage equivalent to 4,800 Pakistanis.

The Application of Support Vector Machine for Speech Classification

O. Gauci, C.J. Debono, E.Gatt, P. Micallef

Department of Communications and Computer Engineering

University of Malta

Msida

{olgauc, cjdebo, ejgatt, pjmica}@eng.um.edu.mt

ABSTRACT

For the classical statistical classification algorithms the probability distribution models are known. However, in many real life applications, such as speech recognition, there is not enough information about the probability distribution function. This is a very common scenario and poses a very serious restriction in classification. Support Vector Machines (SVMs) can help in such situations because they are distribution free algorithms that originated from statistical learning theory and Structural Risk Minimization (SRM). In the most basic approach SVMs use linearly separating Hyperplanes to create classification with maximal margins.

However in application, the classification problem requires a constrained nonlinear approach to be taken during the learning stages, and a quadratic problem has to be solved. For the case where the classes cannot be linearly separable due to overlap, the SVM algorithm will transform the original input space into a higher dimensional feature space, where the new features are potentially linearly separable. In this paper we present a study on the performance of these classifiers when applied to speech classification and provide computational results on phonemes from the TIMIT database.

Categories and Subject Descriptors

I.5.1 [**Pattern Recognition**]: Models – Statistical.

General Terms

Algorithms, Performance, Theory.

Keywords

Speech recognition, Statistical Learning Theory, Support Vector Machine (SVM).

1. INTRODUCTION

In many practical learning algorithms we find many difficulties which manifest themselves in misclassifications during the learning phases[1]. Some of these complications are:

(i) The inefficiency of the learning algorithm itself, for example, the convergence to a local minima in gradient-descent based algorithms.

(ii) The size of the hypothesis that can become very large, thus requiring a large computational time making the solution impractical.

(iii) The available training set can be small. In this case the hypothesis class will become too rich, which leads to overfitting and hence poor generalization performance.

(iv) For a multidimensional search space, the learning algorithm requires a large number of parameters to be tuned, making the system difficult to use.

Support Vector Machines are learning systems that utilize a hypothesis space of linear functions in the implicitly defined feature space, trained using an algorithm from optimization theory that calculates a learning bias resulting from the statistical learning theory. The use of a kernel function ensures that the high dimensional feature space is used efficiently. The overfitting problem in the high dimensional feature space requires a learning bias which can be derived from the statistical learning theory. Optimization theory provides a clear characterization of the properties of the solution which leads to the implementation of efficient learning algorithms and makes sure that the hypothesis is represented in compact form. The convex learning bias will also ensure that local minima are not present so a solution can always be found efficiently even for training sets with thousands of examples[1].

The structure of this paper is as follows: In section 2 we present the theory behind the linear Support Vector Machine. This is followed by the concepts of the nonlinear Support Vector Machine in section 3. Finally sections 4 and 5 present some experimental results and a conclusion respectively.

2. LINEAR SVM

The reason behind using Support Vector Machines for classification is to find an efficient way of learning by separating Hyperplanes in the high dimensional feature space. The Hyperplanes must optimize the generalization bounds and the learning algorithm must be capable of dealing with thousands of training examples. The generalization theory gives a clear set of instructions on how to prevent overfitting by controlling the Hyperplane margin measures. Optimization theory can then be applied to give the necessary mathematical analysis to find the Hyperplanes which optimize these measures.

The Maximal Margin Classifier is the simplest Support Vector Machine[1]. It is based on the linear separability of the data in the high dimensional feature space and thus cannot be used in many real life applications. The Maximal Margin Classifier forms the basic building block of Support Vector Machines, i.e. to find the most separating Hyperplane in a proper kernel-induced feature space. This method is implemented by using a convex

optimization problem, minimizing a quadratic problem under linear inequality constraints.

Suppose that we have N training data points given by $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ where the input $x_i \in R^d$ and the output $y_i \in \{\pm 1\}$. The input is assigned a positive class if f(x) ≥ 0, and a negative class otherwise. Considering the case where f(x) is a linear function of x, f(x) can be written as,

$$f(x) = \langle w \cdot x \rangle + b \tag{1}$$

where $(w, b) \in R^n \times R$ are the parameters that control the decision function, and the decision rule is given by sgn(f(x)). As shown in Figure 1, a geometric interpretation of this hypothesis is that the input space is split into two parts by the Hyperplane,

$$\langle w \cdot x \rangle - b = 0 \tag{2}$$

The vector w defines a direction perpendicular to the Hyperplane while changing the value of b moves the Hyperplane parallel to itself. The parameters w and b are referred to as the weight and the bias terms respectively.

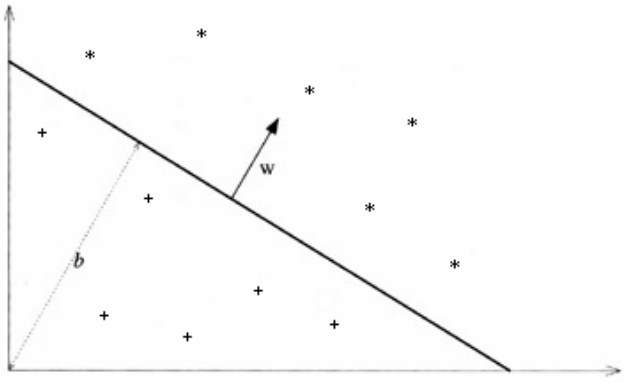

Figure 1. A Hyperplane (w, b) separating two classes

Further, we want this Hyperplane to have the maximum separating margin with respect to the two classes. Mathematically, we want to find the Hyperplane,

$$H : w \cdot x - b = 0 \tag{3}$$

and another two Hyperplanes,

$$H_1 : w \cdot x - b = +1 \tag{4}$$

$$H_{-1} : w \cdot x - b = -1 \tag{5}$$

parallel to it, with the restriction that there are no points between H_1 and H_{-1}, and that the distance between H_1 and H_{-1} is a maximum. Figure 2 shows an example for such a scenario where some positive examples are on Hyperplane H_1 while some negative examples are on Hyperplane H_{-1}. These examples are called Support Vectors because they define the separating Hyperplane.

The distance of a point on H_1 to H is given by:

$$\frac{\|w \cdot x - b\|}{\|w\|} = \frac{1}{\|w\|} \tag{6}$$

Therefore in order to maximize the distance separating the two

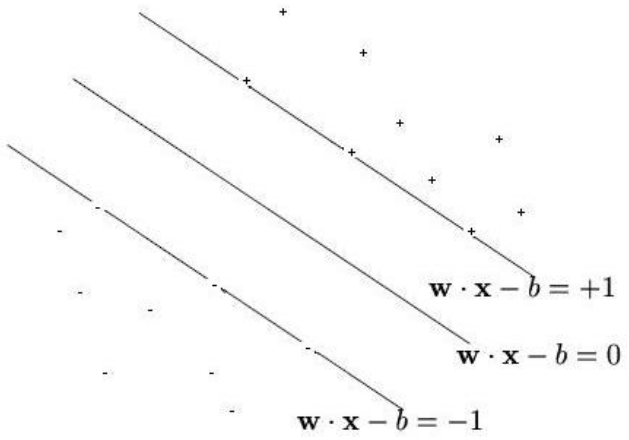

Figure 2. Maximal Margin Classifier

classes, we need to minimize $\|w\| = w^T w$ with the condition that no example is between the two Hyperplanes H_1 and H_{-1}.

Therefore,

$$w \cdot x - b \geq +1 \text{ for } y_i = +1, \tag{7}$$

$$w \cdot x - b \leq -1 \text{ for } y_i = -1. \tag{8}$$

Combining these two conditions we get,

$$y_i (w \cdot x_i - b) \geq 1 \tag{9}$$

Now our problem can be written as,

$$\min \frac{1}{2} w^T w \text{ subject to } y_i (w \cdot x_i - b) \geq 1 \tag{10}$$

However this is a convex, quadratic programming problem in a convex set. We can transform this optimization problem into its corresponding dual form by first considering the primal Lagrangian[1],

$$L(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{N} \alpha_i y_i (w \cdot x_i - b) + \sum_{i=1}^{N} \alpha_i \tag{11}$$

where $\alpha_i \geq 0$ are the Lagrange multipliers. Instead of solving this equation, we can solve the Wolfe dual form by maximizing the function $L(w, b, \alpha)$ subject to the constraint that the gradients of $L(w, b, \alpha)$ with respect to the primal variables w and b vanish, that is:

$$\frac{\partial L}{\partial w} = 0, \tag{12}$$

$$\frac{\partial L}{\partial b} = 0 \tag{13}$$

and the Lagrange multipliers $\alpha \geq 0$. Solving equations (12) and (13), we get,

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{14}$$

and

$$\sum_{i=1}^{N} \alpha_i y_i = 0. \tag{15}$$

Substituting equations (14) and (15) into the function $L(w,b,\alpha)$ we find:

$$L = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{16}$$

In this form the primal variables w and b have been eliminated and we end up with only one variable α. When the Lagrange multipliers are solved, we can go back to (14) to determine w, and we can classify an example x with:

$$f(x) = \text{sgn}(\sum_{i=1}^{N} \alpha_i y_i (x_i \cdot x) + b) \tag{17}$$

3. NONLINEAR SVM

In cases where the surface which separates the two classes is not linear, we have to implicitly transform the data examples into another high dimensional space such that the data points will be linearly separable in that space. Let the transformation into the high dimension feature space be $\Phi(\cdot)$. The dual problem now becomes[1]:

$$L = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(x_i) \cdot \Phi(x_j) \tag{18}$$

The dot product in the high dimensional space is equivalent to a kernel function of the input space,

$$k(x_i, x_j) = \Phi(x_i) \cdot \Phi(x_j) \tag{19}$$

Therefore, we do not need be explicit about the transformation $\Phi(\cdot)$. There are many kernel functions that can be used to solve this such as the radial basis function:

$$K(x_i, x_j) = e^{-\|x_i - x_j\|^2 / 2\sigma^2} \tag{20}$$

SVMs can also be extended to allow for noise or imperfect separation, hence the name soft margin Support Vector Machines. We do not strictly require the total absence of points between the Hyperplanes H_1 and H_{-1}, but we penalize the examples that cross the boundaries with the finite penalty factor C. We have also introduced a positive slack variable $\xi_i \geq 0$ in an attempt to include points which lie outside the Hyperplane separating their family. Figure 3 illustrates graphically the concept of slack variables introduced for an imperfect separation. Therefore, the separating Hyperplanes become:

$$w \cdot x_i - b \geq +1 - \xi_i \text{ for } y_i = +1 \tag{21}$$

$$w \cdot x_i - b \geq -1 + \xi_i \text{ for } y_i = -1, \tag{22}$$

$$\xi_i \geq 0, \forall i \tag{23}$$

We add the penalizing term to the objective function, so that it becomes:

$$\min \frac{1}{2} w^T w + C \sum_i \xi_i \text{ subject to } y_i (w^T x_i - b) + \xi_i - 1 \geq 0 \tag{24}$$

With this change the corresponding Lagrangian becomes[1]:

$$\begin{aligned} L(w,b,\xi,\alpha,r) = {} & \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \\ & - \sum_{i=1}^{N} \alpha_i [y_i (\langle x_i \cdot w \rangle + b) - 1 + \xi_i] - \sum_{i=1}^{N} r_i \xi_i \end{aligned} \tag{25}$$

The Wolfe dual problem can now be stated as:

$$\max L = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i \cdot x_j \tag{26}$$

subject to,

$$0 \leq \alpha_i \leq C, \tag{27}$$

$$\sum_i \alpha_i y_i x_i = 0 \tag{28}$$

The Lagrange multipliers are now bounded by C instead of infinity. The solution is again given by:

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{29}$$

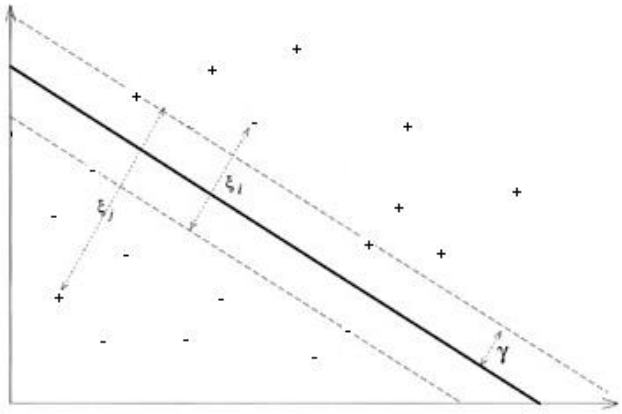

Figure 3. Slack Variable classification

4. EXPERIMENTAL RESULTS

Phonemes from the TIMIT database were segmented into frames of length 20 ms with a frame shift of 10 ms and filtered using a Hamming window. A Daubechies 10 filter was used to create a four level wavelet packet. The energies of the wavelet packets were calculated, thus obtaining a 16 dimensional feature vector.

The complete vowel dataset consisting of 20 phonemes from the TIMIT database were used for our experiments. The 20 class problem thus consisted of: /iy/, /ih/, /eh/, /ey/, /ae/, /aa/, /aw/, /ay/, /ah/, /ao/, /oy/, /ow/, /uh/, /uw/, /ux/, /er/, /ax/, /ix/, /axr/, and /ax-h/. 10, 000 samples were used to train the SVMs while 2,000 samples were used for testing.

During our experimentation the penalization factor C of the SVM was set to 60 while the value of σ for the radial basis kernel was set to 4. Table 1 presents some of the results showing the performance of the Support Vector Machine when vowels from the TIMIT Database were combined as many binary problems.

These results show that SVMs provide a good solution towards classification of binary phonemes. SVMs were also tested for multiclass applications, where the whole 20 phoneme set was used, but the results obtained were far from ideal. Further investigation in optimizing these tools is still necessary to apply satisfactorily these algorithms for speech recognition.

Table 1. Phonemes from TIMIT Database trained as Binary problems

Binary Problem	Precision(%)
'iy' – 'ih'	68.7 – 70.65
'ey' – 'aw'	86.1 – 93.5
'ih' – 'aa'	97.1 – 95.2
'iy' – 'ow'	96.5 – 95.2
'uw' – 'ax'	76.1 – 75.1
'ae' – 'aa'	94.0 – 84.1
'ao' – 'aa'	78.1 – 78.6
'ax' – 'ix'	89.1 – 87.5

5. CONCUSION

In this paper we evaluate the performance of a Support Vector Machine for phoneme recognition. The results obtained clearly show the classification power of Support Vector Machines in this application. Although good results have been achieved in the case of binary problems, future research work is required to extend these Learning systems for multiclass classification.

6. ACKNOWLEDGMENTS

This project is funded by a University of Malta's Research Grant.

7. REFERENCES

[1] N. Cristianini and J. S. Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning methods*, Cambridge University press 2000.

[2] A. Ganapathiraju, J.E. Hamaker, and J. Picone, "Applications of Support Vector Machines to Speech Recognition", *IEEE Trans. Signal Processing*, Vol. 52, No. 8, August 2004.

[3] C. Ma, M. A. Randolph and J. Dirsh, "A Support Vector Machine- Based Rejection Technique for Speech Recognition", *Proc. IEEE Int. Conf on Acoustics, Speech, Signal Processing 2001.*

[4] A. E. Cherif, M. Kohili, A.Benyetteou and M. Benyettou, "Lagrangian Support Vector Machines for Phoneme Classification", *Proc 9th International Conf. on Neural Information Processing,* ICONIP 02', Vol. 5.

[5] P. Clarkson, and P. J. Moreno, "On the use of support vector machines for phonetic classification", *Proc ICASSP March 99',* Vol. 5 pp 585-588, 1999.

[6] J. C. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines", Microsoft Research, Tech. report MSR-TR-98-14. April 1998.

[7] V. N. Vapnik, *The Nature of Statistical Learning Theory*, Springer – Verlang, New York 2000.

[8] R. Herbrich, *Learning Kernal Classifiers, Theory and Algorithms*, MIT Press 2002.

[9] B. Scholkopf and A. J. Smola, *Learning with Kernels, Support Vector Machines, Regularization, Optimization and Beyond*, MIT Press 2002.

Evolving Viable Pitch Contours

Kristian Guillaumier
kguil@cs.um.edu.mt
Dept. of Computer Science and AI
University of Malta

ABSTRACT

At a very basic level, a piece of music can be defined as an organised arrangement of sounds[1] occurring both sequentially (as in melody) and concurrently (as in harmony). As music evolved into a science and an established form of art, people started studying the characteristics of these sounds and drew sets of guidelines and rules, that if followed would produce pieces of music that are aesthetically more pleasing than others. Early examples can be seen in Pythagoras' observations and experiments with blacksmiths' hammers [1]. Allegedly some 2500 years ago, he was walking by a blacksmith's shop when he heard the ringing tones of hammers hitting an anvil. Upon further observation, he realised that a hammer weighing half as much as a previous one sounded twice as high in pitch (an octave – ratio 2:1). A pair of hammers whose weights had a ratio of 3:2 sounded a fifth apart. Eventually he came to the conclusion that simple ratios sounded good.

In this paper, we are concerned with the generation of musical phrases constrained by the rules that governed music developed during the so called Common Practice Period (CPP). This period refers to an era in musical history spanning from the 17^{th} to the early 20^{th} centuries [2] and included the Baroque and Romantic styles amongst others. Colloquially, music in the style of the CPP is sometimes better (but incorrectly) known as 'Classical' music.

General Terms

Aleatoric composition, music theory, common practice period, genetic algorithms.

1. INTRODUCTION

Computers have been used as an aid in composing music since the mid-1950s and the techniques generally employed fall into the categories of aleatoric composition and processes that return permutations of predefined musical elements such as pre-composed measures of music [6]. In the former technique, stochastic methods are used to generate sounds – possibly utilising some musical observations to guide random processes. In the latter, a number of predefined measures of music are selected and attached to each other to yield a piece of music. This technique has been practiced since the 18^{th} century using an algorithm known as the *Musikalisches Würfelspiel* (musical dice game) [7]. Mozart has been known to compose a number of Minuets based on this algorithm. An interactive example of this method can be found at [8].

In this paper, we present a Genetic Algorithm (GA) designed to generate pitch contours[2] that conform to the rules used in the CPP. Clearly, the rules we will be considering form the basis of the fitness function of the GA. Here we assume that the reader is familiar with basic–to–intermediate theory of music. We refer beginners to any introductory textbook on the matter such as [3], [4] and [5].

1.1 Rules for Developing Melodies/Pitch Contours

1. Notes in the melody should be diatonic.
2. Most of the melody must progress in stepwise motion.
3. The melody should contain a number of leaps. The number of leaps depends on the length of the melody and a leap must always occur in the context of contrary motion. Also, leaps must be a consonant interval.
4. Melodies should cover the whole range of notes assigned to it.
5. The highest note (climax) in a melody should occur only once, usually in the latter half. This climax note should be melodically consonant with the tonic.
6. Certain note intervals such as augmented intervals are unacceptable.
   (a) Augmented intervals are not allowed.

[1] In this context we refer to sounds of a musical nature – notes.

[2] A pitch contour is a sequence of notes that sound melodical but without possessing any rhythm (essentially, a melody composed with notes of a single duration only such as crotchets). Melodies are rhythmically more complex than pitch contours, but in this text the terms pitch contour and melody are synonymous.

(b) A diminished interval is not allowed unless the note following the interval falls between that interval by a perfect or imperfect interval.

7. The note on the seventh degree of the scale must rise stepwise to the tonic.
8. Melodies in a major scale should start and end with the tonic or dominant. Melodies in a minor scale should start and end with the tonic or mediant.
9. Only notes in a single soprano, alto, tenor or bass voice should be used. Arbitrarily, the soprano range is considered here (from middle C to G two octaves up).
10. Notes should be repeated rarely.

2. THE GENETIC ALGORITHM

Creating a melody conforming to a set of rules can be construed as a constraint satisfaction problem. Consider an 8-note melody to be composed using any of 7 notes (an octave-worth of diatonic notes). These parameters would yield 5,764,801 (7^8) different melodies – the search space. Longer, more varied melodies would clearly increase the search space immensely making the problem non-trivial. In this section we describe a GA used to search the space for melodies that conform to the rules outlined earlier on. It is assumed that the reader is familiar with the mechanics of a GA and its theory. Readers are referred to [9] for a thorough exposition.

2.1 Chromosome Structure

The scheme used to represent a candidate pitch contour as a chromosome in our algorithm is straightforward. The chromosome is an array of integers, where each integer represents a note value in the contour. The integer values representing each note are borrowed from the Musical Instrument Digital Interface (MIDI) standard, where, for example, middle-C is represented by the value 60, C# by the value 61, D by the value 62, etc... This idea is illustrated in Figure 1.

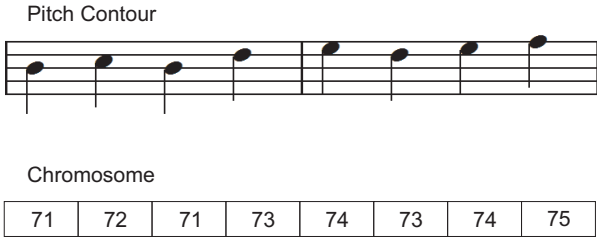

Figure 1: Chromosome representation of a pitch contour.

The simplicity of this representation scheme allows for uncomplicated designs for crossover and mutation operators. Additionally, the fact that the note values map to the MIDI standard allows us to export the contour as a MIDI file for quick auditioning or editing purposes.

2.1.1 Pitch Sets

The possible note values for each locus in the chromosome (i.e. each possible note in the contour) are selected from a set of permissable notes called the *pitch set*. Essentially the pitch set defines which notes the melody can be composed of. Restricting the melody to use only notes from a pitch set has a number of advantages:

1. Ensuring that the notes in the pitch set are diatonic, implies that any candidate pitch contour will be diatonic as well. This implicitly enforces the first rule mentioned previously.
2. Similarly, by ensuring that the notes in the pitch set are within the range of a single voice (e.g. the soprano voice), rule 9 above is implicitly observed.
3. The search space is reduced to the various permutations of the notes in the pitch set rather than all the notes in the range of a particular instrument.

2.1.2 Fixed Notes

During setup of the algorithm, certain loci in the chromosome can be fixed to certain notes. For example, the algorithm can be instructed that the first note in the melody should always be middle-C and no operator, such as a mutation, would be allowed to change it. The allows us to ensure that the pitch contour would, for example, always start with the tonic and end with the dominant. By specifying fixed notes in 'the middle' of the melody we can give it a particular texture or shape. Additionaly, since most notes are required to progress in stepwise motion, the fixed note effectively becomes an attractor for other notes thereby serving as a climax note.

2.1.3 Initialisation

In the initial population, chromosomes are initialised to pitch contours with notes randomly selected from the pitch set. Fixed notes in the pitch contour are observed – the fixed notes in any pitch contour are immutable.

2.2 The Fitness Function

The fitness function developed is penalty-based. A faultless melody has a fitness value of zero whilst the fitness of a flawed melody is negative. Essentially, for each rule that is violated, a penalty value is deducted from the fitness. The penalties for each rule are interpreted as follows:

1. RULE: Notes in the melody should be diatonic.
   INTERPRETATION: This rule can never be violated because a chromosome can only be composed of notes selected from a pitch set whose notes are already guaranteed to be diatonic. This rule is implicitly observed.
2. RULE: Approximately n% of the melody must progress in stepwise motion.
   INTERPRETATION: Determine the number of expected stepwise intervals in the melody from n. By observing the actual intervals in the candidate melody, determine the number of actual stepwise intervals. If the actual number of stepwise intervals is less than expected, apply a penalty (e.g. -5) to the fitness for each expected interval that is not present.
3. RULE: The melody should contain a number of leaps. The number of leaps depends on the length of the melody and a leap must always occur in the context of contrary motion. Also, leaps must be a consonant interval.
   INTERPRETATION: Determine the number of leaps in

the melody. If the actual number of leaps differs from some required amount, penalise in proportion to this difference. If the leap is not consonant, apply a penalty. For contrary motion, if the leap is preceded by a note that is not within its interval, apply a penalty. Finally, if the leap is followed by a note that is not within its interval, apply a penalty too.

4. RULE: Melodies should cover the whole range of notes assigned to it.
   INTERPRETATION: For each note in the pitch set to be used that does not occur in the melody, apply a penalty.

5. RULE: The highest note (climax) in a melody should occur only once, usually in the latter half. This climax note should be melodically consonant with the tonic.
   INTERPRETATION: Let c be the the number of times the highest note in the melody occurs. Apply a penalty to the fitness (c-1) times. The requirement of the climax note occurring in the latter half of the melody and being consonant to the tonic can be implicity observed by setting the climax note as a fixed note in the chromosome.

6. (a) RULE: Augmented intervals are not allowed.
   INTERPRETATION: Apply a penalty for each augmented interval in the melody.

   (b) RULE: A diminished interval is not allowed unless the note following the interval falls between that interval by a perfect or imperfect interval.
   INTERPRETATION: For each diminished interval in the melody, if the next note does not lie between that interval and is a perfect or imperfect interval, apply a penalty. Also, a penalty is applied if the melody ends in a diminished interval (there would not be a note following the interval).

7. RULE: The note on the seventh degree of the scale must rise stepwise to the tonic.
   INTERPRETATION: For each note on the seventh degree of the scale that does not rise to the tonic, apply a penalty.

8. RULE: Melodies in a major scale should start and end with the tonic or dominant. Melodies in a minor scale should start and end with the tonic or mediant.
   INTERPRETATION: This rule is implicitly observed by setting the starting and end notes as fixed notes in the chromosome.

9. RULE: Only notes in a single soprano, alto, tenor or bass voice should be used. The soprano range is considered here (from middle C to G two octaves up).
   INTERPRETATION: This rule is implicitly observed by setting the notes in the allowed pitch set to those in a desired voice range.

10. RULE: Notes should be repeated rarely.
    INTERPRETATION: Count the number of repeated notes in the melody. If this amount varies from some expected value, apply a penalty proportional to this difference.

2.3 Genetic Operators and Other Parameters

2.3.1 Crossover

In our algorithm, the crossover operator is a basic implementation of single-point crossover. A illustrated in Figure 2, a common locus in two parent melodies is randomly selected, and notes between the two parents are swapped along that interval. Parent melodies are selected to participate in crossover using a roulette-wheel selection scheme.

A *crossover rate* value governs how many parents are selected for crossover purposes. A crossover rate of 80% means that after selection, 80% of the new population will be made up of offspring that were generated by mating the parents. The remaining 20% of the new population is populated with new, randomly-created chromosomes.

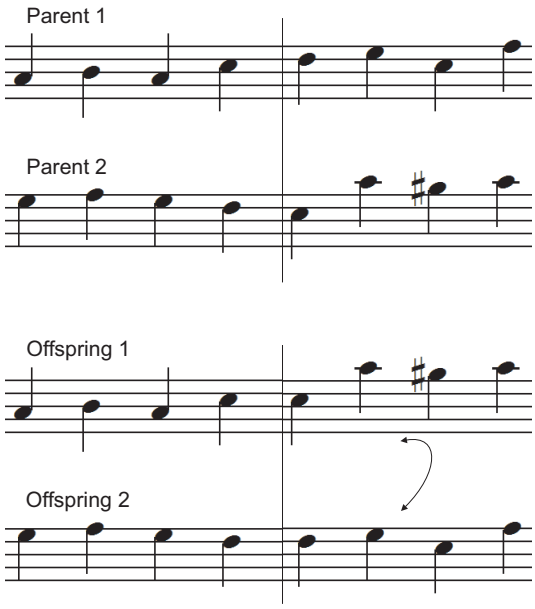

Figure 2: Single-Point Crossover between two pitch contours.

2.3.2 Random Mutations

In the algorithm a random mutation is implemented as:

1. The replacement of a random note in the melody with any random one in the pitch set of allowed notes.

2. The swapping of two, random notes in the melody.

A *random mutation rate* is used to determine the probability that a child chromosome created after crossover will be randomly mutated. The mutation operator is partial to the fixed note configuration of the chromosomes. A mutation is aborted if it would effect a fixed note.

2.3.3 Guided Mutations

This operator, works by altering a note in an attempt to rectify a deficiency in the melody. For example, if it is determined that a melody contains too little stepwise motion, a note in a non-stepwise interval is changed to make that interval stepwise. Similarly, if an augmented interval is found in a melody, a note in that interval is replaced to change the improper interval.

A *guided mutation rate* is used to determine the probability that a random child chromosome created after crossover will be mutated 'intelligently' as described above.

2.3.4 Elitism

The roulette–wheel selection mechanism used ensures that the fittest parents are paired to yield the new offspring pitch contours. Nonetheless, there is always the risk that the pitch contours generated after crossover and possible mutations have a fitness less then that of the original parent contours that spawned them. This implies that there is a possibility that, over time, the overall fitness of the population could degrade. To avoid this risk, an *elitism rate* parameter is used. This parameter represents a percentage of the best contours in the current population that will replace random contours in the next one.

3. EXPERIMENTAL RESULTS

In this section, we present a number of experimental runs of the algorithm and their respective results.

3.1 General Notes

1. Pitch contours have been generated in C Major, G Major, A Minor (harmonic[3]) and E Minor (harmonic). In this section, we only present the results of the algorithm when instructed to compose contours in A minor. This choice is arbitrary since setting a different scale for the algorithm to work in will not effect its performance in any way.

2. Various settings for population size, crossover and mutation rates have been used to determine how the GA converges under these conditions.

3. All pitch contours generated have been set to be 16 and 32 notes long.

4. The algorithm converges to multiple optimal solutions in all cases. The only cases observed when an optimal solution could not found were when they either did not exist or the parameters of the GA conflicted. For example when the pitch set is so limited that it would be impossible to satisfy the rate of stepwise motion desired – if the only notes in the pitch set are C5 and G5, it is clearly impossible to move in stepwise motion.

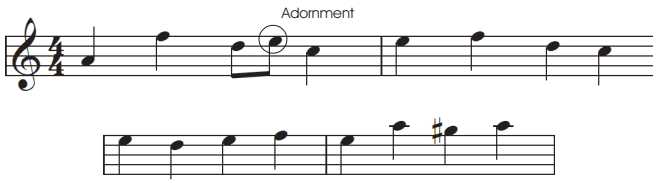

Figure 3: An optimal pitch contour (with an added adornment) generated by the algorithm.

[3]Natural minor with a raised seventh.

3.2 Experiment 1

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 15=G#5, Pos 16=A5}.
- Population Size: 100.
- Maximum Number of Generations: 50.
- Crossover Rate: 90%.
- Random Mutation Rate: 5%.
- Guided Mutation Rate: 5%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: 25.
- Average Number of Optimal Contours Yielded in Final Generation: 3.
- Example result: A4, B4, A4, F5, E5, C5, B4, C5, D5, C5, D5, E5, D5, A5, G#5, A5.

3.3 Experiment 2

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 15=G#5, Pos 16=A5}.
- Population Size: 100.
- Maximum Number of Generations: 50.
- Crossover Rate: 50%.
- Random Mutation Rate: 20%.
- Guided Mutation Rate: 20%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: ¿ 50.
- Average Number of Optimal Contours Yielded in Final Generation: 0.2.
- Example result: A4, E5, D5, C5, D5, E5, D5, E5, F5, C5, D5, E5, F5, E5, G#5, A5.

3.4 Experiment 3

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 15=G#5, Pos 16=A5}.
- Population Size: 100.
- Maximum Number of Generations: 50.
- Crossover Rate: 100%.
- Random Mutation Rate: 5%.
- Guided Mutation Rate: 5%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: 20.
- Average Number of Optimal Contours Yielded in Final Generation: 3.
- Example result: A4, E5, D5, C5, D5, C5, E5, F5, E5, F5, E5, F5, E5, A5, G#5, A5.

3.5 Experiment 4

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 25=E5, Pos 32=A5}.
- Population Size: 200.
- Maximum Number of Generations: 200.
- Crossover Rate: 90%.
- Random Mutation Rate: 5%.
- Guided Mutation Rate: 5%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: 70.
- Average Number of Optimal Contours Yielded in Final Generation: 15.
- Example result: A4, B4, C5, D5, E5, A4, B4, A4, E5, D5, A5, G#5, A5, C5, D5, C5, D5, C5, B4, C5, E5, D5, E5, D5, E5, F5, E5, D5, E5, F5, E5, A5.

3.6 Experiment 5

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 25=E5, Pos 32=A5}.
- Population Size: 200.
- Maximum Number of Generations: 200.
- Crossover Rate: 50%.
- Random Mutation Rate: 20%.
- Guided Mutation Rate: 20%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: ¿ 200.
- Average Number of Optimal Contours Yielded in Final Generation: 0.1.
- Example result: A4, B4, C5, D5, E5, A4, B4, A4, E5, D5, A5, G#5, A5, C5, D5, C5, D5, C5, B4, C5, E5, D5, E5, D5, E5, F5, E5, D5, E5, F5, E5, A5.

3.7 Experiment 6

- Key: A Minor (harmonic).
- Contour Length: 16 notes.
- Pitch Set: {A4, B4, C5, D5, E5, F5, G#5, A5}.
- Note Usage: 80%.
- Fixed Notes: {Pos 1=A4, Pos 15=G#5, Pos 16=A5}.
- Population Size: 200.
- Maximum Number of Generations: 200.
- Crossover Rate: 100%.
- Random Mutation Rate: 5%.
- Guided Mutation Rate: 5%.
- Stepwise motion rate: 80%.
- Leap Rate: 15%.
- Average Generations to Yield Optimal Contours: 50.
- Average Number of Optimal Contours Yielded in Final Generation: 25.
- Example result: A4, B4, C5, D5, F5, E5, D5, C5, B4, A4, C5, B4, D5, A4, C5, B4, E5, D5, E5, F5, C5, D5, C5, F5, E5, F5, E5, D5, E5, D5, C5, A5.

3.8 Some Observations

1. As expected, searches for longer contours require a larger population and in some cases more generations to produce a result. This can be easily correlated to the immensely larger search space.

2. Low crossover rate values (albeit higher mutation rates) hinder a successful search for optimal solutions. As the crossover rate decreases, the GA effectively degenerates into a random search with is inadequate for such large search spaces.

3. Very long optimal pitch contours can be discovered. Optimal 64, 96 and 128–note pitch contours could be found using the exact same parameters used in experiment 4 above with the exception that for 128–note melodies, more generations and a slightly larger population size was required for the algorithm to converge to one or more optimal solutions.

4. CONCLUSION

In the title of this paper we labeled the pitch contours we sought to generate as *viable*. We associate the term viable with whether a contour observes the rules imposed or not. By observing these rules we could safely say that all viable contours do sound melodic and flow smoothly. This observation can be intuitively demonstrated by listening to the results of the algorithm. The issue of whether the contours actually sound beautiful, or whether they express some kind of emotion is another issue altogether. Whilst it is true that certain compositional techniques can give melodies some emotional character[4], in this paper we have not considered them and left the issue as a potential next topic of research. We conclude this work by suggesting some techniques and future projects than can augment the simple pitch contours we generated here and use them as the basis of fully fledged musical compositions:

- Apply note grouping techniques and rhythmic unit presets to the pitch contour for it to become a complete rhythmic melody line.

- Using species counterpoint techniques to enrich the melody harmonically. Chord progression rules may be derived by studying common progressions used in the CPP.

- Observing cadences when harmonising the melody.

- Observing orchestration guidelines when determining the relationships between voices.

- Using instrumentation principles when choosing instruments to play a given voice. For example certain instruments possess timbres that lend themselves to a more dramatic score.

[4]For example, it is a known 'fact' that composing in a major key usually results in 'happy sounding' scores. Composing in minor keys is usually associated with music of the more 'sad' kind.

5. REFERENCES

[1] Anthony Ashton. Harmonograph: A visual guide to the mathematics of music. Wooden Books.

[2] Benjamin Piekut. From No Common Practice: The New Common Practice and its Historical Antecedents. American Music Center. http://www.newmusicbox.org/page.nmbx?id=58tp01

[3] The Associated Boards of the Royal Schools of Music. Rudiments and Theory of Music.

[4] Michael Miller. The Complete Idiot's Guide to Music Theory. Alpha Publishing.

[5] William Lovelock. First Year Harmony. Hammond Textbooks.

[6] Charles Dodge, Thomas A. Jerse. Computer Music: Synthesis, Composition and Performance. Second Edition. Schrimer.

[7] David Cope. Virtual Music: Computer Synthesis of Musical Style. The MIT Press.

[8] John Chuang. Mozart's Musikalisches Würfelspiel. http://sunsite.univie.ac.at/Mozart/dice/.

[9] David E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional.

Testing PRNG's for use in a GA

Clyde Meli
Assistant Lecturer
CIS Department
University of Malta
Tel: 2340-2509

cmeli @ cis.um.edu.mt

ABSTRACT

In this paper, the results of some tests including the Chi-Square and the Diehard Battery tests of randomness on some pseudo random number generators are given. The generators were chosen for use in a Genetic Algorithm (GA) package called GAGENES, written in object-oriented C++. The influence of PRNG's on a GA is discussed briefly.

Categories and Subject Descriptors

I.2.m [**Artificial Intelligence**]: Miscellaneous – *PRNG, testing, genetic algorithms*..

General Terms

Algorithms, Measurement.

Keywords

ga,gagenes,prng,testing,c++,object-oriented

1. INTRODUCTION

Four pseudo random generators were tested using a Chi-Square test [2] and the Diehard tests [3]. Meysenburg [5] by empirical testing showed that a good PRNG may not necessarily give significantly better GA performance. Meysenburg et al [6] determined that there is statistical evidence that certain PRNGs can provide improved GA perfomance. Cantu-Paz [1] showed that in GAs as in other fields, it is preferable to use the best PRNG available especially when initialising the GA population. With this in mind, four PRNG's were tested as follows: 1) a Chi-Square test was implemented ; 2) a 12M data file was generated for every PRNG for testing by Marsaglia's Diehard program. All PRNG's were seeded by the system time. The results are presented in this paper. The generators tested were: the PRNG supplied with GNU GCC C++ 3.4.5 / libstdc++'s linear congruential generator (LCG), Strousoup's PRNG implementation (an LCG) in [7], Knuth's Lehman LCG [2] (this was implemented in C++) and Jasper Bedaux's C++ port of the Mersenne Twister [4]. The latter was expected to be the best PRNG from those tested.

George Marsaglia's **DIEHARD** suite of statistical tests consist of fifteen tests: Birthday Spacings, Overlapping Permutations, Ranks of 31x31 and 32x32 matrices, Ranks of 6x8 matrices, Monkey Tests on 20-bit Words, Monkey Tests OPSO, OQSO and DNA, Count the 1's in a stream of bytes, Count the 1's in specific bytes, Parking Lot, Minimum Distance, 3D Spheres, Squeeze, Overlapping Sums, Runs and Craps. More information about the suite may be found at http://stat.fsu.edu/~geo/diehard.html

To run the test, Marsaglia's Windows binary was used and it was given a twelve megabyte file of random numbers generated using GAGENE's random number test suite (which called the four PRNG's under test). All tests were run on an Athon FX-55 processor computer running XP. The test suite also compiles and runs under Linux and other Posix style OS's.

2. CHI-SQUARE RESULTS

Each simple frequency Chi-Square test was repeated 100000 times and the mean chi value was calculated. The range of output values was subdivided into a fixed number of ranges (101) of equal size. A large number of random numbers were generated using the PRNG being tested, each number ranging between 0 and 1 (inclusive). 1000000 numbers were generated and the number of values falling in each interval were counted. As the intervals are of equal size, it is expected that approximately the same number of values should fall in each interval (around 9900). The chi-square statistic is evaluated based on the expected and actual counts for each interval, an average chi-square value based on 100000 repetitions is evaluated and compared with the critical value for a chi-square random variable with 100 degrees of freedom and a probability of 0.95.

Chi-Square Test 1 (GNU C++ generator)
Total chi value (Averaged over 100000 trials)=90.1802
succeeded
Chi-Square Test 2 (Mersenne generator)
Total chi value (Averaged over 100000 trials)=100.013
succeeded
Chi-Square Test 3 (Stroustroup generator)
Total chi value (Averaged over 100000 trials)=-1.59487e+06
failed
random Chi-Square Test 4 (Knuth generator)
Total chi value (Averaged over 100000 trials)=13768.5
failed

When run on a Linux machine (Pentium 4), the results were:

random Chi-Square Test 1 (GNU C++ generator)
Total chi value (Averaged over 100000 trials)=99.2055
succeeded
random Chi-Square Test 2 (Mersenne generator)
Total chi value (Averaged over 100000 trials)=100.263
succeeded
Chi-Square Test 3 (Stroustroup generator)
Total chi value (Averaged over 100000 trials)=-1.59487e+06
failed
random Chi-Square Test 4 (Knuth generator)
Total chi value (Averaged over 100000 trials)=13941.9
failed

3. DIEHARD RESULTS

The following table shows the p-values produced by the Diehard suite of tests on the GCC C++ generator (resulting log file is rcpp.log).

Table 1. GCC C++ Diehard Test Results Part 1

Birthday Spacing	**Bin. Rank 31x31**		
.384929	.660257		
.718312	**Bin. Rank 32x32**		
.319485	.602642		
.945639	**Bin. Rank 6x8**		
.579601	.927291	.739037	.631103
.489100	.870957	.222028	.475929
.843946	.193213	.093370	.947564
.498052	.073664	.162025	**.220698 KSTEST**
.360428	.967210	.338048	
.510440 KSTEST	.514890	.496957	
OPERM5	.682539	.192736	
.171280	.456832	.455710	
.323761	.046014	.478321	
	.589094	.105389	
	.602133	.445554	

Table 2. GCC C++ Diehard Test Results Part 2

Bitstream	**OPSO**	**OQSO**	**DNA**	**CNT 1sB**	**3D SPH**
.69693	.0282	.0000	1.0000	.774920	90260
.38483	.0372	.0000	.0000	.561840	72882
.76172	.0005	.0246	1.0000	.579544	03619
.28882	.1765	.1693	.0000	.426853	78345
.80420	.0013	.6130	.0008	.486733	69484
.70665	.0751	.0018	.0896	.669792	24076
.70585	.1400	.1971	.2083	.104575	69320
.92183	.0161	.0000	1.0000	.504758	84426
.23998	.0320	.0000	1.0000	.581833	89207
.76244	.1874	.0000	1.0000	.156646	19813
.22081	.0000	.0440	1.0000	.868356	10472
.65583	.0140	.2067	.0000	.554558	70579
.41734	.1332	.7550	.0002	.885662	38517
.44672	.0177	.5508	.0229	.725794	43728
.49503	.0034	.0110	.2383	.062972	05089
.79833	.7007	.0000	.1200	.969570	74042
.11519	.0517	.0000	1.0000	.990735	62511
.25474	.0005	.0000	.0000	.547370	96707
.90331	.0164	.0162	1.0000	.961884	63341
.52764	.0218	.0024	.0000	.934765	55377
	.0011	.9759	.0148	.805098	**.531503**
CNT 1s	.1362	.1162	.2832	.094834	KSTEST
.321054	.0478	.0000	.0001	.345356	
.463177		.0000	1.0000	.297482	**OSUM**
		.0000	1.0000	.619431	.157359
CDPARK		.0000	.0000		.840688
.205562		.0026	1.0000		.147470
.180558		.0001	.0000		.891995
.590298			.0371		.294137
.590298		**SQEEZE**	.0080		.603960
.108811		.317070	.0200		.471933
.246694					.036775
.291865		**RUNS**	**CRAPS WINS**		.573609
.934075		.079560	.765831		.552787
.500000		.964089	**CRAPS THROWS**		**.075029 KSTEST**
.136563		.488655	.104740		
.759000 KSTEST	**MIN .DIST**	.069318			
	.965807				

The next table shows the p-values produced by the Diehard suite of tests on the Mersenne generator (resulting log file is rmersey.log)..

Table 3. Mersenne Diehard Test Results Part 1

Bitstream	OPSO	OQSO	DNA	CNT 1sB	3D SPH
.76244	.9321	.7635	.5513	.906931	.35889
.98490	.4844	.5117	.5963	.103090	.21381
.93460	.0627	.7421	.4098	.268602	.62479
.82600	.9473	.6168	.5883	.666844	.01906
.84785	.3303	.9354	.8368	.849311	.68057
.41734	.4149	.3208	.5963	.021068	.85892
.56097	.1626	.9930	.2109	.851932	.71178
.58480	.0843	.6711	.1870	.203728	.83036
.00862	.8611	.8745	.6468	.941688	.08710
.11702	.9620	.8302	.6100	.497766	.27134
.55359	.5989	.2638	.6134	.577195	.06129
.26306	.8765	.1426	.6695	.277723	.74412
.30581	.7797	.2841	.7547	.992178	.00882
.41006	.6069	.8961	.7260	.845530	.98379
.57384	.4029	.6513	.9778	.466425	.35548
.29282	.3673	.1110	.2964	.147168	.88393
.29282	.0966	.1433	.8609	.482845	.30497
.87659	.9399	.8215	.9981	.255032	.26642
.24143	.9543	.5468	.9500	.705604	.63249
.07200	.4830	.4363	.4259	.294328	.95606
	.1362	.9033	.8888	.496514	**.291919**
CNT 1s	.8900	.5360	.8331	.151208	**KSTEST**
.668328	.8291	.2627	.3245	.729378	
.044188		.0437	.3213	.961371	**OSUM**
		.3605	.6302	.981730	.523352
CDPARK		.9201	.6357		.955821
.753306		.0837	.4926		.000946
.554479		.7789	.2903		.613807
.708135			.2704		.111499
.246694		**SQEEZE**	.0371		.564409
.192812		.508527	.7808		.929658
.914635					.326266
.000951		**RUNS**	**CRAPS WINS**		.427841
.625377		.675138	.854552		.442707
.536382		.678659	**CRAPS THROWS**		**.386011 KSTEST**
.015932	**MIN**	.129089	..040774		
.637284 KSTEST	**DIST**	.429987			
	.536019				

Table 4. Mersenne Diehard Test Results Part 2

Birthday Spacing	Bin. Rank 31x31		
.126317	.432115		
.954330	**Bin. Rank 32x32**		
.283559	.956891		
.743912	**Bin. Rank 6x8**		
.553217	.250621	.417829	.097548
.581776	.083185	.539246	.354312
.585145	.328546	.369348	.053574
.357520	.699241	.156892	**.239078 KSTEST**
.508808	.767513	.919956	
.196512 KSTEST	.887186	.479579	
OPERM5	.230224	.375273	
.148375	.541184	.718831	
.271223	.034657	.881512	
	.227694	.915275	
	.654906	.411544	

The next table shows the p-values produced by the Diehard suite of tests on the Stroustroup generator (resulting log file is rmersey.log).

Table 5. Stroustroup Diehard Test Results Part 1

Bitstream	OPSO	OQSO	DNA	CNT 1sB	3D SPH
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	.5548	1.000000	.00000
1.00000	1.00000	1.00000	.0000	1.000000	.00000
1.00000	1.00000	1.00000	1.0000	1.000000	.00000
	1.00000	1.00000	1.0000	1.000000	**1.000000 KSTEST**
CNT 1s	1.00000	1.00000	1.0000	1.000000	**KSTEST**
1.000000	1.00000	1.00000	1.0000	1.000000	
1.000000		1.00000	1.0000	1.000000	**OSUM**
		1.00000	.0000	1.000000	1.000000
CDPARK		1.00000	.0000		1.000000
.000000		1.00000	.0000		1.000000
.000000		1.00000	1.0000		1.000000
.000000			1.0000		1.000000
.000000		**SQEEZE**	1.0000		1.000000
.000000		RTE	1.0000		1.000000
.000000					1.000000
.000000		**RUNS**	**CRAPS WINS**		1.000000
.000000		.481974	RTE		1.000000
.000000		.816563	**CRAPS THROWS**		**1.000000 KSTEST**
.000000	**MIN**	.595713	RTE		
1.000000 KSTEST	**DIST**	.510214			
	1.000000				

RTE means the test produced a Runtime Error and had to be skipped.

Table 6. Stroustroup Diehard Test Results Part 2

Birthday Spacing	Bin. Rank 31x31		
1.000000	1.000000		
1.000000	**Bin. Rank 32x32**		
1.000000	1.000000		
1.000000	**Bin. Rank 6x8**		
1.000000	1.000000	1.000000	1.000000
1.000000	1.000000	1.000000	1.000000
1.000000	1.000000	1.000000	1.000000
1.000000	1.000000	1.000000	**1.000000 KSTEST**
1.000000	1.000000	1.000000	
1.00000 KSTEST	1.000000	1.000000	
OPERM5	1.000000	1.000000	
.676957	1.000000	1.000000	
.024215	1.000000	1.000000	
	1.000000	1.000000	
	1.000000	1.000000	

The next table shows the p-values produced by the Diehard suite of tests on the Knuth generator (resulting log file is rknut.log)..

Table 7. Knuth Diehard Test Results Part 1

Bitstream	OPSO	OQSO	DNA	CNT 1sB	3D SPH
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97455
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97528
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
1.00000	1.0000	1.0000	1.0000	1.000000	.97344
	1.0000	1.0000	1.0000	1.000000	**1.000000** KSTEST
	1.0000	1.0000	1.0000	1.000000	
	1.0000	1.0000	1.0000	1.000000	
		1.0000	1.0000	1.000000	
		1.0000	1.0000	1.000000	
		1.0000	1.0000		
		1.0000	1.0000		
		1.0000	1.0000		
			1.0000		
			1.0000		
			1.0000		

CNT 1s	SQEEZE	RUNS	CRAPS WINS	CRAPS THROWS	MIN DIST
1.000000	1.000000	1.000000	.295951	.********	1.000000
1.000000		1.000000			
		1.000000			
		1.000000			

CDPARK	OSUM
.000000	.759934
.000000	.980955
.000000	.978755
.000000	.962520
.000000	.946317
.000000	.940524
.000000	.981306
.000000	.919689
.000000	.951057
.000000	.965671
1.000000 KSTEST	**1.000000 KSTEST**

Table 8. Knuth Diehard Test Results Part 2

Birthday Spacing	OPERM5	Bin. Rank 31x31	Bin. Rank 32x32
1.000000	1.000000	1.000000	1.000000
1.000000	1.000000		
1.000000			
1.000000			
1.000000			
1.000000			
1.000000			
1.000000			
1.000000			
1.000000 KSTEST			

Bin. Rank 6x8		
1.000000	1.000000	1.000000
1.000000	1.000000	1.000000
1.000000	1.000000	1.000000
1.000000	1.000000	**1.000000 KSTEST**
1.000000	1.000000	
1.000000	1.000000	
1.000000	1.000000	
1.000000	1.000000	
1.000000	1.000000	
1.000000	1.000000	
1.000000	1.000000	

4. DIEHARD ANALYSIS

A note provided in the Diehard log files says

> NOTE: Most of the tests in DIEHARD return a p-value,
> should be uniform on [0,1) if the input file contain
> independent random bits. Those p-values are obtain
> p=F(X), where F is the assumed distribution of the s
> random variable X---often normal. But that assumed F
> an asymptotic approximation, for which the fit will
> in the tails. Thus you should not be surprised with
> occasional p-values near 0 or 1, such as .0012 or .9
> When a bit stream really FAILS BIG, you will get p's
> 1 to six or more places. By all means, do not, as a
> Statistician might, think that a p < .025 or p> .975
> that the RNG has "failed the test at the .05 level".
> p's happen among the hundreds that DIEHARD produces,
> with good RNG's. So keep in mind that " p happens".

For the C++ PRNG, the smallest p-value was 0 and the largest was 1.000. This PRNG passed most Diehard tests with some failures in OPSO, OQSO and mainly DNA tests.

The Mersenne Twister PRNG passed all the Diehard tests successfully. The smallest p-value was .000951 and the largest p-value was .9930. The only areas with p-values over .975 were OPSO, OQSO, DNA, 3D Spheres and Count-The-1's for specific bytes. Areas with p-values under 0.025 were OSUM, CDPARK and 3D Spheres.

Diehard gave a runtime error when running the Squeeze and Craps tests on the Stroustroup PRNG. The only tests which it passed were OPERM5 and RUNS. P-values ranged from 0 to 1.

The Knuth linear congruential generator passed the 3D Spheres test (except for the KSTEST (Kolmogorov-Smirnov test) and Craps Wins (for Craps Throws Diehard outputted just '*****' instead of a p-value, it is not clear whether this is some overflow or underflow).

5. CONCLUSION

The Mersenne Twister PRNG clearly emerged as the best PRNG from the ones tested and the one most likely to be suitable for use during the critical initalisation phase of a GA. For the normal GA run after initialisation, however the standard C++ PRNG may be used. Further research may give a more clear picture on the effects of a bad PRNG (one that fails the tests) with regards to the running of the GA (mainly post-initialisation), for some particular problem and whether this depends on population size.

A copy of the random binary files generated, the Diehard logfiles and other sources will be made available on the GAGENES website at http://www.gagenes.com.

6. REFERENCES

[1] Cantu-Paz, E. On Random Numbers and the Performance of Genetic Algorithms.

[2] Knuth, D.E. *The Art of Computer Programming*. Addison-Wesley, 1997.

[3] Marsaglia, G., Tsang, Wai Wan Some difficult-to-pass tests of randomness. *Journal of Statistics Software*, 7(3), 2003.

[4] Matsumoto, M., Nishimura, T Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. on Modeling and Computer Simulation*, *8* (1). 3-30.

[5] Meysenburg, M.M. The Effect of Pseudo-Random Number Generator Quality on the Performance of a Simple Genetic Algorithm *Computer Science*, University of Idaho, 1997, 82.

[6] Meysenburg, M.M., Foster, James, Randomness and GA Performance, Revisited. in *Proceedings of the Genetic and Evolutionary Computation Conference*, (San Francisco, California, 1999), Morgan Kauffman Publishers, 425-432.

[7] Stroustroup, B. *The C++ Programming Language Special Edition*. Addison-Wesley, 2005.

Improving Polygonal Hybrid Systems Reachability Analysis through the use of the Phase Portrait

Gordon J. Pace
Department of Computer Science and AI
University of Malta, Malta
gordon.pace@um.edu.mt

Gerardo Schneider
Department of Informatics
University of Oslo, Norway
gerardo@ifi.uio.no

ABSTRACT

Polygonal hybrid systems (SPDI) are a subclass of planar hybrid automata which can be represented by piecewise constant dierential inclusions. The computation of certain objects of the phase portrait of an SPDI, namely the viability, controllability, invariance kernels and semi-separatrix curves have been shown to be eciently decidable. On the other hand, although the reachability problem for SPDIs is known to be decidable, its complexity makes it unfeasible on large systems. We summarise our recent results on the use of the SPDI phase portraits for improving reachability analysis by (i) state-space reduction and (ii) decomposition techniques of the state space, enabling compositional parallelisation of the analysis. Both techniques contribute to increasing the feasability of reachability analysis on large SPDI systems.

1. INTRODUCTION

Hybrid systems combining discrete and continuous dynamics arise as mathematical models of various artificial and natural systems, and as approximations to complex continuous systems. They have been used in various domains, including avionics, robotics and bioinformatics. Reachability analysis has been the principal research question in the verification of hybrid systems, even if it is a well-known result that for most subclasses of hybrid systems most verification questions are undecidable. Various decidable subclasses have, subsequently, been identified, including timed [1] and rectangular automata [10], hybrid automata with linear vector fields [11], piecewise constant derivative systems (PCDs) [12] and polygonal hybrid systems (SPDIs) [4].

Compared to reachability verification, qualitative analysis of hybrid systems is a relatively neglected area [8, 9, 13, 19, 22]. Typical qualitative questions include: 'Are there 'sink' regions where a trajectory can never leave once it enters the region?' and 'Are there regions in which every point in the region is reachable from every other?'. The collection of objects in a system satisfying these and similar properties is called the *phase portrait* of the system.

Defining and constructing phase portraits of hybrid systems has been directly addressed for PCDs in [13], and for SPDIs in [5]. Given a cycle on a SPDI, the *viability* kernel is the largest set of points in the cycle which may loop forever within the cycle. The *controllability* kernel is the largest set of strongly connected points in the cycle (such that any point in the set may be reached from any other). An *invariant set* is a set of points such that each point must keep rotating within the set forever, and the *invariance kernel* is the largest such set. Algorithms for computing these kernels have been presented in [5, 21] and implemented in the tool set SPeeDI$^+$ [14].

Given the non-compositional nature of hybrid systems, decomposing SPDIs to reduce the state space and to distribute the reachability algorithms is a challenging task. A qualitative analysis of hybrid systems does, however, provide useful information for partitioning the state-space in independent subspaces, thus helping in achieving compositional analysis.

In this paper we summarise and combine some recent results [17, 16] we have obtained, showing how kernels can be used to improve the reachability analysis of SPDIs. We use kernel information to (i) reduce the number of states of the SPDI graph, based on topological properties of the plane (and in particular, those of SPDIs); and (ii) partition the reachability questions in a compositional manner, dividing the problem into independent smaller ones and combining the partial results to answer the original question, hence enabling parallelization with minimal communication costs.

2. THEORETICAL BACKGROUND

We summarize here the main definitions and results about SPDIs; for a more detailed description refer to [20]. A (positive) *affine* function $f : \mathbb{R} \to \mathbb{R}$ is such that $f(x) = ax + b$ with $a > 0$. An *affine multivalued* function $F : \mathbb{R} \to 2^{\mathbb{R}}$, denoted $F = \langle f_l, f_u \rangle$, is defined by $F(x) = \langle f_l(x), f_u(x) \rangle$ where f_l and f_u are affine and $\langle \cdot, \cdot \rangle$ denotes an interval. For notational convenience, we do not make explicit whether intervals are open, closed, left-open or right-open, unless required for comprehension. For an interval $I = \langle l, u \rangle$ we have that $F(\langle l, u \rangle) = \langle f_l(l), f_u(u) \rangle$. The *inverse* of F is defined by $F^{-1}(x) = \{y \mid x \in F(y)\}$. The *universal inverse* of F is defined by $\tilde{F}^{-1}(I) = I'$ where I' is the greatest non-empty interval satisfying $\forall x \in I' \cdot F(x) \subseteq I$.

Clearly, $F^{-1} = \langle f_u^{-1}, f_l^{-1} \rangle$ and $\tilde{F}^{-1} = \langle f_l^{-1}, f_u^{-1} \rangle$, provided that $\langle f_l^{-1}, f_u^{-1} \rangle \neq \emptyset$.

A *truncated affine multivalued* function (TAMF) $\mathcal{F} : \mathbb{R} \to 2^{\mathbb{R}}$ is defined by an affine multivalued function F and in-

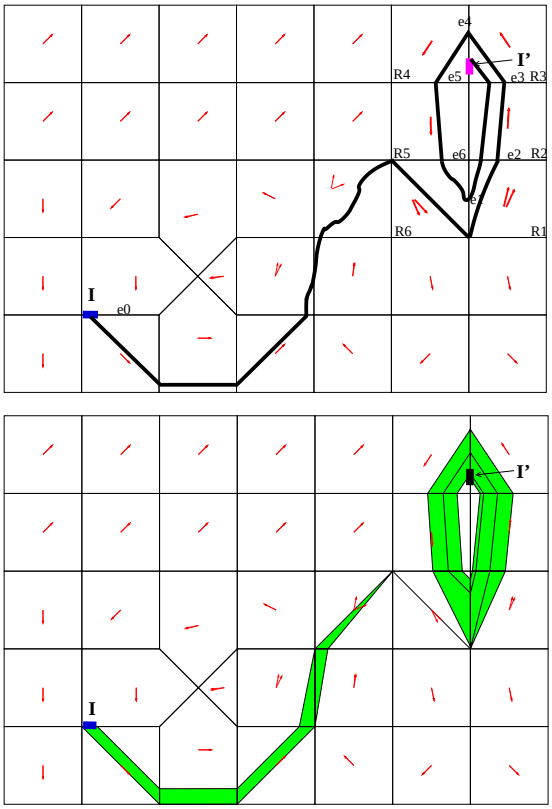

Figure 1: (a) An SPDI and its trajectory segment; (b) Reachability analysis

tervals $S \subseteq \mathbb{R}^+$ and $J \subseteq \mathbb{R}^+$ as follows: $\mathcal{F}(x) = F(x) \cap J$ if $x \in S$, otherwise $\mathcal{F}(x) = \emptyset$. For convenience we write $\mathcal{F}(x) = F(\{x\} \cap S) \cap J$. For an interval I, $\mathcal{F}(I) = F(I \cap S) \cap J$ and $\mathcal{F}^{-1}(I) = F^{-1}(I \cap J) \cap S$. The *universal inverse* of $\mathcal{F}$ is defined by $\tilde{\mathcal{F}}^{-1}(I) = I'$ if and only if I' is the greatest non-empty interval such that for all $x \in I'$, $F(x) \subseteq I$ and $F(x) = \mathcal{F}(x)$. We say that $\mathcal{F}$ is *normalized* if $S = \mathsf{Dom}(\mathcal{F}) = \{x \mid F(x) \cap J \neq \emptyset\}$ (thus, $S \subseteq F^{-1}(J)$) and $J = \mathsf{Im}(\mathcal{F}) = \mathcal{F}(S)$.

It can be proved [4], that TAMFs are closed under composition.

Theorem 1 *The composition of TAMFs $\mathcal{F}_1(I) = F_1(I \cap S_1) \cap J_1$ and $\mathcal{F}_2(I) = F_2(I \cap S_2) \cap J_2$, is the TAMF $(\mathcal{F}_2 \circ \mathcal{F}_1)(I) = \mathcal{F}(I) = F(I \cap S) \cap J$, where $F = F_2 \circ F_1$, $S = S_1 \cap F_1^{-1}(J_1 \cap S_2)$ and $J = J_2 \cap F_2(J_1 \cap S_2)$.* □

2.1 SPDIs

An *angle* $\angle_{\mathbf{a}}^{\mathbf{b}}$ on the plane, defined by two non-zero vectors $\mathbf{a}, \mathbf{b}$, is the set of all positive linear combinations $\mathbf{x} = \alpha\,\mathbf{a} + \beta\,\mathbf{b}$, with $\alpha, \beta \geq 0$, and $\alpha + \beta > 0$. We will assume that $\mathbf{b}$ is situated in the counter-clockwise direction from $\mathbf{a}$.

A *polygonal hybrid system*[1] (SPDI) is a finite partition $\mathbb{P}$ of the plane into convex polygonal sets, such that for each $P \in \mathbb{P}$ we have two vectors $\mathbf{a}_P$ and $\mathbf{b}_P$. Let $\phi(P) = \angle_{\mathbf{a}_P}^{\mathbf{b}_P}$. The SPDI is determined by $\dot{\mathbf{x}} \in \phi(P)$ for $\mathbf{x} \in P$.

Let $E(P)$ be the set of edges of P. We say that e is an *entry* of P if for all $\mathbf{x} \in e$ and for all $\mathbf{c} \in \phi(P)$, $\mathbf{x} + \mathbf{c}\epsilon \in P$ for some $\epsilon > 0$. We say that e is an *exit* of P if the same condition holds for some $\epsilon < 0$. We denote by $in(P) \subseteq E(P)$ the set of all entries of P and by $out(P) \subseteq E(P)$ the set of all exits of P.

Assumption 1 *All the edges in $E(P)$ are either entries or exits, that is, $E(P) = in(P) \cup out(P)$.*

Reachability for SPDIs is decidable provided the above assumption holds [4]; without such assumption it is not know whether reachability is decidable.

A *trajectory segment* of an SPDI is a continuous function $\xi : [0, T] \to \mathbb{R}^2$ which is smooth everywhere except in a discrete set of points, and such that for all $t \in [0, T]$, if $\xi(t) \in P$ and $\dot{\xi}(t)$ is defined then $\dot{\xi}(t) \in \phi(P)$. The *signature*, denoted $\mathsf{Sig}(\xi)$, is the ordered sequence of edges traversed by the trajectory segment, that is, $e_1, e_2, \ldots$, where $\xi(t_i) \in e_i$ and $t_i < t_{i+1}$. If $T = \infty$, a trajectory segment is called a *trajectory*.

Example 1 *Consider the SPDI illustrated in Fig. 1-(a). For sake of simplicity we will only show the dynamics associated to regions R_1 to R_6 in the picture. For each region R_i, $1 \leq i \leq 6$, there is a pair of vectors $(\mathbf{a}_i, \mathbf{b}_i)$, where: $\mathbf{a}_1 = (45, 100), \mathbf{b}_1 = (1, 4)$, $\mathbf{a}_2 = \mathbf{b}_2 = (1, 10)$, $\mathbf{a}_3 = \mathbf{b}_3 = (-2, 3)$, $\mathbf{a}_4 = \mathbf{b}_4 = (-2, -3)$, $\mathbf{a}_5 = \mathbf{b}_5 = (1, -15)$, $\mathbf{a}_6 = (1, -2), \mathbf{b}_6 = (1, -1)$. A trajectory segment starting on interval $I \subset e_0$ and finishing in interval $I' \subseteq e_4$ is depicted.* ■

We say that a signature σ is *feasible* if and only if there exists a trajectory segment ξ with signature σ, i.e., $\mathsf{Sig}(\xi) = \sigma$. From this definition, it immediately follows that extending an unfeasible signature can never make it feasible:

Proposition 1 *If a signature σ is not feasible, then neither is any extension of the signature — for any signatures σ' and σ'', the signature $\sigma'\sigma\sigma''$ is not feasible.* □

Given an SPDI $\mathcal{S}$, let $\mathcal{E}$ be the set of edges of $\mathcal{S}$, then we can define a graph $\mathcal{G}_\mathcal{S}$ where nodes correspond to edges of $\mathcal{S}$ and such that there exists an arc from one node to another if there exists a trajectory segment from the first edge to the second one without traversing any other edge. More formally: Given an SPDI $\mathcal{S}$, the *underlying graph of* $\mathcal{S}$ (or simply the *graph of* $\mathcal{S}$), is a graph $\mathcal{G}_\mathcal{S} = (N_\mathcal{G}, A_\mathcal{G})$, with $N_\mathcal{G} = \mathcal{E}$ and $A_\mathcal{G} = \{(e, e') \mid \exists \xi, t\,.\,\xi(0) \in e \wedge \xi(t) \in e' \wedge \mathsf{Sig}(\xi) = ee'\}$. We say that a sequence $e_0 e_1 \ldots e_k$ of nodes in $\mathcal{G}_\mathcal{S}$ is a *path* whenever $(e_i, e_{i+1}) \in A_\mathcal{G}$ for $0 \leq i \leq k-1$.

The following lemma shows the relation between edge signatures in an SPDI and paths in its corresponding graph.

Lemma 1 *If ξ is a trajectory segment of $\mathcal{S}$ with edge signature $\mathsf{Sig}(\xi) = \sigma = e_0 \ldots e_p$, it follows that σ is a path in $\mathcal{G}_\mathcal{S}$.* □

Note that the converse of the above lemma is not true in general. It is possible to find a counter-example where there exists a path from node e to e', but no trajectory from edge e to edge e' in the SPDI.

[1] In the literature the names *polygonal differential inclusion* and *simple planar differential inclusion* have been used to describe the same systems.

2.2 Successors and Predecessors

Given an SPDI, we fix a one-dimensional coordinate system on each edge to represent points laying on edges [4]. For notational convenience, we indistinctly use letter e to denote the edge or its one-dimensional representation. Accordingly, we write $\mathbf{x} \in e$ or $x \in e$, to mean "point $\mathbf{x}$ in edge e with coordinate x in the one-dimensional coordinate system of e". The same convention is applied to sets of points of e represented as intervals (e.g., $\mathbf{x} \in I$ or $x \in I$, where $I \subseteq e$) and to trajectories (e.g., "ξ starting in x" or "ξ starting in $\mathbf{x}$").

Now, let $P \in \mathbb{P}$, $e \in in(P)$ and $e' \in out(P)$. For $I \subseteq e$, $\mathsf{Succ}_{e,e'}(I)$ is the set of all points in e' reachable from some point in I by a trajectory segment $\xi : [0,t] \to \mathbb{R}^2$ in P (i.e., $\xi(0) \in I \wedge \xi(t) \in e' \wedge \mathsf{Sig}(\xi) = ee'$). $\mathsf{Succ}_{e,e'}$ is a TAMF [4].

Example 2 *Let $e_1, \ldots, e_6$ be as in Fig. 1-(a), where all the edges have local coordinates over $[0, 10]$, and $I = [l, u]$. We assume a one-dimensional coordinate system. We show only the first and last edge-to-edge TAMF of the cycle:*

$$\begin{array}{lll} F_{e_1e_2}(I) = \left[\frac{l}{4}, \frac{9}{20}u\right], & S_1 = [0, 10], & J_1 = \left[0, \frac{9}{2}\right] \\ F_{e_6e_1}(I) = [l, 2u], & S_6 = [0, 10], & J_6 = [0, 10] \end{array}$$

with $\mathsf{Succ}_{e_ie_{i+1}}(I) = F_{e_ie_{i+1}}(I \cap S_i) \cap J_i$, for $1 \leq i \leq 6$; S_i and J_i are computed as shown in Theorem 1. ■

Given a sequence $w = e_1, e_2, \ldots, e_n$, since TAMFs are closed under composition, the successor of I along w, defined as $\mathsf{Succ}_w(I) = \mathsf{Succ}_{e_{n-1},e_n} \circ \cdots \circ \mathsf{Succ}_{e_1,e_2}(I)$, is a TAMF.

Example 3 *Let $\sigma = e_1 \cdots e_6 e_1$. We have that $\mathsf{Succ}_\sigma(I) = F(I \cap S_\sigma) \cap J_\sigma$, where: $F(I) = [\frac{l}{4} + \frac{1}{3}, \frac{9}{10}u + \frac{2}{3}]$, with $S_\sigma = [0, 10]$ and $J_\sigma = [\frac{1}{3}, \frac{29}{3}]$.* ■

For $I \subseteq e'$, $\mathsf{Pre}_{e,e'}(I)$ is the set of points in e that can reach a point in I by a trajectory segment in P. The $\forall$-*predecessor* $\widetilde{\mathsf{Pre}}(I)$ is defined in a similar way to $\mathsf{Pre}(I)$ using the universal inverse instead of just the inverse: For $I \subseteq e'$, $\widetilde{\mathsf{Pre}}_{ee'}(I)$ is the set of points in e such that *any* successor of such points are in I by a trajectory segment in P. Both definitions can be extended straightforwardly to signatures $\sigma = e_1 \cdots e_n$: $\mathsf{Pre}_\sigma(I)$ and $\widetilde{\mathsf{Pre}}_\sigma(I)$. The successor operator thus has two "inverse" operators.

2.3 Qualitative Analysis of Simple Edge-Cycles

Let $\sigma = e_1 \cdots e_k e_1$ be a simple edge-cycle, i.e., $e_i \neq e_j$ for all $1 \leq i \neq j \leq k$. Let $\mathsf{Succ}_\sigma(I) = F(I \cap S_\sigma) \cap J_\sigma$ with $F = \langle f_l, f_u \rangle$ (we suppose that this representation is normalized). We denote by $\mathcal{D}_\sigma$ the one-dimensional discrete-time dynamical system defined by Succ_σ, that is $x_{n+1} \in \mathsf{Succ}_\sigma(x_n)$.

Assumption 2 *None of the two functions f_l, f_u is the identity.*

Without the above assumption the results are still valid but need a special treatment making the presentation more complicated.

Let l^* and u^* be the fixpoints[2] of f_l and f_u, respectively, and $S_\sigma \cap J_\sigma = \langle L, U \rangle$. A simple cycle is of one of the following types [4]: STAY, the cycle is not abandoned neither by the leftmost nor the rightmost trajectory, that is, $L \leq l^* \leq u^* \leq U$; DIE, the rightmost trajectory exits the cycle through the left (consequently the leftmost one also exits) or the leftmost trajectory exits the cycle through the right (consequently the rightmost one also exits), that is, $u^* < L \vee l^* > U$; EXIT-BOTH, the leftmost trajectory exits the cycle through the left and the rightmost one through the right, that is, $l^* < L \wedge u^* > U$; EXIT-LEFT, the leftmost trajectory exits the cycle (through the left) but the rightmost one stays inside, that is, $l^* < L \leq u^* \leq U$; EXIT-RIGHT, the rightmost trajectory exits the cycle (through the right) but the leftmost one stays inside, that is, $L \leq l^* \leq U < u^*$.

[2]The fixpoint x^* is the solution of $f(x^*) = x^*$, where $f(\cdot)$ is positive affine.

Example 4 *Let $\sigma = e_1 \cdots e_6 e_1$. Then, $S_\sigma \cap J_\sigma = \langle L, U \rangle = [\frac{1}{3}, \frac{29}{3}]$. The fixpoints from Example 3 are $\frac{1}{3} < l^* = \frac{11}{25} < u^* = \frac{20}{3} < \frac{29}{3}$. Thus, σ is a STAY.* ■

Any trajectory that enters a cycle of type DIE will eventually quit it after a finite number of turns. If the cycle is of type STAY, all trajectories that happen to enter it will keep turning inside it forever. In all other cases, some trajectories will turn for a while and then exit, and others will continue turning forever. This information is crucial for proving decidability of the reachability problem.

Example 5 *Consider the SPDI of Fig. 1-(a). Fig. 1-(b) shows part of the reach set of the interval $[8, 10] \subset e_0$, answering positively to the reachability question: Is $[1, 2] \subset e_4$ reachable from $[8, 10] \subset e_0$? Fig. 1-(b) has been automatically generated by the SPeeDI toolbox we have developed for reachability analysis of SPDIs [2, 14].* ■

2.4 Reachability Analysis

It has been shown that reachability is decidable for SPDIs. Proof of the decidability result is constructive, giving an algorithmic procedure $Reach(\mathcal{S}, e, e')$ based on a depth-first search algorithm. An alternative breadth-first search algorithm which can deal with multiple edges has been presented in [15].

Theorem 2 ([4]) *The reachability problem for SPDIs is decidable.* □

An edgelist is a set of intervals of edges. Given edgelists I and I', we denote the reachability of (some part of) I' from (some part of) I as $I \xrightarrow{\mathcal{S}} I'$. Clearly, using the decidability result on edge intervals, reachability between edgelists is decidable. Although decidability may be point-to-point, edge-to-edge, edgelist-to-edgelist and region-to-region, in the rest of this paper, we will only talk about edgelist reachability.

Example 6 *Consider the SPDI of Fig. 1-(a). Fig. 1-(b) shows part of the reach set of the interval $[8, 10] \subset e_0$, answering positively to the reachability question: Is $[1, 2] \subset e_4$ reachable from $[8, 10] \subset e_0$? Fig. 1-(b) has been automatically generated by the SPeeDI toolbox [14] we have developed for reachability analysis of SPDIs based on the results of [4].* ■

2.5 Kernels

We present now how to compute the invariance, controllability and viability kernels of an SPDI. Proofs are omitted

but for further details, refer to [5] and [21]. In the following, for σ a cyclic signature, we define $K_\sigma \subseteq \mathbb{R}^2$ as follows: $K_\sigma = \bigcup_{i=1}^{k}(int(P_i) \cup e_i)$ where P_i is such that $e_{i-1} \in in(P_i)$, $e_i \in out(P_i)$ and $int(P_i)$ is P_i's interior.

2.5.1 Viability Kernel

We now recall the definition of *viability kernel* [6]. A trajectory ξ is *viable* in K if $\xi(t) \in K$ for all $t \geq 0$. K is a *viability domain* if for every $\mathbf{x} \in K$, there exists at least one trajectory ξ, with $\xi(0) = \mathbf{x}$, which is viable in K. The *viability kernel* of K, denoted $\mathsf{Viab}(K)$, is the largest viability domain contained in K.

For $I \subseteq e_1$ we define $\overline{\mathsf{Pre}}_\sigma(I)$ to be the set of all $\mathbf{x} \in \mathbb{R}^2$ for which there exists a trajectory segment ξ starting in $\mathbf{x}$, that reaches some point in I, such that $\mathsf{Sig}(\xi)$ is a suffix of $e_2 \ldots e_k e_1$. It is easy to see that $\overline{\mathsf{Pre}}_\sigma(I)$ is a polygonal subset of the plane which can be calculated using the following procedure. We start by defining $\overline{\mathsf{Pre}}_e(I) = \{\mathbf{x} \mid \exists \xi : [0,t] \rightarrow \mathbb{R}^2, t > 0 \;.\; \xi(0) = \mathbf{x} \wedge \xi(t) \in I \wedge \mathsf{Sig}(\xi) = e\}$ and apply this operation k times: $\overline{\mathsf{Pre}}_\sigma(I) = \bigcup_{i=1}^{k} \overline{\mathsf{Pre}}_{e_i}(I_i)$ with $I_1 = I$, $I_k = \mathsf{Pre}_{e_k,e_1}(I_1)$ and $I_i = \mathsf{Pre}_{e_i,e_{i+1}}(I_{i+1})$, for $2 \leq i \leq k-1$.

The following result provides a non-iterative algorithmic procedure for computing the viability kernel of K_σ on an SPDI:

Theorem 3 *If σ is a DIE cycle, then $\mathsf{Viab}(K_\sigma) = \emptyset$, otherwise $\mathsf{Viab}(K_\sigma) = \overline{\mathsf{Pre}}_\sigma(S_\sigma)$.* □

Example 7 *Fig. 2-(a) shows all the viability kernels of the SPDI given in Example 1. There are 4 cycles with viability kernels — in the picture two of the kernels are overlapping.* ■

2.5.2 Controllability Kernel

We say K is *controllable* if for any two points $\mathbf{x}$ and $\mathbf{y}$ in K there exists a trajectory segment ξ starting in $\mathbf{x}$ that reaches an arbitrarily small neighborhood of $\mathbf{y}$ without leaving K. More formally: A set K is controllable if $\forall \mathbf{x}, \mathbf{y} \in K, \forall \delta > 0, \exists \xi : [0,t] \rightarrow \mathbb{R}^2, t > 0 \;.\; (\xi(0) = \mathbf{x} \wedge |\xi(t) - \mathbf{y}| < \delta \wedge \forall t' \in [0,t] \;.\; \xi(t') \in K)$. The *controllability kernel* of K, denoted $\mathsf{Cntr}(K)$, is the largest controllable subset of K.

For a given cyclic signature σ, we define $\mathcal{C}_\mathcal{D}(\sigma)$ as follows:

$$\mathcal{C}_\mathcal{D}(\sigma) = \begin{cases} \langle L, U \rangle & \text{if } \sigma \text{ is EXIT-BOTH} \\ \langle L, u^* \rangle & \text{if } \sigma \text{ is EXIT-LEFT} \\ \langle l^*, U \rangle & \text{if } \sigma \text{ is EXIT-RIGHT} \\ \langle l^*, u^* \rangle & \text{if } \sigma \text{ is STAY} \\ \emptyset & \text{if } \sigma \text{ is DIE} \end{cases} \quad (1)$$

For $I \subseteq e_1$ let us define $\overline{\mathsf{Succ}}_\sigma(I)$ as the set of all points $\mathbf{y} \in \mathbb{R}^2$ for which there exists a trajectory segment ξ starting in some point $x \in I$, that reaches $\mathbf{y}$, such that $\mathsf{Sig}(\xi)$ is a prefix of $e_1 \ldots e_k$. The successor $\overline{\mathsf{Succ}}_\sigma(I)$ is a polygonal subset of the plane which can be computed similarly to $\overline{\mathsf{Pre}}_\sigma(I)$. Define $\mathcal{C}(\sigma) = (\overline{\mathsf{Succ}}_\sigma \cap \overline{\mathsf{Pre}}_\sigma)(\mathcal{C}_\mathcal{D}(\sigma))$. We compute the controllability kernel of K_σ as follows:

Theorem 4 $\mathsf{Cntr}(K_\sigma) = \mathcal{C}(\sigma)$. □

Example 8 *Fig. 2-(b) shows all the controllability kernels of the SPDI given in Example 1. There are 4 cycles with controllability kernels — in the picture two of the kernels are overlapping.* ■

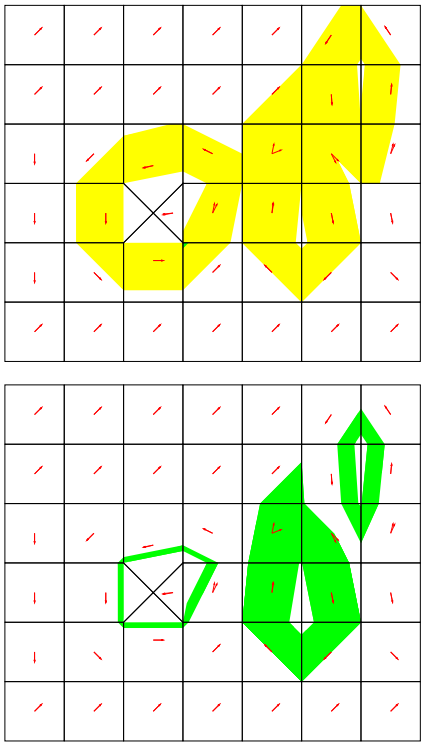

Figure 2: (a) Viability kernels; (b) Controllability kernels

The following result which relates controllability and viability kernels, states that the viability kernel of a given cycle is the local basin of attraction of the corresponding controllability kernel.

Proposition 2 *Any viable trajectory in K_σ converges to $\mathsf{Cntr}(K_\sigma)$.* □

Let $\mathsf{Cntr}^l(K_\sigma)$ be the closed curve obtained by taking the leftmost trajectory and $\mathsf{Cntr}^u(K_\sigma)$ be the closed curve obtained by taking the rightmost trajectory which can remain inside the controllability kernel. In other words, $\mathsf{Cntr}^l(K_\sigma)$ and $\mathsf{Cntr}^u(K_\sigma)$ are the two polygons defining the controllability kernel.

A non-empty controllability kernel $\mathsf{Cntr}(K_\sigma)$ of a given cyclic signature σ partitions the plane into three disjoint subsets: (1) the controllability kernel itself, (2) the set of points limited by $\mathsf{Cntr}^l(K_\sigma)$ (and not including $\mathsf{Cntr}^l(K_\sigma)$) and (3) the set of points limited by $\mathsf{Cntr}^u(K_\sigma)$ (and not including $\mathsf{Cntr}^u(K_\sigma)$). We define the *inner* of $\mathsf{Cntr}(K_\sigma)$ (denoted by $\mathsf{Cntr}_{in}(K_\sigma)$) to be the subset defined by (2) above if the cycle is counter-clockwise or to be the subset defined by (3) if it is clockwise. The *outer* of $\mathsf{Cntr}(K_\sigma)$ (denoted by $\mathsf{Cntr}_{out}(K_\sigma)$) is defined to be the subset which is not the inner nor the controllability itself. Note that an edge in the SPDI may intersect a controllability kernel. In such cases, we can generate a different SPDI, with the same dynamics but with the edge split into parts, such that each part is completely inside, on or outside the kernel. Although the signatures will obviously change, it is easy to prove that the behaviour of the SPDI remains identical to the original. In the rest of the paper, we will assume that all edges are either

completely inside, on or completely outside the kernels. We note that in practice splitting is not necessary since we can just consider parts of edges.

Proposition 3 *Given two edges e and e', one lying completely inside a controllability kernel, and the other outside or on the same controllability kernel, such that ee' is feasible, then there exists a point on the controllability kernel, which is reachable from e and from which e' is reachable.* □

2.5.3 Invariance Kernel

In general, an *invariant set* is a set of points such that for any point in the set, every trajectory starting in such point remains in the set forever and the *invariance kernel* is the largest of such sets. In particular, for an SPDI, given a cyclic signature, an *invariant set* is a set of points which keep rotating in the cycle forever and the *invariance kernel* is the largest of such sets. More formally: A set K is said to be *invariant* if for any $x \in K$ there exists at least one trajectory starting in it and every trajectory starting in x is viable in K. Given a set K, its largest invariant subset is called the *invariance kernel* of K and is denoted by $\mathsf{Inv}(K)$. We need some preliminary definitions before showing how to compute the kernel. The *extended* $\forall$*-predecessor* of an output edge e of a region R is the set of points in R such that every trajectory segment starting in such point reaches e without traversing any other edge. More formally, let R be a region and e be an edge in $out(R)$, then the *e-extended* $\forall$*-predecessor* of I, $\widetilde{\overline{\mathsf{Pre}}}_e(I)$ is defined as: $\widetilde{\overline{\mathsf{Pre}}}_e(I) = \{\mathbf{x} \mid \forall \xi \,.\, (\xi(0) = \mathbf{x} \Rightarrow \exists t \geq 0 \,.\, (\xi(t) \in I \wedge \mathsf{Sig}(\xi[0,t]) = e))\}$. It is easy to see that $\widetilde{\overline{\mathsf{Pre}}}_\sigma(I)$ is a polygonal subset of the plane which can be calculated using a similar procedure as for $\overline{\mathsf{Pre}}_\sigma(I)$. We compute the invariance kernel of K_σ as follows:

Theorem 5 *If σ is STAY then $\mathsf{Inv}(K_\sigma) = \widetilde{\overline{\mathsf{Pre}}}_\sigma(\widetilde{\overline{\mathsf{Pre}}}_\sigma(J_\sigma))$, otherwise it is $\emptyset$.* □

Example 9 *Fig. 3-(a) shows the unique invariance kernel of the SPDI given in Example 1.* ■

An interesting property of invariance kernels is that the limits are included in the invariance kernel, i.e. $[l^*, u^*] \subseteq \mathsf{Inv}(K_\sigma)$. In other words:

Proposition 4 *The set delimited by the polygons defined by the interval $[l^*, u^*]$ is an invariance set of STAY cycles.* □

The following result relates controllability and invariance kernels.

Proposition 5 *If σ is STAY then $\mathsf{Cntr}(K_\sigma) \subseteq \mathsf{Inv}(K_\sigma)$.* □

Example 10 *Fig. 3-(b) shows the viability, controllability and invariance kernels of the SPDI given in Example 1. For any point in the viability kernel of a cycle there exists a trajectory which will converge to its controllability kernel (proposition 2). It is possible to see in the picture that $\mathsf{Cntr}(\cdot) \subset \mathsf{Inv}(.)$ (proposition 5). All the above pictures has been obtained with the toolbox SPeeDI$^+$ [14].* ■

In a similar way as for the controllability kernel, we define $\mathsf{Inv}^l(K_\sigma)$ and $\mathsf{Inv}^u(K_\sigma)$.

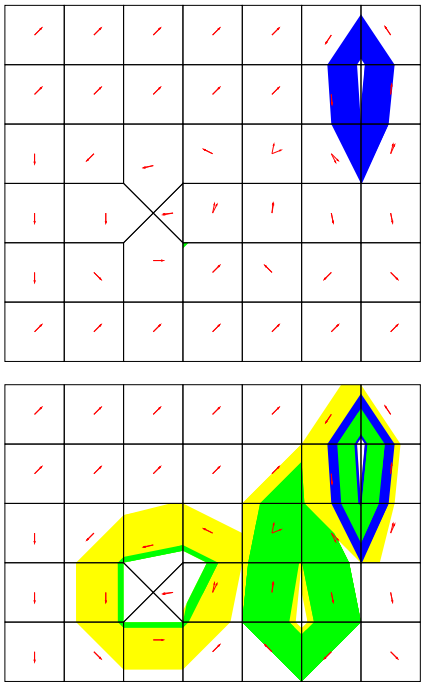

Figure 3: (a) Invariance kernel; (b) All the kernels

2.5.4 Kernel Properties

Controllability and viability kernels can be related together in the following manner.

Definition 1 *Given a controllability kernel C (of a loop σ — $C = \mathsf{Cntr}(K_\sigma)$), then let C^+ be the related viability kernel ($C^+ = \mathsf{Viab}(K_\sigma)$), C_{in} be the inside of the kernel, and C_{out} be the outside.*

Proposition 3 in [18] gives conditions for feasible trajectories traversing controllability kernels. The following is a generalization of such result:

Proposition 6 *Given two edges e and e', one lying completely inside a kernel, and the other outside or on the same kernel, such that ee' is feasible, then there exists a point on the kernel, which is reachable from e and from which e' is reachable.* □

The following corollary follows from [18, Proposition 2], asserting that the controllability kernel is the local basin of attraction of the viability kernel:

Corollary 1 *Given an controllability kernel C, and related viability kernel C^+, then for any $e \subseteq C^+$, $e' \subseteq C$, there exists a feasible path $e\sigma e'$.* □

2.6 Semi-Separatrix Curves

In this section we define the notion of *separatrix curves*, which are curves dissecting the plane into two mutually non-reachable subsets, and *semi-separatrix curves* which can only be crossed in one direction. All the proofs of this and forthcoming sections may be found in [17]. We start by defining these notions independently of SPDIs.

Definition 2 *Let $K \subseteq \mathbb{R}^2$. A* separatrix *in K is a closed curve γ partitioning K into three sets K_A, K_B and γ itself, such that K_A, K_B and γ are pairwise disjoint, $K = K_A \cup K_B \cup \gamma$ and the following conditions hold: (1) For any point $\mathbf{x}_0 \in K_A$ and trajectory ξ, with $\xi(0) = \mathbf{x}_0$, there is no t such that $\xi(t) \in K_B$; and (2) For any point $\mathbf{x}_0 \in K_B$ and trajectory ξ, with $\xi(0) = \mathbf{x}_0$, there is no t such that $\xi(t) \in K_A$. If only one of the above conditions holds then we say that the curve is a* semi-separatrix. *If only condition 1 holds, then we say that K_A is the* inner *of γ (written γ_{in}) and K_B is the* outer *of γ (written γ_{out}). If only condition 2 holds, K_B is the* inner *and K_A is the* outer *of γ.*

Notice that, as in the case of the controllability kernel, an edge of the SPDI may be split into two by a semi-separatrix — part inside, and part outside. As before, we can split the edge into parts, such that each part is completely inside, or completely outside the semi-separatrix.

The above notions are extended to SPDIs straightforwardly. The set of all the separatrices of an SPDI $\mathcal{S}$ is denoted by $\mathsf{Sep}(\mathcal{S})$, or simply Sep.

Now, let $\sigma = e_1 \dots e_n e_1$ be a simple cycle, $\angle^{\mathbf{b}_i}_{\mathbf{a}_i}$ $(1 \leq i \leq n)$ be the dynamics of the regions for which e_i is an entry edge and $I = [l, u]$ an interval on edge e_1. Remember that $\mathsf{Succ}_{e_1 e_2}(I) = F(I \cap S_1) \cap J_1$, where $F(x) = [a_1 x + b_1, a_2 x + b_2]$. Let $\mathbf{l}$ be the vector corresponding to the point on e_1 with local coordinates l and $\mathbf{l}'$ be the vector corresponding to the point on e_2 with local coordinates $F(l)$ (similarly, we define $\mathbf{u}$ and $\mathbf{u}'$ for $F(u)$). We define first $\overline{\mathsf{Succ}}^{\mathbf{b}_1}_{e_1}(I) = \{\mathbf{l} + \alpha(\mathbf{l}' - \mathbf{l}) \mid 0 < \alpha < 1\}$ and $\overline{\mathsf{Succ}}^{\mathbf{a}_1}_{e_1}(I) = \{\mathbf{u} + \alpha(\mathbf{u}' - \mathbf{u}) \mid 0 < \alpha < 1\}$. We extend these definitions in a straight way to any (cyclic) signature $\sigma = e_1 \dots e_n e_1$, denoting them by $\overline{\mathsf{Succ}}^{\mathbf{b}}_{\sigma}(I)$ and $\overline{\mathsf{Succ}}^{\mathbf{a}}_{\sigma}(I)$, respectively; we can compute them similarly as for $\overline{\mathsf{Pre}}$. Whenever applied to the fixpoint $I^* = [l^*, u^*]$, we denote $\overline{\mathsf{Succ}}^{\mathbf{b}}_{\sigma}(I^*)$ and $\overline{\mathsf{Succ}}^{\mathbf{a}}_{\sigma}(I^*)$ by ξ^l_σ and ξ^u_σ respectively. Intuitively, ξ^l_σ (ξ^u_σ) denotes the piece-wise affine closed curve defined by the leftmost (rightmost) fixpoint l^* (u^*).

We show now how to identify semi-separatrices for simple cycles.

Theorem 6 *Given an SPDI, let σ be a simple cycle, then the following hold:*

1. *If σ is EXIT-RIGHT then ξ^l_σ is a semi-separatrix curve (filtering trajectories from "left" to "right");*
2. *If σ is EXIT-LEFT then ξ^u_σ is a semi-separatrix curve (filtering trajectories from "right" to "left");*
3. *If σ is STAY, then the two polygons defining the invariance kernel ($\mathsf{Inv}^l(K_\sigma)$ and $\mathsf{Inv}^u(K_\sigma)$), are semi-separatrices.* □

In the case of STAY cycles, ξ^l_σ and ξ^u_σ are both also semi-separatrices. Notice that in the above result, computing a semi-separatrix depends only on one simple cycle, and the corresponding algorithm is then reduced to find simple cycles in the SPDI and checking whether it is STAY, EXIT-RIGHT or EXIT-LEFT. DIE cycles induce an infinite number of semi-separatrices and are not treated in this setting.

Example 11 *Fig. 4 shows all the semi-separatrices of the SPDI given in Example 1, obtained as shown in Theorem 6. The small arrows traversing the semi-separatrices show the inner and outer of each semi-separatrix: a trajectory may traverse the semi-separatrix following the direction of the arrow, but not vice-versa.* ■

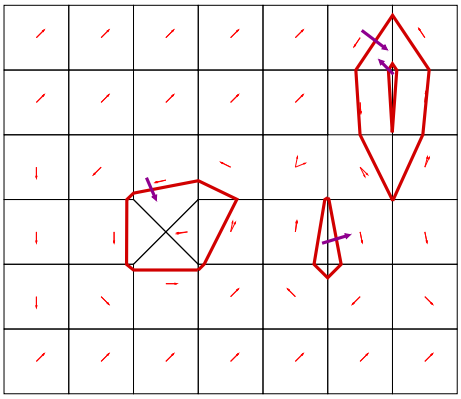

Figure 4: Semi-separatrices

The following two results relate feasible signatures and semi-separatrices.

Proposition 7 *If, for some semi-separatrix γ, $e \in \gamma_{in}$ and $e' \in \gamma_{out}$, then the signature ee' is not feasible.* □

Proposition 8 *If, for some semi-separatrix γ, and signature σ (of at least length 2), then, if $head(\sigma) \in \gamma_{in}$ and $last(\sigma) \in \gamma_{out}$, σ is not feasible.* □

3. STATE-SPACE REDUCTION

3.1 Reduction using Semi-Separatrices

Semi-separatrices partition the state space into two parts[3] – once one crosses such a border, all states outside the region can be ignored. We present a technique, which, given an SPDI and a reachability question, enables us to discard portions of the state space based on this information. The approach is based on identifying *inert* states (edges in the SPDI) not playing a role in the reachability analysis.

Definition 3 *Given an SPDI $\mathcal{S}$, a semi-separatrix $\gamma \in \mathsf{Sep}$, a source edge $e0$ and a destination edge $e1$, an edge e is said to be* inert *if it lies outside the semi-separatrix while $e0$ lies inside, or it lies inside, while $e1$ lies outside:*

$$inert^{\gamma}_{e0 \to e1} = \{e : \mathcal{E} \mid e0 \in \gamma_{in} \wedge e \in \gamma_{out}\} \cup \{e : \mathcal{E} \mid e1 \in \gamma_{out} \wedge e \in \gamma_{in}\}.$$

We can prove that these inert edges can never appear in a feasible signature:

Lemma 2 *Given an SPDI $\mathcal{S}$, a semi-separatrix γ, a source edge $e0$ and a destination edge $e1$, and a feasible signature $e0\sigma e1$ in $\mathcal{S}$. No inert edge from $inert^{\gamma}_{e0 \to e1}$ may appear in $e0\sigma e1$.* □

[3]Here, we do not consider the semi-separatrix itself.

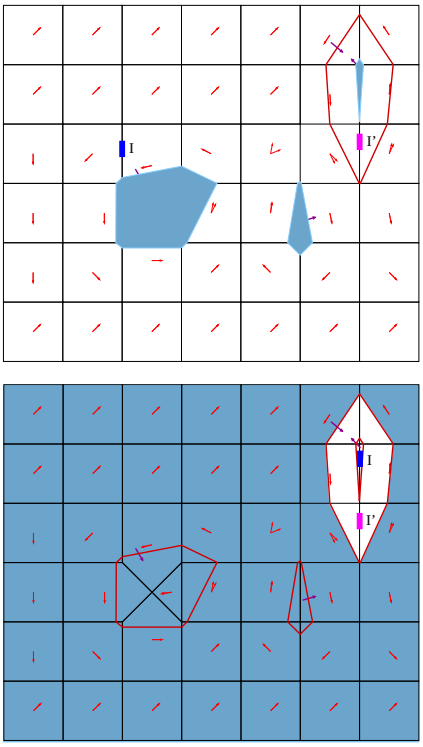

Figure 5: Reduction using semi-separatrices

Given an SPDI, we can reduce the state space by discarding inert edges.

Definition 4 *Given an SPDI $\mathcal{S}$, a semi-separatrix γ, and two edges, a source edge $e0$ and a destination edge $e1$, we define the reduced SPDI $\mathcal{S}^{\gamma}_{e0\to e1}$ to be the same as $\mathcal{S}$ but without the inert edges.*

Clearly, the resulting SPDI is not bigger than the original one. Finally, we prove that checking reachability on the reduced SPDI is equivalent to checking reachability on the original SPDI:

Theorem 7 *Given an SPDI $\mathcal{S}$, a semi-separatrix γ, and edges $e0$ and $e1$, then, $e1$ is reachable from $e0$ in $\mathcal{S}$ if and only if $e1$ is reachable from $e0$ in $\mathcal{S}^{\gamma}_{e0\to e1}$.* □

We have shown, that once semi-separatrices are identified, given a reachability question, we can reduce the size of the SPDI to be verified by removing inert edges of all the known semi-separatrices.

Example 12 *The shaded areas of Fig. 5 (a) and (b) are examples of subsets of the SPDI edges of the reachability graph, eliminated by the reduction presented in this section applied to all semi-separatrices, when answering reachability questions (in this case to the question: Is I' reachable from I?).* ■

This result enables us to verify SPDIs much more efficiently. It is important to note that model-checking an SPDI requires identification of simple loops, which means that the calculation of the semi-separatrices is not more expensive than the initial pass of the model-checking algorithm. Furthermore, we can perform this analysis only once for an SPDI and store the information to be used in any reachability analysis on that SPDI. Reduction, however, can only be applied once we know the source and destination states.

3.2 State-space Reduction using Kernels

We have already shown that any invariant set is essentially a pair of semi-separatrices, and since the invariance kernel is an invariant set, we can use the results from section 2.6 to abstract an SPDI using invariance kernels. We now turn our attention to state space reduction using controllability kernels:

Definition 5 *Given an SPDI $\mathcal{S}$, a loop σ, a source edge $e0$ and a destination edge $e1$, an edge e is said to be* redundant *if it lies on the opposite side of a controllability kernel as both $e0$ and $e1$:*

$$\begin{aligned}&\mathit{redundant}^{\sigma}_{e0\to e1}\\ &\{e:\mathcal{E} \mid \{e0,\ e1\} \subseteq \mathsf{Cntr}_{in}(\sigma) \cup \mathsf{Cntr}(\sigma) \wedge e \in \mathsf{Cntr}_{out}(\sigma)\} \cup\\ &\{e:\mathcal{E} \mid \{e0,\ e1\} \subseteq \mathsf{Cntr}_{out}(\sigma) \cup \mathsf{Cntr}(\sigma) \wedge e \in \mathsf{Cntr}_{in}(\sigma)\}\end{aligned}$$

We can prove that we can do without these edges to check feasibility:

Lemma 3 *Given an SPDI $\mathcal{S}$, a loop σ, a source edge $e0$, a destination edge $e1$, and a feasible signature $e0\sigma e1$ then there exists a feasible signature $e0\sigma' e1$ such that σ' contains no redundant edge from $\mathit{redundant}^{\sigma}_{e0\to e1}$.* □

Given an SPDI, we can reduce the state space by discarding redundant edges.

Definition 6 *Given an SPDI $\mathcal{S}$, a loop σ, a source edge $e0$ and a destination edge $e1$, we define the reduced SPDI $\mathcal{S}^{\sigma}_{e0\to e1}$ to be the same as $\mathcal{S}$ but without redundant edges.*

Clearly, the resulting SPDI is smaller than the original one. Finally, based on proposition 3, we prove that reachability on the reduced SPDI is equivalent to reachability on the original one:

Theorem 8 *Given an SPDI $\mathcal{S}$, a loop σ, a source edge $e0$ and a destination edge $e1$, then, $e1$ is reachable from $e0$ in $\mathcal{S}$ if and only if $e1$ is reachable from $e0$ in $\mathcal{S}^{\sigma}_{e0\to e1}$.* □

Given a loop which has a controllability kernel, we can thus reduce the state space to explore. In practice, we apply this state space reduction for each controllability kernel in the SPDI. Once a loop in the SPDI is identified, it is straightforward to apply the reduction algorithm.

3.3 Immediate Answers

By definition of the controllability kernel, any two points inside it are mutually reachable. This can be used to answer reachability questions in which both the source and destination edge lie (possibly partially) within the same controllability kernel. Using proposition 2, we know that any point in the viability kernel of a loop can eventually reach the controllability kernel of the same loop, which allows us to relax the condition about the source edge to just check whether it (partially) lies within the viability kernel. Finally, we note that the union of non-disjoint controllability sets is itself a controllability set which allows us to extend the result to

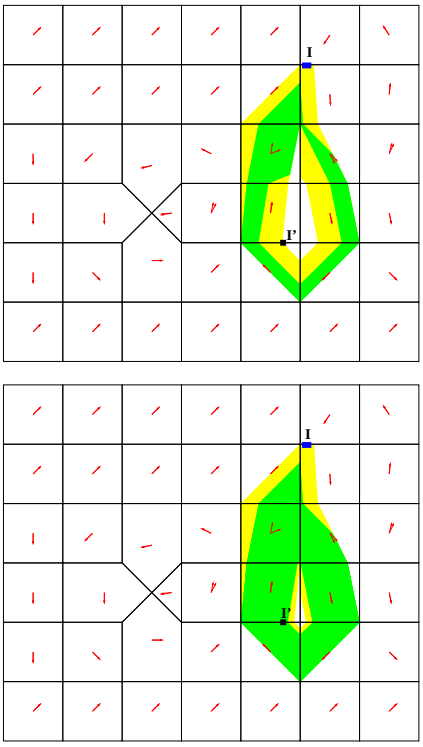

Figure 6: Answering reachability using kernels

work for a collection of loops whose controllability kernels form a strongly connected set.

Definition 7 *We extend viability and controllability kernels for a set of loops Σ by taking the union of the kernels of the individual loops, with $\mathsf{Viab}(K_\Sigma)$ being the union of all viability kernels of loops in Σ, and similarly $\mathsf{Cntr}(K_\Sigma)$.*

Definition 8 *Two loops σ and σ' are said to be compatible ($\sigma \leftrightsquigarrow \sigma'$) if their controllability kernels overlap: $\mathsf{Cntr}(K_\sigma) \cap \mathsf{Cntr}(K_{\sigma'}) \neq \emptyset$.*

We extend the notion of compatibility to a set of loops Σ to mean that all loops in the set are transitively compatible: $\forall \sigma,\ \sigma' \in \Sigma \cdot \sigma \leftrightsquigarrow^* \sigma'$.

Based on proposition 2, we can prove the following:

Theorem 9 *Given a source edge e_{src} and a destination edge e_{dst}, if for some compatible set of loops Σ, we know that $e_{src} \cap \mathsf{Viab}(K_\Sigma) \neq \emptyset$ and $e_{dst} \cap \mathsf{Cntr}(K_\Sigma) \neq \emptyset$, then e_{dst} is reachable from e_{src}.* □

Example 13 *Fig. 6-(a) shows a viability and a controllability kernel of a cycle and two intervals I and I'. Whether I' is reachable from I cannot be answered immediately in this case, but Fig. 6-(b) shows the overlapping of the viability and controllability kernels depicted in Fig. 6-(a) with the kernels of an inner cycle. I' thus lies in a compatible controllability kernel, and we can immediately conclude (by theorem 9) that I' is reachable from I.* ■

In practice, we propose to use these theorems to enable answering certain reachability questions without having to explore the complete state space. It can also be used to reduce reachability questions to (possibly) simpler ones by trying to reach a viability kernel rather than a particular edge. As in the case of semi-separatrices, a preliminary analysis of an SPDI's kernels be used in all subsequent reachability queries. SPeeDI [14] starts by calculating and caching all loops in the given SPDI, and can thus easily identify maximal compatible sets of loops. Combining this technique with the semi-separatrix reduction technique we envisage substantial gains.

4. COMPOSITIONAL ANALYSIS

4.1 SPDI Decomposition

In this section, we propose a number of theorems which, given an SPDI and a reachability question, for each controllability kernel, allow us to either (i) answer the reachability question without any further analysis; or (ii) reduce the state space necessary for reachability analysis; or (iii) decompose the reachability question into two smaller, and independent reachability questions.

The following theorem enables us to answer certain reachability questions without any analysis, other than the identification of controllability and viability kernels. This result is based on two properties, that within the controllability kernel of a loop, any two points are mutually reachable, and that any point on the viability kernel of the same loop can eventually reach the controllability kernel. Therefore if the source edgelist lies (possibly partially) within the viability kernel of a loop, and the destination edgelist lies (possibly partially) within the controllability kernel of the same loop, then, there must exist a trajectory from the source to the destination edgelist. The full proof of this result can be found in [18].

Theorem 10 *Given an SPDI $\mathcal{S}$, two edgelists I and I' and a controllability kernel C, then if $I \subseteq C^+$ and $I' \subseteq C$, then $I \xrightarrow{\mathcal{S}} I'$.* □

The following theorem allows us to reduce the state space based on controllability kernels. If both the source and destination edgelists lie on the same side of a controllability kernel, then we can disregard all edges on the other side of the kernel. The full proof of this result can be found in [18].

Theorem 11 *Given an SPDI $\mathcal{S}$, two edgelists I and I' and a controllability kernel C, then if $I \subseteq C_{in}$ and $I' \subseteq C_{in}$, then $I \xrightarrow{\mathcal{S}} I'$ if and only if $I \xrightarrow{\mathcal{S} \setminus C_{out}} I'$. Similarly, if $I \subseteq C_{out}$ and $I' \subseteq C_{out}$, then $I \xrightarrow{\mathcal{S}} I'$ if and only if $I \xrightarrow{\mathcal{S} \setminus C_{in}} I'$.* □

Finally, the following new result allows us to decompose a reachability question into two smaller questions independent of each other. The theorem states that if the source and destination edgelists lie on opposite sides of a controllability kernel, then we can try (i) to reach the related viability kernel from the source edgelist, and (ii) to reach the destination from the controllability kernel. The original reachability question can be answered affirmatively if and only if both these questions are answered affirmatively.

Theorem 12 *Given an SPDI $\mathcal{S}$, two edgelists I and I' and a controllability kernel C, then if $I \subseteq C_{in}$ and $I' \subseteq C_{out}$,*

then $I \xrightarrow{\mathcal{S}} I'$ *if and only if* $I \xrightarrow{\mathcal{S}\setminus C_{out}} C^+ \wedge C \xrightarrow{\mathcal{S}\setminus C_{in}} I'$. *Similarly, if* $I \subseteq C_{out}$ *and* $I' \subseteq C_{in}$, *then* $I \xrightarrow{\mathcal{S}} I'$ *if and only if* $I \xrightarrow{\mathcal{S}\setminus C_{in}} C^+ \wedge C \xrightarrow{\mathcal{S}\setminus C_{out}} I'$.

4.2 Unavoidable Kernels

Unavoidable kernels are defined geometrically to be kernels which a straight line from the source interval to the destination interval 'intersects' an odd number of times. We call the kernel unavoidable since it can be proved that any path from the source to the destination will have to pass through the kernel.

Definition 9 *Given an SPDI* $\mathcal{S}$ *and two edgelists* I *and* I', *we say that a controllability kernel* $\mathsf{Cntr}(K_\sigma)$ *is* unavoidable *if any segment of line with extremes on points lying on* I *and* I' *intersects with both the edges of* $\mathsf{Cntr}^l(K_\sigma)$ *and those of* $\mathsf{Cntr}^u(K_\sigma)$ *an odd number of times (disregarding tangential intersections with vertices).*

The following theorem enables us to discover separating controllability kernels using a simple geometric test.

Theorem 13 *Given an SPDI* $\mathcal{S}$, *two edgelists* I *and* I', *and a controllability kernel* $C = \mathsf{Cntr}(K_\sigma)$, *then* C *is an unavoidable kernel if and only if one of the following conditions holds (i)* $I \subseteq C_{in}$ *and* $I' \subseteq C_{out}$; *or (ii)* $I \subseteq C_{out}$ *and* $I' \subseteq C_{in}$.

Corollary 2 *Given an SPDI* $\mathcal{S}$, *two edgelists* I *and* I', *and an unavoidable controllability kernel* $C = \mathsf{Cntr}(K_\sigma)$, *then* $I \xrightarrow{\mathcal{S}} I'$ *if and only if* $I \xrightarrow{\mathcal{S}} C$ *and* $C \xrightarrow{\mathcal{S}} I'$.

The following result relates unavoidable kernels:

Proposition 9 *Given two disjoint controllability kernels* C *and* C', *both unavoidable from* I *to* I', *then either* C' *is unavoidable from* I *to* C *or* C' *is unavoidable from* C *to* I', *but not both.*

4.3 Parallel Reachability Algorithm

In Fig. 8 we give an algorithm for parallel reachability analysis of SPDIs using parallel recursive calls corresponding to independent reachability questions.

The function `ReachParKernels` is called with the SPDI to consider, a list of kernels still to be used for reduction, and the source and destination edgelists. With no kernels to consider, the algorithm simply calls the standard sequential algorithm (`Reach`). Otherwise, one of the kernels is analyzed, with three possible cases:

1. If the source lies (possibly partially) on the extended kernel, and the destination lies (possibly partially) on the kernel, then we can give an immediate answer (using theorem 10).

2. If both the edgelists lie on the same side of the kernel, then we simply eliminate redundant parts of the SPDI — anything on the other side of the kernel (theorem 11).

3. Otherwise, if the kernels both lie on opposite sides of the kernel, we can split the problem into two independent questions (reaching the kernel from the source, and the destination from the kernel) which can be run in parallel (theorem 12). An affirmative answer from both these subquestions is equivalent to an affirmative answer to the original question.

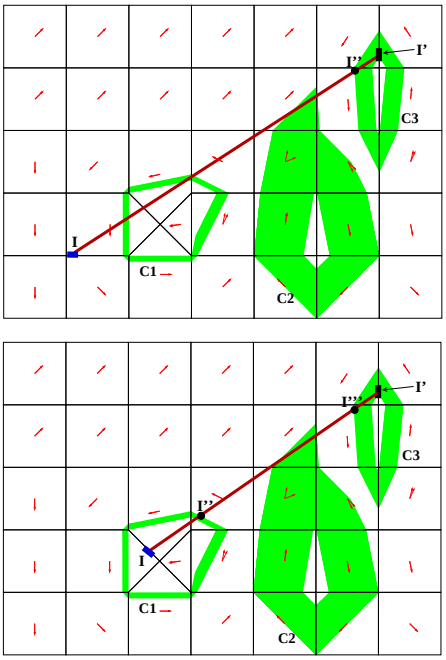

Figure 7: Unavoidable kernels and independent reachability questions

Note that the function `ReachParKernels` is compositional in the sense that each recursive call launch a process which operates in (most cases in) disjoint state spaces which are smaller than the original one ($\mathcal{S}$). The final answer is the composition of the partial reachability questions.

Given two edgelists I and I', we define the following predicate $I \xrightarrow{\mathcal{S}}_{\|} I' \equiv \mathtt{ReachPar}(\mathcal{S}, I, I')$. The following theorem states that the (compositional) parallel algorithm exactly answers the reachability question, also giving a soundness and completeness proof of the algorithm:

Theorem 14 *Given an SPDI* $\mathcal{S}$ *and two intervals* $I \subseteq e$ *and* $I' \subseteq e'$, $I \xrightarrow{\mathcal{S}} I'$ *if and only if* $I \xrightarrow{\mathcal{S}}_{\|} I'$.

5. CONCLUDING REMARKS

We have given an overview of our recent results on the optimisation of SPDI reachability analysis. Using semi-separatrices and kernels, we presented techniques to improve reachability analysis on SPDIs. In all cases, the techniques require the identification and analysis of loops in the SPDI. We note that most of this information is still required in reachability analysis, and thus no extra work is required to perform the optimization presented in this paper. We have also shown how answering reachability on an SPDI can be reduced to a number of smaller reachability questions. These two techniques can be combined together applying the reduction tecniques globally (on the whole SPDI) or locally on the decomposed SPDIs).

```
function ReachPar(S, I, I') =
  ReachParKernels (S, ControllabilityKernels(S), I, I')

function ReachParKernels(S, [], I, I') =
  Reach(S, I, I');

function ReachParKernels(S, k:ks, I, I') =
  if (ImmedieteAnswer(S, I, I')) then
     True;
  elsif (SameSideOfKernel(S, k, I, I')) then
     S_I := S \ EdgesOnOtherSideOf(S, k, I');
     ReachParKernels(S_I, ks, I, I');
  else
     S_I := S \ EdgesOnOtherSideOf(S, k, I);
     S_I' := S \ EdgesOnOtherSideOf(S k, I');
     parbegin
        r1 := ReachParKernels(S_I, ks, I, viability(k));
        r2 := ReachParKernels(S_I', ks, k, I');
     parend;
     return (r1 and r2);
```

Figure 8: Parallel algorithm for reachability of SPDIs.

Our work is obviously restricted to planar systems, which enables us to compute these kernels exactly. In higher dimensions and hybrid systems with higher complexity, calculation of kernels is not computable. Other work is thus based on calculations of approximations of these kernels (e.g., [8, 7, 19]). We are not aware of any work using kernels and semi-separatrices to reduce the state-space of the reachability graph as presented in this paper.

We are currently exploring the implementation of the optimizations presented in this paper to improve the efficiency of SPeeDI$^+$ [14]. We are also investigating other applications of these kernels in the model-checking of SPDIs.

One current research direction is to apply our results to semi-decide the reachability question for SPDIs defined on 2-dimensional manifolds, for which the decidability of reachability remains unresolved [3]. Maybe the most prominent application of SPDIs is for approximating complex non-linear differential equations on the plane, for which an exact solution is not known. The decidability of SPDIs reachability and of its phase portrait construction would be of invaluable help for the qualitative analysis of such equations. The challenge would be to find an "intelligent" partition of the plane in order to get an optimal approximation of the equations. Since such partition might produce a high number of regions, our parallel algorithm might be extremely useful here.

6. REFERENCES

[1] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[2] E. Asarin, G. Pace, G. Schneider, and S. Yovine. SPeeDI: a verification tool for polygonal hybrid systems. In *CAV'2002*, volume 2404 of *LNCS*, 2002.

[3] E. Asarin and G. Schneider. Widening the boundary between decidable and undecidable hybrid systems. In *CONCUR'2002*, volume 2421 of *LNCS*, 2002.

[4] E. Asarin, G. Schneider, and S. Yovine. On the decidability of the reachability problem for planar differential inclusions. In *HSCC'2001*, number 2034 in LNCS, pages 89–104, 2001.

[5] E. Asarin, G. Schneider, and S. Yovine. Towards computing phase portraits of polygonal differential inclusions. In *HSCC'02*, volume LNCS 2289, 2002.

[6] J.-P. Aubin. The substratum of impulse and hybrid control systems. In *HSCC'01*, volume 2034 of *LNCS*, pages 105–118. Springer, 2001.

[7] J.-P. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube. Towards a viability theory for hybrid systems. In *European Control Conference*, 2001.

[8] J.-P. Aubin, J. Lygeros, M. Quincampoix, S. Sastry, and N. Seube. Viability and invariance kernels of impulse differential inclusions. In *Conference on Decision and Control*, volume 40 of *IEEE*, pages 340–345, December 2001.

[9] A. Deshpande and P. Varaiya. Viable control of hybrid systems. In *Hybrid Systems II*, number 999 in LNCS, pages 128–147, 1995.

[10] T. Henzinger, P. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *STOC'95*, pages 373–382. ACM Press, 1995.

[11] G. Lafferriere, G. Pappas, and S. Yovine. Symbolic reachability computation of families of linear vector fields. *Journal of Symbolic Computation*, 32(3):231–253, Sept. 2001.

[12] O. Maler and A. Pnueli. Reachability analysis of planar multi-linear systems. In *CAV'93*, pages 194–209. LNCS 697, Springer Verlag, July 1993.

[13] A. Matveev and A. Savkin. *Qualitative theory of hybrid dynamical systems*. Birkhäuser Boston, 2000.

[14] G. Pace and G. Schneider. SPeeDI$^+$. `http:\\www.cs.um.edu.mt\speedi`.

[15] G. Pace and G. Schneider. Model checking polygonal differential inclusions using invariance kernels. In *VMCAI'04*, number 2937 in LNCS, pages 110–121. Springer Verlag, December 2003.

[16] G. Pace and G. Schneider. Compositional algorithm for parallel model checking of polygonal hybrid systems. In *3rd International Colloquium on Theoretical Aspects of Computing (ICTAC'06)*, volume 4281 of *LNCS*. Springer-Verlag, 2006.

[17] G. Pace and G. Schneider. Static analysis for state-space reduction of polygonal hybrid systems. In *Formal Modelling and Analysis of Timed Systems (FORMATS'06)*, volume 4202 of *LNCS*. Springer-Verlag, 2006.

[18] G. Pace and G. Schneider. Static analysis of SPDIs for state-space reduction. Technical Report 336, Department of Informatics, University of Oslo, PO Box 1080 Blindern, N-0316 Oslo, Norway, April 2006.

[19] P. Saint-Pierre. Hybrid kernels and capture basins for impulse constrained systems. In *HSCC'02*, volume 2289 of *LNCS*. Springer-Verlag, 2002.

[20] G. Schneider. *Algorithmic Analysis of Polygonal Hybrid Systems*. PhD thesis, VERIMAG – UJF, Grenoble, France, July 2002.

[21] G. Schneider. Computing invariance kernels of polygonal hybrid systems. *Nordic Journal of Computing*, 11(2):194–210, 2004.

[22] S. Simić, K. Johansson, S. Sastry, and J. Lygeros. Towards a geometric theory of hybrid systems. In *HSCC'00*, number 1790 in LNCS, 2000.

MLRS, a Resource Server for the Maltese Language

Mike Rosner
Department CSAI
University of Malta
Malta
mike.rosner@um.edu.m

Ray Fabri
Institute of Linguistics
University of Malta
Malta
ray.fabri@um.edu.mt

Duncan Attard
Department of CSAI
University of Malta
Malta
mike.rosner@um.edu.m

Albert Gatt
Dept Computing Science,
King's College
Aberdeen, UK
agatt@csd.abdn.ac.uk

ABSTRACT

This paper decribes the aims, objectives, and current achievements of MLRS (Maltese Language Resource Server), a two-year RTDI project that has been running for just under one year. The overall aim of the project, as suggested by the title, is the provision of such language resources, on a par with those that are available for other languages, together with a framework for their collection, development and maintenance. The paper focuses primarily on the design and current implementation of the corpus server and a lexicon development toolkit. We will conclude with plans for the coming year.

Keywords

Natural Language Processing, Machine Translation, Parallel and Cluster Computing, Grid Computing, Networking, Collaborative Systems.

ACKNOWLEDGEMENT

The work described has been supported under the RTDI initiative administered through the Malta Council for Science and Technology. We are indebted to MITTS, a partner on the project, for the supply of corpus materials from Government sources.

1. INTRODUCTION

The term "language resources" refers to a set of speech or language data and descriptions in machine readable form. They are regarded as key enablers in three main areas for developments in (i) natural language processing systems and tools, (ii) linguistic research that yields new knowledge about the language itself, and (iii) language-related industries such as software localization, translation, publishing etc.

This paper decribes the aims, objectives, and current achievements of MLRS (Maltese Language Resource Server), a two-year RTDI project that has been running for just under one year. The overall aim of the project, as suggested by the name, is the provision of such language resources, on a par with those that are available for other languages, together with a framework for their collection, development and maintenance. It shares some of the objectives and approaches of an earlier project called Maltilex as described in Rosner et. al (1998), Dalli (2001).

In the project we are interested in both *primary* language resources, which can be more or less directly obtained at source, such as newspaper text or audio recordings, and *secondary* ones, which are derived using a combination of automated tools and expert knowledge. Examples include corpora marked up for parts-of-speech or other information such as named entities.

We include in particular amongst such secondary resources a *computational lexicon* - that is, a structured collection of information about words that can be used by a whole series of "language enabled" user applicatons including spell and style checkers. The compilation of such a resource demands extensive interaction with linguists, and we have catered for this in the design of the system.

Section 2 of the paper discusses the structure of the Maltese National Corpus. Section 3 describes the morpho-syntactic tagging scheme that has been developed for Maltese. Section 4 elaborates on ODL, a metalanguage for the description of lexical information whilst section 5 concentrates on implementation issues. Section 6 discusses future work. We will conclude with plans for the coming year.

2. CORPUS

2.1 Maltese National Corpus

We are in the process of building a Maltese National Corpus. The basic philosophy behind this resource is similar to the British National Corpus (BNC), its British counterpart (see Aston and Burnard [1]). There are several characteristics that we would like such a corpus to have when complete. These include

Representativeness. By representative we mean that the content – a collection of written and spoken language resources, should have characteristics and properties that reflect the language as a whole. We expect this collection to be diverse in terms of genre and subject matter (including, for example, newspaper text,

fiction, technical documents in different areas, educational materials, laws etc.) and of a certain size: the BNC comprises about 100 million words and there is no reason to believe that, given adequate resources, the MNC will end up being much smaller.

Accessibility. A second important desideratum is that the corpus should be easily accessible. It goes without saying that the primary access mechanism will be the Internet, and that where possible, web services will be used to deliver resources and tools. Different categories of user need to catered for, and we are in the process of designing such services for originators, annotators, and consumers of corpus materials. An indexing scheme is in the process of being devised.

Annotation Support. Corpus materials are likely to arrive from contributors in any form, and we wish to impose as few conditions as possible on what we deem to be an acceptable form. For example, text resources received to date have been in PDF, Word, PS, and ASCII. We do not even exclude contributions on paper. At the same time, we want the possibility to annotate individual documents or sets of documents with different kinds of information and furthermore, we want to provide support for such annotation. In some cases, the annotation itself will be manual. In others – part of speech tagging comes to mind - it will be automated. In order for annotation to be possible at all, documents must conform to certain basic standards, and for this purpose, we have adopted different levels of representation, as shown in Figure 1.The idea is that the higher the level of representation, the more refined the level of automated linguistic processing.

2.2 Corpus Contents

At the outset of the project, the corpus consisted of about 1M words of news text, collected on an opportunistic basis primarily from Maltese language newspapers. The texts represent a broad spectrum of the types of articles found in these media.

Submissions are currently managed by hand using an ftp server. This is a temporary solution pending the development of a lightweight web service that will allow upload of documents and browsing of the index up to the level of filenames

The current directory structure is highly rudimentary, being divided into the following main categories, some of which are subdivided and others which may become subdivided as the corpus grows.

- **Academic:** academic and scholarly papers.
- **Blog:** weblogs
- **Chat:** multi-party text based interaction over the internet.
- **Email:**
- **Government**. Goverrnment documents, legal notices and publications. Will probably subdivide into different government offices. EU documents will also find their place here.
- **Law**. Legalistic documents including laws and judgements.
- **Literature:** Literary works of all kinds. Dwill probably subdivide into poetry, prose, and drama.
- **Press**. Newspaper text from daily and weekly newspapers. The directory is currently subdivided by newspaper (Nazzjion; Il-Mument; Kull-Hadd; Lehen)
- **Religion**. Religious and biblical texts
- **Speech**: speech data (currently empty)
- **Miscellaneous**

Table 1 summarises the current contents, not counting some documents that have been submitted on CD ad on paper.

Item	Remarks	Date Added	Size (MB)
Press		2005	1
Government		2006	123
Law		2005	25
Religion		2006	16
TOTAL			165

Table 1. Current Level 0 Contents

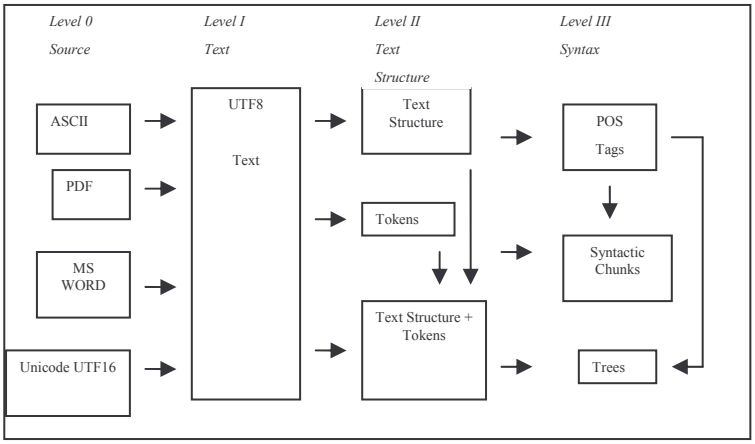

Figure 1: Levels of Representation

2.3 Corpus Representation Levels

Level 0. At the bottom of the heap, so to speak, we designate resources received in the ftp directory from the originator as being at Level 0, sometimes referred to as the source level. At level 0, very few assumptions about the nature of the resource are possible except basic information concerning author, title, date etc.

Level I. The text level is the first level at which we are in a position to define a standard.The idea here is that this represents a minimal input standard for any kind of subsequent automated processing. Whilst the mapping between level 0 and level I is undertaken on an ad-hoc basis (this includes manual conversion, use of OCR etc), we aim for completely automated mappings from level I onwards. The most important aspect is the text encoding – for us it is UTF8. Concretely, level 1 is encoded in UTF8 and includes a lightweight TEI-style (cf. TEI Consortium 2002) header identifying and certain key features of the source text including title, author, date.

Level II. This epresents a first level of actual annotation where text structure is made explicit. In contrast to level 1, we foresee several different annotation schemes operating at level II, since there is some debate about what constitutes basic text structure.

Minimally, we assume that the representation of certain characters with distinct UTF8 codes has been standardized (e.g " and " are mapped to "), tabs are mapped to spaces, consecutive whitespaces are mapped to single whitespace, and each distinguishable token appears on separate line, with blank lines indicating paragraph break. Other kinds of text structure might also be annotated at this level, e.g.

- tokenization with type information (i.e. with distinction between word, punctuation, number)
- sentence and paragraph boundaries
- higher level text structure. For example, in the LT4eL project (Monachesi et. al 2006), a "basic XML" format has been defined which includes a subset of the information present in an HTML document that is likely to be useful for the purposes of automatic content extraction: font styles, headings, lists, and so on.

There are many possible ways in which such annotations for text structure might be arrived at. One might compute them all together, or one might choose to identify the sentence and paragraph boundaries using one annotation scheme, tokenization using another scheme, finally combining the two together in a third scheme. Cristea and Butnariu (2004) describe how different annotation schemes can be regarded as points on a lattice so that paths between points automatically define pipeline computations for transforming one annotation scheme into another.

Level III Here linguistic annotation is introduced in order identify chunk structures (cf. Abney 1991) or trees.

Finally, higher levels of annotation, involving semantic information, such as named entities coreferencing etc, come in at level IV and beyong. These are not shown for the sake of clarity.

3. Morphosyntactic Annotation

This section describes a tagset that has been specially developed for Maltese. The tagset plays a crucial role in level III corpus annotation (the first level of annotation to include morpho-syntactic information). It also plays a crucial role in the lexicon, where the morpho-syntactic properties of words are actually stored. We first describe the tagset itself. This is followed by a discussion of the annotation process

3.1 Tagset

The tagset for Maltese is an extension of earlier work which drew directly on the EAGLES recommendations (Leech and Wilson, 1996). Since the original category set proposed by EAGLES proved to be somewhat limited to cover the full range of morphosyntactic phenomena in Maltese (cf. Gatt et al. 2003), this design was extended. Apart from the major grammatical categories and morphosyntactic features, tags for punctuation, as well as special particles specific to the Maltese language have also been included.

The tagset is structured such that every morphosyntactic tag is represented as a pair (C,F), where *C* is a major category, and *F* is a set of *features*, each of which is represented as an attribute-value pair. Table 2 lists the major categories, whilst Table 3 shows the full feature set.

Major categories
noun
verb
modifier
pronoun
determiner
adposition
negation
verb aspect marker
conjunction
existential
interjection
punctuation
residual

Table 2. Major categories in the Maltese tagset

Feature set	
Level 1 Features	Level 2 Features
type	dimension
person	degree
number	aspect
gender	tense
attachment	mood
negation	case

Table 3. Major categories and features in the Maltese tagset

As shown in table 3, features are divided into two levels. This division was primarily aimed at splitting the task of annotation into two stages: Level 1 features were annotated during a first pass; Level 2 features, which tend to be more limited in their distribution, were annotated during the second pass.

Within any major category, the most important feature is *type*, whose values distinguish between different subclasses of the

category. Thus, elements of category *noun* can be of type *common* or *proper*, verbs fall into *main* and *auxiliary* types, and so on.

To illustrate, table 4 displays the main types of verbs and nouns (i.e. the values of the *type* attribute for these categories).

Major Category	Types
noun	1. common 2. proper
verb	2. main 3. auxiliary 4. copular 5. participle 6. modal

Table 4. Values of type for the main categories of *noun* and *verb*

The *type* for any category is meant to make distinctions that exert an influence on the morphosyntactic structure of a phrase, rather than semantic distinctions. For example, the personal pronouns in Maltese have a copular function, exemplified in by *huwa* in (1); in such cases, the pronoun is marked as copular verbs.

(1) dak huwa ħija
dem-3SgM he-3SgM (Pro-3SgM) brother-1Sg-Gen
'that is my brother'

Clearly, not all features are defined for all categories. Thus, *aspect* pertains only to elements of the verbal category, while *case* is only annotated in two cases: that of nouns explicitly marked for the genitive (which is limited to a subset of nouns in the Construct State, exemplified in (2)), and the adposition *lil*, which functions as an non-nominative marker, as shown in (3).

(2) Mart Toni
wife-gen Tony
'Tony's wife'

(3) Pietru ċempel lil Ġanni.
Peter call-P3Sg prep-*non-nom* John
'Peter called John'

One of the advantages of this layout is that it is possible to define which features pertain to a particular category type. This is important, in that the application of features such as *tense* and *aspect* is not always straightforward (cf. Fabri, 1996 for discussion). Thus, it has been argued that main verbs in Maltese are only inflected for aspect, rather than tense. The exception is the auxiliary *kien*, which functions as a tense marker, distinguishing between *past* and *non-past*. A further category, that of verb aspect markers, deserves special mention. This category was introduced to accomodate particles whose function is to further mark the event structure (*aksionsart*) of verb phrases, over and above the information supplied by the inflectional morphology of the verb itself. The category could not be incorporated with verbs, because none of the features that apply to the different verb types applies to them (e.g. they are not inflected for gender or number). Moreover, the *aspect* attribute does not apply to these elements, because it is incorporated in the *type* attribute. Examples of the two types of aspect markers, *progressive* and *prospective* are shown in (4) and (5).

(4) kien qed jiekol
be-past prog eat-prog
'he was eating'

(5) kien se jiekol
be-past fut eat-prog
'he was about to eat'

Another category, that of *modifiers* incorporates both adverbs and adjectives. These are distinguished by the *type* attribute. The reason for this is that there is a large class of modifiers with both an adjectival and an adverbial function in the Maltese language. Distinguishing these using different categories would fail to capture a potentially useful generalisation for future users of the corpus. Of the other features, one of the most important is *attachment*. This was inherited from the original EAGLES recommendations, and distinguishes between morphosyntactically independent elements, and clitics.

3.2 Level III Annotation

Morpho-syntactic annotation of the corpus is highly labour intensive, and yet is a prerequisite of almost any semantic analysis of corpus materials. Clearly, an automatic solution needs to be devised using an appropriate part of speech tagger. Many tagging algorihms are available. We are currently using the Rule-Based Tagger developed by Brill (1996), to automatically annotate increasingly large samples of the corpus.

The work is being carried out in three stages. During the first stage, a random sample of ca. 3000 words was taken from the newspaper corpus. This was manually annotated with Level 1 features during a first pass. Following validation of the output of the first pass, Level 2 features were added. During this initial phase, the tagset was expanded to accomodate distinctions that were deemed useful because of their frequency. For instance, the category of *existentials* was introduced at this stage (following the practise adopted for Version 2 of the CLAWS tagset for the British National Corpus), to mark existential uses of demonstrative pronouns, as shown in (6).

(6) hawn wieħed raġel
dem-3Sg/Ex one-SgM man
'here is a man'

Following this initial phase, a new sample, of a further 3000 words was collected, and annotated manually.

Our decision to use the Brill tagger was motivated by the fact that it encodes generalisations made from training data as rules which can be manually edited for corrections. The automatic annotation is proceeding in train-test-retrain cycles. Small samples are annotated using the tagger and manually corrected. Subsequently, the newly annotated samples are added to the training data and the tagger is retrained.

Future work on annotation is now proceeding in two directions. The first, and most urgent, is the completion of the automatic annotation for a large text sample. The main sticking point here is the provision of training data for the tagger.

Secondly, we are investigating the relationship between the tagset and the contents and organisation of entries in the lexicon. For this to be possible we need to have very clear ideas about :

- The nature and representation of tag information;
- the relationship between information carried by tags and information carried by lexical entries;
- the nature and representation of lexical information;
- the sharing of information between lexical entries.

For the purpose of answering these questions as well as developing a practical system we are developing a special-purpose description language called ODL (Object Description Language) for lexical information. The language provides a way for the linguist to specify what constitutes a possible lexical entry taking account of internal dependencies and constraints between data fields,

A key component of this investigation concerns developing a simple description language to handle inheritance of features and values within the lexicon. This is based on the kind of inheritance found in object oriented programming languages, hence the name: ODL (Object Description Language). This is discussed in the next section.

4. OBJECT DESCRIPTION LANGUAGE (ODL)

4.1 Lexical Information

ODL is a very simple object-oriented description language for lexical information. It is based on a notion of class which is very similar to that used in object oriented programming languages such as Java or C#.

The main assumption underlying the language is that lexical information, i.e. the information associated with lexical entries, is expressed in the form of classes, and that a class is a collection of attribute/value pairs. This approach has a long tradition, underlying the constraint-based unification grammar approach to language processing typified by, e.g. PATR2 (Shieber 1984), FUG (Kay 1985) and many others (GPSG, HPSG, TFS etc).

An important and useful aspect of being able to create classes (e.g. part of speech classes) is that of providing each with a general default set of attributes and values which would first of all be applied automatically to new words entering the lexicon, and secondly would list only those values which are applicable to certain attributes.

The language thus includes three classes of symbol which, for the purpose of this article, are distinguished notationally as follows:

- Classes (written in uppercase)
- Attributes (written with initial uppercase letter)
- Values (lowercase or integer)

Examples of classes might be `NOUN`, `PROPER_NOUN`, `COMMON_NOUN`, `VERB`, `TRANSITIVE_VERB`, etc. Typical attributes for these classes might be `Category`, `Number`, `Gender`, `Case`, and typical values of the respective attributes might be `noun`, `sing`, `masc`, `gen`. With these symbols we can define arbitrary sets of attributes and values.

A Maltese word like *kelb* (dog), for example, might have a lexical entry that comprises the following set of attributes and values:

```
{(Category,noun),
 (Number,sing),(Gender,masc) }
```

Clearly, only some collections of attributes and values make sense, linguistically speaking. For example, the following is linguistic nonsense

```
{(Category,verb),
 (Case,gen),(Dimension,masc) }
```

The idea behind an ODL description is (a) to precisely characterise the set of well formed classes and (b) to serve as a formal basis for the efficient compilation of the entire system of well-formed classes.

Currently we are experimenting with the possibility of flattening out classes into a bit string which is stored as a single record in a conventional database table. The result is that the database data structure, based on the ODL code, is compiled and stored prior to the building of the actual lexicon.

An ODL description contains the following parts in order:

1. Enumeration Declarations
2. Class Declarations
3. Rules (Optional)
4. Replacement Declarations (Optional)
5. Macro Definitions (Optional)

The next sections discuss each of these in turn

4.2 Enumeration Declarations

These enumerate the set of allowed values for a given attribute:

```
enum Cat      {noun, verb, adj, pronoun}
enum Type     {common, proper, main, aux}
enum Number   {sing, dual, plur. *}
enum Case     {gen, non-nom, *}
```

The value * is a special wildcard symbol which allows an

attribute of a word in the lexicon to remain "unspecified". During subsequent processing, this unspecified value is consistent with the any of the normal values in the enumeration.

The absence of * in an enumeration implies that a normal value *must* be present for a well–formed class involving that attribute.

4.3 Class Declarations

These define the attributes and values that make up a class, e.g.

```
class NOUN
{
  Cat           = noun;
  Type          = common | proper;
  Number        = *;
}
```

The compiler checks that the class declaration is consistent with the enumerations already specified.

The example shown is valid because {noun}⊆Cat, {common,proper}⊆Type, and Number is unspecified.

4.4 Class Inheritance

As in object oriented programming languages, the class system supports inheritance. For instance:

```
class PRONOUN: NOUN
{
 Case = *;
}
```

states that the `PRONOUN` class extends the `NOUN` class by adding a `Case` attribute to the set of attributes already belonging to NOUN.

There is of course potential for conflict in such a system. For instance, noticing that, as a consequence of inheritance, PRONOUN will have Cat = noun, we might wish to write:

```
class PRONOUN: NOUN
{
 Cat  = pronoun
 Case = *;
}
```

for which the value of the `Cat` attribute conflicts with that inherited from `NOUN`. The solution to all such conflicts is to establish precedence rules which allow information to be overwritten in a systematic way. In this case we require that he more specific inheriting class (`PRONOUN`) to overwrite information supplied by the more general inherited class (`NOUN`).

Sometimes we will want to define a class which inherits attributes and values from more than one subclass. In this case, we write

```
Class CLASS: CLASS1, CLASS2,…,CLASSN
```

Once again, the conflict can be resolved provided that we can establish unambiguous precedence relations between the classes.

This problem is not a new one, having been discussed extensively within AI/Knowledge Representation communities some 20 years ago. In the special case of language, several articles on inheritance and the lexicon appeared in a special issue of Computational Linguistics (Gazdar & Daelemans 1992).

We are still experimenting with an appropriate solution but in the first instance, propose to follow the example of Russell et. al. (1992) by using an algorithm based upon that used in Python 2.3. This is currently described on the Python website at www.python.org.

4.5 Rules

A well-known fact about Maltese is that plural entities are without gender. This fact implies that any well-formed class having a `Number` attribute equal to `plur` should not have a `Gender` attribute. ODL includes rules of the following kind to state such information:

```
if (Number == Plural) { !Gender }
```

This kind of information is used to maintain the well-formedness of defined classes in two ways:

- to prevent invalid information from entering the system in the first place (e.g the lexicon editor should disallow any attempts to provide a value for gender if the number is equal to plural) and
- to ensure that any combinatory operation that is employed by the system (e.g. class inheritance) never produces a class that conflicts with the rules.

Since these are the only ways classes can come about, we can thus guarantee that information represented by classes is completely consistent with the linguist's specification.

Currently, these rules are employed at user interface level. If a class satisfies the rule antecedent, the consequent is automatically asserted.

Where a consequent has the form !Attribute, as in the example above, we assign the distinguished symbol "~" as value of that attribute. This behaves just like any other constant, in that an attribute having it as value will only combine with another attribute having the same value. In other words, there is no possibiity of combining a non-existent attribute with one that exists.

Finally, unlike other attributes, we require that those having "~" be invisible to the user. This will be taken into account by the display algorithms which are still being designed.

4.6 Replacement declarations.

These rules are used displaying attribute values in a

shorthand fashion. To illustrate, the rule

```
from Category replace Noun with n;
```

indicates that Noun is usually written as n. in a dictionary

4.7 Tag Definitions

The tagset described in section 3 is in the process of being linked to the object system. This is achieved by defining each tag using tag definitions.

An example of a tag definition would be:

```
tagdef NNSM on Noun
{ Category = Noun;
  Type     = Common;
  Number   = Singular;
  Gender   = Masculine; }
```

A tag is defined on a particular class, and thus, can only include attributes which are allowable in that class. Therefore, a tag defined for the *Noun* class cannot be applied to a *Verb* class and vice versa. Moreover, the attributes in the tag defined on the *Noun* class must exist in the *Noun* class itself. Finally, the values specified must be normal values (not wildcards and no disjunctions).

5. Implementation

The project is in an intermediate state of implementation.

A database server provides the central storage for the system, and all transactions that take place are processed on the server.

There are currently two main databases: one for handling corpora as described in the section 2 of the paper, one for the lexicon, and one for have been created were created for this purpose. The corpus database holds information about corpus documents. one which maintains the wordlist and serves as an outside contact with the external world, and the other which serves as the actual lexicon storage, which in turn, can only be modified internally.

5.1 Corpus Database

In order to support multiple levels of annotation, our corpus is organized along two dimensions. At the top level is a catalogue of Level I raw text files. File information, which currently includes author, category of text and other attributes are maintained in a database table – as depicted by the table structure in figure 2 pointing to individual untagged files (shown shaded solid). In parallel to this, every raw file is associated with an arbitrary number of Level II files representing different levels of annotation. These different versions of the same file are depicted using different kinds of stripe in the same figure.

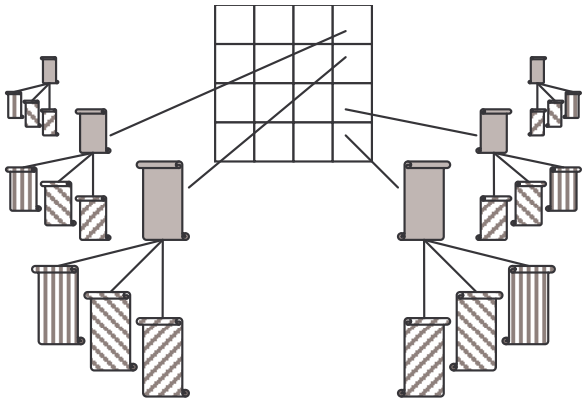

figure 2. Corpus Database

.

5.2 Lexicon Database

The lexicon itself is implemented as a database of entries which provide information about words. In order not to prejudge the kind of information that might be represented, we assume that, conceptually at least, it takes the form of attribute value pairs. We have seen how in section 3 how this works for morphosyntactic information, and subsequently, we hope to extend the approach to deal with the semantic and phonological properties of words.

The current implementation is structured around a general purpose database table containing information the word itself, its stem or root, and associated morphosyntax. The latter property was implemented as a long integer bit field, since this is extremely efficient for the purpose of when queries are made. Using a bit field to represent all the possible parts of speech implies the following assumption however: the total number of parts of speech values must not exceed 63. For simple parts of speech, this is more than enough. However, for more general attribute-value structures, it is is clearly inadequate.

Clearly, a database based on hierarchical feature structures provides for a more logical and elegant data representation. However, a disadvantage of such systems is that at the moment of writing is that they are not able to efficiently cope with large amounts of data.

A challenge is therefore going to be how to reconcile the efficiency of the relational database approach with flexibility of feature structures.

This can be fairly easily achieved by relational databases, though admittedly, these do not provide such an elegant and hierarchical structure to the data within since they are oriented towards general purpose solutions.

A issue which we will have to revisit is how to best harness the limited capacity but efficient bit string implementation with the more general, more elegant but less efficient representation offered by feature structures.

A simple list of records lying in a database can to some extent provide reasonable ‘dynamic’ structure through the use of queries. Thus, if for example, all the nouns are required, comparing a parts

of speech bit fields with each other should be fairly efficient. The drawback with this approach is that queries must be computed in real time.

5.3 Lexicon Editor

The system currently includes a Lexicon Editor (Led) whose main function is to enable a linguist to create and maintain lexical entries. Led uses ODL in two ways.

At start-up, the lexicon editor parses the ODL file containing all the class, rule and macro definitions. Then, it modifies its word-entry interface to reflect the classes found in the ODL file, together with its attributes and values each one can take.

Since the database back-end receives data which is already in relational database format, the lexicon editor takes on the important role of processing the ODL definitions which then are applied on new words, thus transforming them correctly into database records.

Apart from allowing additions and deletions, the lexicon editor also caters for the editing of entries as well as the construction of queries using a query editor. It must be said that the ODL file read in the editor at start-up is verified and checked to see that it has not be modified, as such modifications can compromise the lexicon. One central ODL file, residing on the server is maintained, and any remote lexicon editor starting up compares its ODL file version with the one maintained on the server. If the former or the latter differ, the ODL file on the server is automatically downloaded on the client, parsed and used to start up the lexicon editor.

5.4 Wordlist Database

The wordlist database, which essentially is open to anyone who would like to submit words, feeds the lexicon in a controlled manner, the lexicon administrators deciding which words should go in and which should be rejected

5.5 Website and Web Services

The system resides at http://mlrs.cs.um.edu.mt The website currently provides some basic public information about the project, access to ftp services for uploading source materials and downloading updated project documents..

Since the system will in the future be inevitably used by other applications or projects, which may or may not be written for the same framework in which the system in question is written, web services have been used wherever possible to mediate database access. Currently web services are mainly provided to support lexicon construction as discussed in greater detail in section 3 of this paper.

The starting point for lexicon construction is a wordlist, and we therefore provide a web services to support a web interface for extraction of words from documents, and maintenance of wordlists. The functionality of the service is implemented as follows:

1. User submits text, files or page URLs.
2. These resources are scanned and the words extracted from them and displayed.
3. User edits the resulting lists of extracted words manually
4. User submits final version for incorporation into the wordlist database.

The wordlist submission website also provides a simple interface for submitting documents to the Corpus Database mentioned in section 5.1

6. FUTURE WORK

6.1 Web Portal

Since we are developing a national resource, we place a very high priority on the public profile of this project. The current website does not yet add up to a portal containing a an integrated suite of tools and resources. Moreover, the contents are only partially accessible to members of the public.

Our immediate goals is to remedy this shortcoming by developing some simple public web services (as opposed to raw ftp) which will allow submission of primary content, and browsing of primary content filenames. For legalistic reasons, viewing of primary content will be restricted to registered users. A legal framework within which defines the copyright of submitted materials is currently in preparation.

In the medium term, the web services available will depend largely upon the availability of the associated linguistic tools. Some of these, (e.g. POS tagging) are relatively straightforward to implement Others (e.g involving morphological analysis) raise further questions and will take longer.

6.2 Lexicon Structure and Morphological Analysis

The present lexicon is full form. This means that even though words such as *kiteb* (book) and *kotba* (books) are closely related, they have entirely separate lexical entries. Clearly, there is considerable scope for the sharing of lexical information between such entries. We want to write the lexical information just once, associate it with the most general entry (in this case the singular word, or possibly the root k-t-b) and arrange for the information to be inherited by more the more specific variant or variants.

This has two implications. One concerns lexical structure and inheritance. If the information associated with lexical entries are classes, we need to elaborate not only on the inheritance mechanisms at work, but also on various categories of lexical entry. Which are the most general forms? Are there special classes which have the role of abstract classes not attached to an actual word - for example - triconsonantal roots? We are looking for an appropriate lexical structure, that is, a *system* of classes and associated inheritance mechanisms that fit the special case of Maltese.

The second issue concerns the ability to recognize the relationship between the word forms through morphological analysis. This is particularly important for a morphologically rich language like Maltese where a word can have tens and possibly hundreds of different forms. Unfortunately, a *computationally tractable*

description of Maltese morphology is not available off-the-shelf. Some preliminary work has been carried out using a finite state approach (Rosner 2003) but much work remains to be done.

6.3 Semantic Information: Wordnet

The incorporation of semantic information into lexical entries is not only important is not only a standard feature of dictionaries and thus in its own right, but essential for certain classes of application program. Examples include automatic query expansion (synonyms required), document classification (are the high frequency words associated with domain X or Y), and of course machine translation.

Fortunately, quite a lot of the work has already been done for other languages in this area by the WordNet community (Fellbaum 1998) who have succeded in creating a lexically based hierarchy of concepts called *synsets* for English together with various interfaces for accessing the information We intend in the first instance to include references to the EuroWordNet concept hierarchy (Vossen 1998) into our own lexical entries.

6.4 Linguistically Sensitive Tools

The longer term aim of the groundwork being carried out in this project is the creation of computational tools that make use of the information available in the lexicon. Many such tools are envisaged.

A tokeniser is already available. A part-of-speech tagger will become available before the project is finished, and this will enable is to tag large quantities of text.

In the immediate future, we hope to develop a chunker for recognizing noun and verb groups will be within our grasp soon afterwards. These will act as enablers for more elaborated document-processing tools including those for exploring semantic content.

In the medium term we see grammar and style checking as an achievable goal, but these depend upon the elaboration of an appropriate grammar model for Maltese. Such work involves exactly the kind of cross-disciplinary effort that the project is intended to promote.

7. CONCLUSION

This paper represents ongoing work, describing the first year of a two year project. Many of the loose ends that have been identified will be resolved before the end of the project. Others of course, remain long term research goals, and we hope to be able to participate in their achievement beyond the confines of the project.

8. REFERENCES

[1] Guy Aston and Lou Burnard. The BNC Handbook, Exploring the British National Corpus, Edinburgh University Press, 1998.

[2] Eric Brill, A Corpus-Based Approach to Language Learning, PhD Thesis, University of Pennsylvania, 1993.

[3] National Corpus with SARA. Edinburgh University Press, 1998.

[4] Dan Cristea and Cristina Butnariu. 2004. Hierarchical XML representation for heavily annotated corpora. In *Proceedings of the LREC 2004 Workshop on XML-Based Richly Annotated Corpora*, Lisbon, Portugal.

[5] Dalli, A. (2001) Interoperable Extensible Linguistic Databases, Proc. IRCS Workshop on Linguistic Databases, University of Pennsylvania

[6] Fellbaum, C, (1998) Wordnet, an Electronic Lexical Database, Cambridge: MIT Press.

[7] Gazdar G., Daelemans W. (1992) (eds), Computational Linguistics, vold 18.2 and 18.3, Special Issues on Inheritance I and II

[8] Paola Monachesi, Dan Cristea, Diane Evans, Alex Killing, Lothar Lemnitzer, Kiril Simov, Cristina Vertan. *Integrating Language Technology and Semantic Web techniques in eLearning*. To appear in *Proceedings of ICL 2006*

[9] Leech GN, Wilson A 1996 *EAGLES Recommendations for the Morphosyntactic Annotation of Corpora*. EAGLES-Guidelines EAG--TCWG--MAC/R. Final version of 3.1996. EAGLES, Istituto di Linguistica Computazionale, Pisa.

[10] Gatt, A., Vella, A., and Caruana, J. (2003). Annotating textual and speech data in Maltese. Technical Note ISO/TC 37/SC 4, International Standards Organisation Language Resource Management.

[11] Fabri, R. (1996). Kongruenz und die Grammatik des Maltesischen. Tubingen: Niemeyer.

[12] Kay, Martin (1985). "Parsing in functional unification grammar." In *Natural Language Parsing,* edited by David R. Dowty, Lauri Karttunen, and Arnold M. Zwicky, 251-278. Cambridge University Press

[13] TEI Consortium (2002), Guidelines for Electronic Text Encoding and Interchan (TEI-P4), University of Virginia Press

[14] Rosner M. (2003), Finite State Models of Linguistic Phenomena in Maltese, Proceedings of CSAW03, University of Malta.

[15] M. Rosner, J. Caruana, and R. Fabri. (1998) Maltilex: A computational lexicon for maltese. In M. Rosner, editor, Computational Approaches to Semitic Languages: Proceedings of the Workshop held at COLING-ACL98, Université de Montréal, Canada, pages 97–105, 1998.

[16] Russell, G., Carroll, J., Ballim, A ,.Warwick-Armstrong, S, (1992), A Practical Approach to Multiple Default Inheritance for Unification-Based Lexicons, Computational Linguistics, vol. 18.3, pp311-337.

[17] Shieber, Stuart M. (1984). "The design of a computer language for linguistic information." In Proceedings of 1984 International Computational Linguistics Conference, Stanford, pp 362-366

[18] Vossen, P, Introduction to EuroWordNet,, in Ide N., Greenstein D. and Vossen, P, (eds), Special Issue on EuroWordNet, Computers and the Humanities, vol 32 nos 2-3 pp 73-8

How Did I Find That? Automatically Constructing Queries from Bookmarked Web Pages and Categories

Chris Staff
University of Malta,
Department of Computer Science and AI,
Malta
cstaff@cs.um.edu.mt

ABSTRACT

We present 'How Did I Find That?' (HDIFT), an algorithm to find web pages related to categories of bookmarks (bookmark folders) or individual bookmarks stored in a user's bookmark (or favorites) file. HDIFT automatically generates a query from the selected bookmarks and categories, submits the query to a third-party search engine, and presents the results to the user. HDIFT's approach is innovative in that we select keywords to generate the query from a bookmarked web page's parents (other web-based documents that contain a link to the bookmarked web page), rather than from the bookmarked web page itself. Our initial limited evaluation results are promising. Volunteers who participated in the evaluation considered 20% of all query results to be relevant and interesting enough to bookmark. Additionally, 56.9% of the queries generated yielded results sets (of at most 10 results) containing at least one interesting and bookmarkable web page.

1. INTRODUCTION

Users who browse the Web store references to pages that they would like to revisit in their Web browser's Bookmark (of Favorites) file [1][2]. Sometimes, some users organise their bookmark file to collect related bookmarked Web pages into the same category, [1][2]. Frequently, bookmark files become disorganised over time, with categories containing references to pages about unrelated topics, and stale references (e.g., to pages that no longer exist) [9]. Some research has been conducted into why users bookmark Web pages and how they use those bookmark files [1][9][2]. There seems to be little research into how a user's bookmarks can be used to automatically find related content on the Web [6].

Manually finding Web pages that are interesting, relevant, and useful enough to bookmark usually takes considerable effort. Users must think of a query that might satisfy a requirement. Search results must be looked at to see if the query has really been satsified. The query may undergo a number of reformulations until a satisfactory Web page is found by a search engine. Once the user has visited the page, the user may bookmark it, to support a re-visit some time in the future. Sometimes a user may decide to search for more information related to a bookmark or a collection of related bookmarks. But all the effort that was put into deriving queries in the first place has been thrown away. Can the user remember what query was used to find each bookmark in the category? [14][5][10][5] store queries submitted during a user session that eventually results in a bookmarked page.

Rather than relying on user generated queries, 'How Did I Find That?' uses a Web page's 'context' to automatically generate a query. The query is submitted to a third-party search engine[1], and a page of up to 10 results is shown to the user. Eventually, 'How Did I Find That?' will be incorporated into a Web browser, and will automatically and periodically search for relevant web pages. For the time being, users can select individual bookmarks or whole categories of bookmarks as the basis of their search. A query is composed from the top ranking terms that occur in the context of the selected bookmarked web pages, and at most ten results are shown to the user. In our evaluation (see section 5), 56.9% of the results sets for 58 automatically generated queries contained at least one relevant result that was also bookmarkable (on average there were 1.72 bookmarkable results per results set).

This paper is organised as follows: Section 2 discusses similar work. We describe our approach to automatically generating queries from Web browser bookmark files in section 3. The evaluation approach and results are described in sections 4 and 5 respectively. Finally, we discuss the results and our future work in section 6.

2. BACKGROUND AND SIMILAR SYSTEMS

There is little research about automatically generating queries from a collection of bookmarks maintained by a user. Web browsers, such as Microsoft Internet Explorer, Mozilla, Mozilla Firefox, and Safari, contain tools for managing interesting pages that a user intends to revisit as bookmarks, but the tools are mostly deficient, unhelpful at reminding users how and in what context the pages had been bookmarked [9]. The bookmark files themselves are not without problems.

[1]For this study, we used Google at `http://www.google.com`.

Although users may organise bookmarks into a hierarchy of folders (or categories, as we call them here), effort is required to maintain them, and they can quickly become out of date or disorganised [1][2]. However, from an adaptivity point of view, bookmarks are useful because they contain documents that a user has found relevant and useful enough at some point to actually keep a record of them [7].

Queries to automatically search the Web are generated for a number of reasons, including helping users find relevant information while browsing [4][3][9][15], acting as a search intermediary [13][14][10][5], or creating and sharing paths through related material [6].

Queries may be automatically generated by identifying terms that describe a user's interest. Systems that help users to find relevant information while browsing typically use terms extracted from the pages that the user has visited recently. HyperContext identifies a "context path" as a collection of pages visited during a "context session" [15]. Nakajima, *et. al.*, defines context as the collection of pages that have been visited between a search page and a bookmarked page [10]. El-Beltagy, *et. al.*, define context as the description extracted from the centroid of a document cluster [7]. Bugeja also automatically extracts terms from web pages that are bookmarked in the same category and uses them to construct a query to find relevant web pages [5]. [14] incorporate an explicit query submitted by the user which is modified following relevance feedback .

The most common approach to identifying significant terms is based on a modified TFIDF [7][10][14][15][5], though others use, for example, a Simple Bayesian Classifier [3] or Association Rules [4]. In TFIDF, terms occurring in documents are ranked according to a weight that takes into account the frequency of term occurrence within a document (Term Frequency) as well as the number of documents in which the term occurs (Document Frequency) [12]. In information retrieval, a term that has a high frequency of occurrence within a particular (relevant) document and that occurs in few documents in the collection is considered to be a good discriminator between relevant and non-relevant documents, so the Inverse Document Frequency (IDF) is used to calculate a term weight. On the other hand, we are more interested in identifying those terms that occur in documents seen recently by a user (or which occur in some cluster or co-occur in some bookmark category) and that are likely to enable us to find more relevant documents. In this case, terms which occur frequently in individual documents *and* in a large number of documents seen by the user are likely to be good descriptors of a user's interest. Like Nakajima, *et. al.* [10], HyperContext extracts terms from documents visited during a "context session", but unlike Nakajima, *et. al.*, it does not require that a 'context' begins with a search page (so it can operate with no user input), and it does not count all the terms that occur in the documents. Rather, it identifies segments called "context blocks" around the links that are followed by users. Terms that are used in the automatically generated query occur frequently in context blocks as well as in a large number of documents visited during the context session [15].

3. APPROACH

Let's say that we have a Web browser bookmark category containing a number of bookmarks that have been manually bookmarked and organised by a user. If we can find out what it is about these Web pages that makes them related, then we should be able to generate a query that would return other related documents in a query results set.

We could have identified frequently occurring terms that occur in most of the documents contained in a bookmark category (or, simply, the highest occurring terms if a single bookmarked document is selected by the user), or created a cluster centroid from the category members, but instead we have chosen to apply a modified HyperContext [15] approach. In HyperContext, an accessed document is *interpreted* in the context of the link, or parent, that was followed to access it. An interpretation is a collection of keywords describing the document that are relevant in that context [15]. To generate a query from a bookmark category, we can create a category centroid based on the bookmarks' interpretations. HyperContext does not generate queries from bookmark files, and in 'How Did We Find That?' we do not have access to a user's path of traversal that eventually resulted in a document being bookmarked.

We have modified the HyperContext approach for 'How Did I Find That?' to look at a number of Web pages that contain a link to the bookmarked document (parents), because we do not know which link was followed by the user before the page that was bookmarked was accessed. We extract the region in each parent for each parent of each bookmark that contains the source of the link and create a centroid representation. We rank the centroid's (stemmed) terms (after stop-words have been removed). In this way, the query is automatically created from a bookmarked page's parents, rather than from the bookmarked page itself.

3.1 Processing Steps

HDIFT is a Python 2.3.4 program that interfaces with the Extended Boolean document indexing and retrieval system SWISH-E[2] and the Google Web API[3].

As shown in Fig. 1, there are six processing steps involved. We first enable a user to select a bookmark or category about which she would like to find similar documents. We then find twenty of each of the selected documents' parents, and process them to find the 'context blocks': the regions that contain links to the selected bookmarked web page. We merge the context blocks to create a centroid, and construct a query from the 10 top-ranking terms in the centroid. The query is submitted to Google through the API, and the results are displayed to the user.

The first step assumes that a user has chosen a bookmark or category for which she would like to find relevant Web pages. HDIFT then interfaces with Google through the Google Web API to identify the parents of a Web page (using the 'link:' operator in the Google query). As the processing overhead to identify a context block in a parent can be quite high, we limit the number of parents per bookmarked page to 20. As

[2] `http://www.swish-e.org`

[3] The Google API is available from `http://www.google.com/apis`. There is a limit of 1000 queries and 1000 results per query per day using the API.

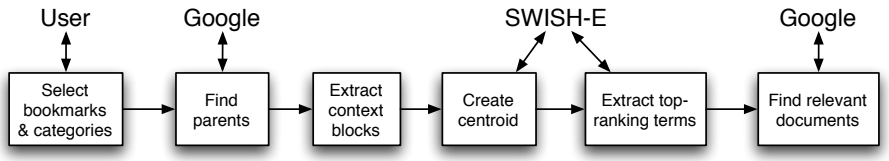

Figure 1: 'How Did I Find That?' processing steps.

these are ranked according to Google's PageRank algorithm [11], we have identified the 20 pages most likely to be used to access the bookmarked page.

The parents must be English-language documents. We define a document to be an English-language document if at least 5% of the terms in the document (ignoring HTML tags) are English language stopwords. If less than 5% of the terms in the document are English language stopwords, then we reject the document[4]. The next task is to identify the context blocks, the regions in each parent that contain a link to the bookmarked web page. We search through the text to identify the location of the link anchor, and once we find it, we scan the document to identify the context block. The context block is an HTML chunk between delimiters. Examples of delimiters are the heading tags (<H1>, <H2>, etc.), horizontal rule (<HR>), and table elements (<TR>, <TD>). In addition, we limit the context block to 100 words or less.

Once a context block is identified it is written to a text file for subsequent indexing. If a user selects a category of bookmarked pages, then rather than just one bookmarked page, each context block from each parent of each bookmark in the category is written to a separate text file for indexing. Once all the context blocks have been identified, SWISH-E is used to generate a central index. Stopwords are removed during the indexing process, and the terms are stemmed using SWISH-E's stemming option. Eventually, when the top-ranking terms are used in a Google query, the original terms, rather than their stems, must be submitted. Therefore, we also keep a map of the stems to their originals.

Once the central index has been created, we can process it to create a centroid representation of the cluster of context blocks to extract n terms with the highest rank to construct the Google query. SWISH-E provides a mechanism whereby all term information can be accessed. The frequency information is in the form of a set of document IDs and an indication of where in the document the term has occurred (e.g., in a title or heading, etc.) for each occurrence of a term. We convert the frequency information into a term weight, which takes into account each term's Document Frequency (expressed as a percentage of contexts of bookmarked web pages in the collection in which the term occurs), and derive an average weight for the term. We then extract the top 10 term stems with the highest average weights (after removing terms shorter than 2 characters in length), look up the original terms for each stem (ORing the terms if the stem has been mapped to more than one original), and concatenate the terms into a query string. A completed query string might look like 'firefox extension OR extensions download OR downloading OR downloads OR downloader tab OR tabbed OR tabs opera mozilla web windows OR window page OR pages links OR link google'.

Finally, we submit the query to the Google Web API using PyGoogle[5], and display the top 10 results to the user.

4. DATA COLLECTION

We evaluated 'How Did I Find That?' during September-October 2006 by asking volunteers to anonymously upload their Bookmark (or Favorites) files to the HDIFT Web server[6] and then to select one or more bookmarks or categories of bookmarks to run HDIFT against. Volunteers were recruited by e-mail from undergraduate and postgraduate students in the Department of Computer Science and AI at the University of Malta, the WebIR mailing list[7], and lecturers and students in all disciplines at the University of Malta. A total of 20 unique bookmark files were uploaded. Once a bookmark file was uploaded, each volunteer was given a unique HDIFT ID to enable them to return to view their own results and give feedback.

Once a selection was made, a server-side script extracted the URLs associated with the bookmark or category and prepared a file for input to the HDIFT algorithm. The HDIFT algorithm generated a query and submitted it to Google, using the method described in section 3.1, and stored the results of the query on the server. As HDIFT generates queries based on frequently occurring terms that appear in the context blocks of the bookmarked web pages' parents, if a page does not have parents (a request to Google with the 'link:' operator finds no documents) then it may not be possible to generate a query. In addition, a generated query may have no results returned by Google. Each query has a maximum of 10 results, equivalent to the first page of results that Google returns. It is possible that there are more than 10 results, but given Jansen's *et. al.* findings [8] we assume that for the majority of users, there should be relevant information in the first 10 results. There are five feedback levels: 'I would bookmark this', if the page was relevant and the user would bookmark it; 'Relevant, but I wouldn't bookmark it'; 'Not given yet', a default value to indicate that the user has not evaluated the result; 'Error opening page', because it is possible for a page to be off-line or removed; and 'Not relevant'. Evaluators were asked to try to evaluate as many results as they could. Once feedback had been submitted, it was not possible to modify it.

[4]Note that we do not perform a language test on bookmarked Web pages themselves.

[5]PyGoogle is a Google Web API wrapper for Python available from `http://pygoogle.sourceforge.net/`

[6]`http://poseidon.cs.um.edu.mt/~csta1/hdift/uploadbk.php`.

[7]`http://groups.yahoo.com/group/webir/`

Volunteers could select as many categories and bookmarks as they liked. In addition, volunteers could return on up to five separate occasions to make additional selections. Finally, there was no way to limit file uploads (unless we ran out of HDIFT IDs), so it is possible for the same volunteer to upload the same bookmarks file several times and make more selections. We could tell if the bookmark files were bit identical that they were probably submitted by the same person, but this happened only once, and selections were only made off one of the duplicate bookmark files anyway.

In all we collected 20 bookmark files, at least one selection was made off 16 of them, 267 valid queries were generated in all (a valid query has at least one Google result) and feedback was given on 58 of the valid queries (21.7%). In the next section, we evaluate the results of the valid queries.

5. RESULTS

Although only limited feedback was obtained (only 20 bookmarks files submitted, and feedback on results given on only 58 out of 267 valid queries), the results are promising. A detailed breakdown is given later in this section, but overall, Google returned 502 results in all for the 58 queries. Of these, 19.9% were considered good enough to bookmark, 23.9% were relevant but the user did not think they were worth bookmarking, and 38.5% were not relevant. Feedback was not given at all on 16.5% of the results, and there were HTTP problems with the remaining 1.2% (Google would have returned the page in the results set, but the page may have been non-responsive when the user tried to access it). Furthermore, the results sets for 56.9% of the queries on which feedback was given contained at least one result that the user considered worthy of bookmarking.

5.1 Analysis of the Bookmark Files and Overall Selections

In all, 20 bookmark files were uploaded, with users making selections of categories and individual bookmarks from 16, and giving feedback on the results of queries generated from selections made from 7 bookmark files. In all, 145 categories, containing an average of 16 bookmarks each, and 331 individual bookmarks were selected from 16 bookmark files, yielding a total of 476 attempts to generate a query. There were 267 valid queries generated (queries which when submitted to Google returned at least one result), meaning that there were 209 failed attempts. The failed attempts can be differentiated between failure to generate a query, and failure to obtain results following a query. We generated a query but failed to obtain results 11 times for category selections and 18 times for individual bookmark selection. We failed to generate a query 13 times for selected categories, and 167 times for selected individual bookmarks. We explain the high failure rate for individual bookmarks in subsection 5.2.

There were 121 valid queries (i.e., queries with results) from selected bookmark categories, and 146 valid queries from selected individual bookmarks. Feedback was given on the results of 58 queries in all, 31 queries generated from categories and 27 generated from individual bookmarks. In all, 502 results were provided for the 58 queries, and feedback was given on 413 of them.

As shown in Table 1, more results overall are considered bookmarkable when the query is generated from a category, rather than from an individual bookmark. The number of non-relevant results increases when the query is generated from individual bookmarks. On average, 1.72 results per query were considered bookmarkable, with 0 the lowest and 6 the greatest number of bookmarkable results per query. We also measured the degree of satisfaction with the results by the rank in which Google returned results (Fig. 2).

We can see that the number of non-relevant results (Fig. 2c) seems to be unaffected by rank. The number of bookmarkable results (Fig. 2a) is highest at P1 independently of whether the query is generated from a category or an individual bookmark.

5.2 Discussion of Results

Apart from the low number of responses to the request for volunteers to assist with the evaluation of HDIFT, and the large number of user selections from bookmark files made without feedback being given on the results, the most significant figure is the failure to generate a query for a high number of selections made from bookmark files. In all, out of 476 bookmark selections made, 209 (43.9%) failed to result in a generated query. This was a far greater problem for individual bookmarks than for categories of bookmarks, with only 13 failed queries out of 145 (9%) selected categories, but 167 failed queries out of 331 (50.5%) selected individual bookmarks. We analysed the reasons for the high percentage of failed queries for individual bookmarks and discovered that 47 (28.1%) HTTP requests for the Web page resulted in an error code; the servers hosting 6 (3.6%) pages were unresponsive at the time of the request, and for the remaining 114 (68.3%), a 'link:' request to Google for documents linking to the page yielded no results.

Feedback was given on the results of 58 queries. In all there were 413 Google results generated on which users gave feedback for the 58 queries (83 additional results were not given feedback on, and six results gave the user an HTTP error code on access). Of these 413 results, only 7 web pages in the results already existed in the users bookmark file (in the same category or individual bookmark that was used as a basis to generate the query). The feedback given on these results have not been separated out from the global results given above. Interestingly, and even though the users had already bookmarked these web pages for the selected category or individual bookmark, only two of the results were considered interesting and bookmarkable. Another two results were considered relevant but not bookmarkable, and another two were considered not relevant! User feedback was not given on the final, already bookmarked, result. In future, we will remove Google results that already exist in the same user selection.

6. FUTURE WORK AND CONCLUSIONS

We have described 'How Did I Find That?' (HDIFT), an algorithm to find Web-based material that is related to Web pages that a user has bookmarked in the past. A user can select a category of bookmarked web pages, or individual bookmarked web pages from their personal bookmark file and HDIFT will automatically generate a query based on the selection, submit the query to Google, and present the

Table 1: Overall Feedback Levels

	'I would bookmark this'	'Relevant, but I wouldn't bookmark it'	'Not relevant'
Total = 413	100	120	193
Total %	24.2	29.1	46.7
Queries from Categories only %	27.4	31.5	41.1
Queries from Individual Bookmarks only %	21.3	26.9	51.9

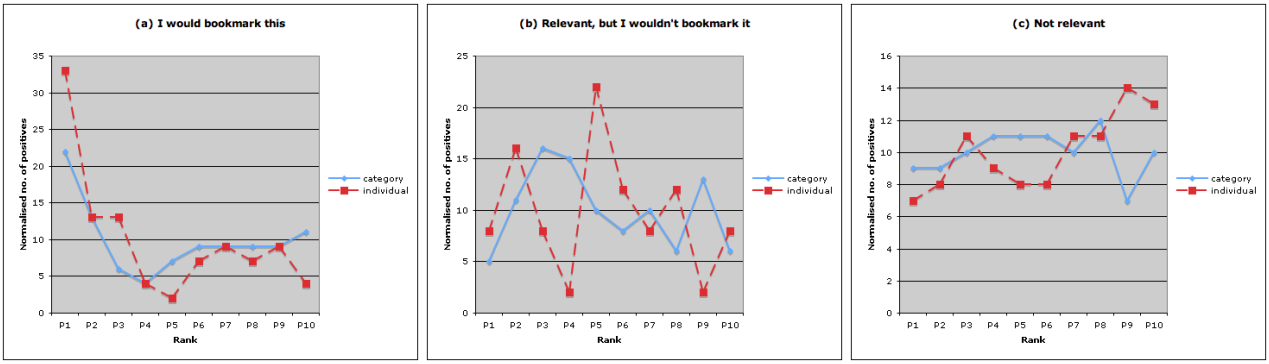

Figure 2: Feedback levels according to results rank.

results to the user. Rather than generating the query directly from the bookmarked web pages, we download up to 20 of the document's parents (found using Google's 'link:' modifier) and create a centroid representation of the context. We use the centroid representation to construct a query. Although the number of participants in the evaluation was low, results are promising and indicate that HDIFT is able to find relevant bookmarkable web pages. The results appear to be equally good for queries generated from categories of bookmarks and from individual bookmarks, although an issue still to be resolved is the inability to generate a query for an individual bookmark if Google cannot find any parents for it.

We intend to conduct studies with a smaller group of people to compare HDIFT with the results of extracting terms from the centroid of documents in category, and manually generated queries (by the participants) from the same category. In the same study, we will identify synonyms in the bag-of-words representation of the centroid and the generated query so they can be ORed in the query. We intend to analyse the bookmark files for information about the frequency with which bookmarks are added; the order in which they are added; average gaps between returning to a category to add new bookmarks; the number of stale links in bookmark files, etc. Finally, we intend to utilise the HDIFT algorithm to perform automatic bookmark classification to help users keep bookmark files automatically organised.

7. REFERENCES

[1] D. Abrams and R. Baecker. How people use WWW bookmarks. In *CHI '97: CHI '97 extended abstracts on Human factors in computing systems*, pages 341–342, New York, NY, USA, 1997. ACM Press.

[2] D. Abrams, R. Baecker, and M. Chignell. Information archiving with bookmarks: personal web space construction and organization. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–48, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[3] D. Billsus and M. Pazzani. Learning probabilistic user models. In *Proceedings of the Workshop on Machine Learning for User Models, International Conference on User Modeling*. Springer-Verlag, 1997.

[4] D. Boley, M. Gini, R. Gross, E.-H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the World Wide Web using WebACE. *AI Review*, 13(5-6):365–391, 1999.

[5] I. Bugeja. Managing WWW browser's bookmarks and history (a Firefox extension). Final year project report, Department of Computer Science & AI, University of Malta, 2006.

[6] P. Dave, I. Paul Logasa Bogen, U. P. Karadkar, L. Francisco-Revilla, R. Furuta, and F. Shipman. Dynamically growing hypertext collections. In *HYPERTEXT '04: Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*, pages 171–180, New York, NY, USA, 2004. ACM Press.

[7] S. R. El-Beltagy, W. Hall, D. D. Roure, and L. Carr. Linking in context. In *HYPERTEXT '01: Proceedings of the twelfth ACM conference on Hypertext and Hypermedia*, pages 151–160, New York, NY, USA, 2001. ACM Press.

[8] B. J. Jansen, A. Spink, J. Bateman, and T. Saracevic. Real life information retrieval: a study of user queries on the web. *SIGIR Forum*, 32(1):5–17, 1998.

[9] W. Jones, H. Bruce, and S. Dumais. Keeping and re-finding information on the web: What do people do and what do they need? In *ASIST 2004 Annual Meeting, Managing and Enhancing Information: Cultures and Conflicts*, November 2004.

[10] S. Nakajima, S. Kinoshita, and K. Tanaka. Context-dependent information exploration. In *Proceeding of the the 11th World Wide Web Conference (WWW2002)*, New York, NY, USA, 2002. ACM Press.

[11] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[12] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[13] A. Sieg, B. Mobasher, S. Lytinen, and R. Burke. Concept based query enhancement in the ARCH search agent, 2003.

[14] G. Somlo and A. E. Howe. Querytracker: An agent for tracking persistent information needs. In *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 488–495, Washington, DC, USA, 2004. IEEE Computer Society.

[15] C. Staff. *HyperContext: A Framework for Adaptive and Adaptable Hypertext*. PhD thesis, University of Sussex, 2001.

Automatic Classification of Web Pages into Bookmark Categories*

Chris Staff
University of Malta,
Department of Computer Science and AI,
Malta
cstaff@cs.um.edu.mt

ABSTRACT

We describe a technique to automatically classify a web page into an existing bookmark category whenever a user decides to bookmark a page. HyperBK compares a bag-of-words representation of the page to descriptions of categories in the user's bookmark file. Unlike default web browser dialogs in which the user may be presented with the category into which he or she saved the last bookmarked file, HyperBK also offers the category most similar to the page being bookmarked. The user can opt to save the page to the last category used; create a new category; or save the page elsewhere. In an evaluation, the user's preferred category was offered on average 67% of the time

1. INTRODUCTION

Bookmark management systems that can help classify bookmarked web pages, track web pages that have moved since they were bookmarked, help a user to find web pages similar to pages that were bookmarked, and that generally assist with their own organisation are becoming increasingly important. Recent surveys indicate that a user's bookmark file contains on average 184 entries [4], and that approximately 73.7% of pages visited are page revisits [8], with interaction through either a bookmark file, or the history list of recently visited sites, or the browser's back button being the most common ways of revisiting a page. Modern Web browsing software, such as Mozilla, Microsoft Internet Explorer, and Safari, provide only limited support for automatic management of bookmarked web pages (see section 2). Even less support is provided for navigating through the list of recently visited web pages to enable a user to return to a recently visited page. Bugeja's HyperBK [3] addresses some of the issues. A moved bookmarked Web page can be tracked via a third-party search engine. A user can be reminded of the query that had been used to find a web page before it was bookmarked, or HyperBK can suggest a query to use to find web pages similar to a category of bookmarked web pages. HyperBK provides a variety of perspectives or views to potentially make it easier for a user to find an entry in the list of recently visited web pages. Finally, HyperBK automatically suggests a bookmark category into which to store a web page whenever the user requests to bookmark a web page. This last feature is the subject of this paper.

In section 2 we discuss general bookmark management issues. We describe HyperBK in section 3. The web page classification algorithm is described in section 4, and results of the evaluation are presented in section 5. Similar systems are reported in section 6. We give our future work and conclusion in section 7.

2. BOOKMARK MANAGEMENT ISSUES

Bookmark files tend to be partially organised, with some web pages carefully clustered into a single category or a hierarchy of categories [2]. Many other web pages are not assigned to any category in particular, and are either lumped together in the top-most category or into a generically named category (e.g., 'My Favourite Pages'). The complete or partial URL of frequently used bookmarks may be accurately remembered so that as it is being keyed into the browser's address bar, the browser will assist with a simple URL completion task. Infrequently visited, but bookmarked, web pages may be easy to find, if they have been stored in the category the user is looking through, but frequently, the page will not have been placed into its most relevant category, and may prove difficult to find again. If pages are (almost) always saved to the most relevant bookmark category, then they may be easier to find again in the future. As bookmark files tend to be poorly organised, the user could probably do with some assistance. Safari, Mozilla FireFox, and now Microsoft Internet Explorer show the last category saved to, instead of the root category, so the user must either knowingly save a bookmark to the wrong category, or must otherwise locate or create a more appropriate category.

Every once in a while, a user may wish to update a category by looking for more related information. Here, a frequently or infrequently visited category or page may be selected, and the user will try to remember the strategy that was used to find this page in the first place. If the user had foresight,

*This paper is based on the work of Ian Bugeja, a BSc IT (Hons) student at the University of Malta, who built the system described in this paper for a Final Year Project under my supervision.

he or she may have also bookmarked the results page of the search engine query that had been used, so that rather than trying to remember the query, the URL containing the query can simply be re-requested. This makes a number of assumptions: that the user remembered to bookmark the query; that the page bookmarked was actually related to the query; and that the user wants to find pages that are relevant to a single page (many search engines allow a straightforward search for pages similar to a given URL). Sometimes, however, a user may wish to find pages that are similar to a category or cluster of related bookmarks.

Other bookmark management related issues are concerned with the freshness or currency of the bookmarked URLs. Web servers do go down or are renamed. Pages may exist for a short period of time. The page contents may change, even though the URL continues to exist. Hard disk space is becoming incredibly cheap. Given the relatively small number of bookmarks on average, and the small average size of bookmarked files, it should not be too expensive to allocate a permanent local cache for web pages that have been bookmarked. If the address of the original of the bookmarked page is later changed, then rather than having only a vague memory of what important thing the page contained, the page can be reloaded from the local cache. In addition, assistance can be given to automatically relocate the page on the Web (if it still exists).

Web browsers and hypertext systems in general have always had problems finding a suitable interface to help users navigate their way around recently visited nodes. Historically, this was identified as one of the issues leading to the 'Lost in Hyperspace' syndrome [5], in particular because it is difficult to build a representation of the user's recent browsing activity that accurately matches his or her mental model of the recent past. Web browsers have tended to adopt a flat, linear representation of what may be a complex user session including cycles, backtracking, and branching. Current Web browsers have increased the complexity of the problem by retaining the flat, linear path representation, even though users can now have multiple tabbed windows and multiple windows, potentially containing concurrent tasks that may or may not be related to each other. The problem of searching for information in history is likely to increase in the future, as systems such as MyLifeBits [7] contemplate digitally storing *everything* seen by a user throughout his or her lifetime.

3. BACKGROUND TO HYPERBK

HyperBK was developed as a Mozilla Firefox extension[1] to provide simple mechanisms to address some of the issues referred to in section 2, but the primary motivation was to automatically classify a web page that a user in the process of bookmarking, according to the bookmark classifications or categories that the user has already created. We describe the automatic classification task more fully in section 4, but here we give a brief overview of the other tasks.

3.1 Views of the History List

[1]HyperBK is available at https://addons.mozilla.org/firefox/2539/

Bugeja provides multiple representations of the history list. On visiting a page, the page is classified to find a suitable category in the bookmark file, just in case it will be bookmarked. Apart from the ability to see the recently visited pages in the order of most recently visited, the history list resembles the bookmark file, with visited pages allocated to bookmark categories (see Fig. 1). This enables users to locate visited pages according to their topic. This does require a high degree of correctness in the classification of web pages, otherwise pages will be stored in the wrong category and users will have difficultly using the topic-related view to locate previously visited web pages. Other filters available to find visited pages are by frequency count, to quickly identify frequently visited pages; by keyword, a representation of the contents of the page are kept so keyword search is possible on the content of visited pages, rather than just their title; and, finally, by co-incidence. In this last case, we assume that the user cannot remember any distinguishing characteristics of the visited web page, but remembers another page visited in roughly the same period. This other page can be located using the tools described above, and then a time filter can be used to pull out the other pages visited just before or after the remembered page. One possible weakness of Bugeja's approach, which is inherited from the way Web browsers, and Mozilla in particular, implement their history lists, is that only the last visit to a web page is recorded. If the user is looking for web page, P_1, which has been visited at time T_1 and again at the later time T_2, and the user remembers visiting another page, P_2 close to time T_1, and T_2 - T_1 is larger than the time filter, P_1's only recorded visit will be at time T_2, so looking up P_2 will not help the user to find page P_1. However, if P_2 is a very frequently revisited page, then remembering it is also of little use, because the user will have far too many instances to choose from. The best page to remember is an infrequently visited page (preferably just once), so that remembering it will lead to the page P_1 that the user seeks.

3.2 Web Page Tracker

A Web page is bookmarked because the user intends to revisit the page [1] and so saves the page's address in a bookmark or favorites file. However, Web pages may be moved to a new address by the maintainers of the web page, in which case the stored address is out of date, and the user may be given a server response that the page cannot be found when the user next tries to revisit it. HyperBK stores bookmarked web pages in a local cache, to enable the user to see the contents of the page even if the page is no longer available online, and also to assist with the automatic rediscovery of the page on the internet (if it still exists and is indexed in its new location by the search engine). We do not distinguish yet between a dynamic page and a static page. It may be useful to do so, because if a dynamic page changes frequently, then it may be likely that the page has been bookmarked for reasons other than the content, and attempting to rediscover a dynamic page according to its content is likely to fail.

3.3 Search for Related Pages

Each time a web page is bookmarked two things happen. First, we track back through the pages that the user has visited that act as 'Search Engine (SE) Referrers' (that is, a visited page that contains a link to the next visited page)

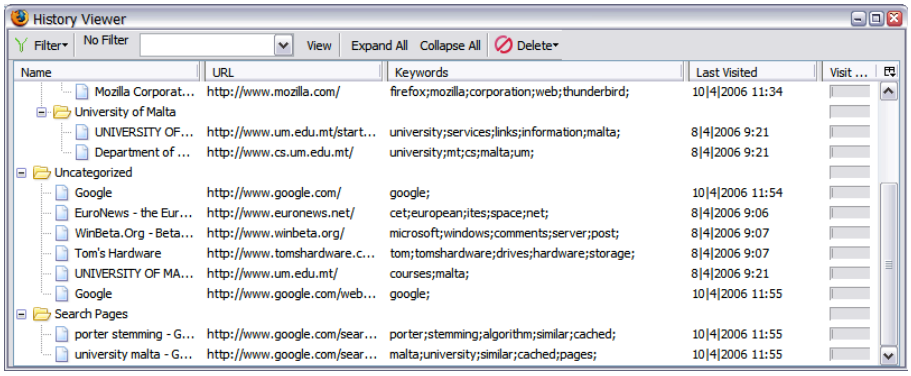

Figure 1: History viewer (from [3]).

until we have a chain of visited pages with either a search engine results page as the root or else the first page visited in the chain. It is possible to correctly identify the SE Referrer, even if a session is split across multiple tabs and windows, and even if a session is started in one tab or window and continues in another tab or window, as long as a link is actually followed (rather than, for instance, copying a URL and pasting it into an address bar). If a search engine results page is the root of the chain, then the keywords that were used to generate the results page are extracted and stored. The second thing that happens is that keywords from the bookmarked page are extracted and stored. These keywords are used to perform a keyword search on the history list (while the page is still in the history); to rediscover the address of a bookmarked web page if the web page has changed address; to generate a query to search for similar pages on the web; and to contribute to the description of the category to which the bookmark belongs (if any) to i) generate a query to search for web pages that could belong to the category on the web, and ii) to decide if a web page in the process of being bookmarked could belong to this category.

A user can invoke the contextual menu from a category name in the bookmark file and select the "See Also" option to search the Web for web pages similar to the bookmarks in that category (see Fig. 2). HyperBK can use either the query that the user had originally used to find web pages that were eventually bookmarked into the category (the Search Engine Referers described earlier in this section), or else HyperBK can construct a query from the keyword representations of web pages in that category. In a limited evaluation involving 7 users, users claimed that they were "Very Satisfied" or "Satisfied" with 82% of overall results of the queries generated automatically from categories in their bookmark files (the use of previous user created queries has not yet been evaluated, as it requires reasonably long term use of HyperBK).

HyperBK currently submits the automatically generated query to Google's Web Directory, rather than to the Google Search Engine *per se*, because in initial personal trials Bugeja obtained better results from the Web Directory ([3], pg. 71). The algorithm is being improved to take context into account, and to obtain useful results even when the query is submitted to the Google Search Engine [13].

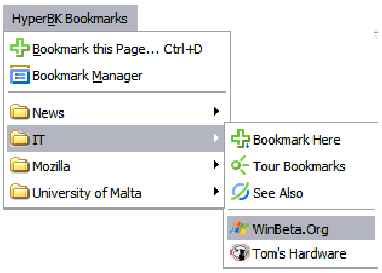

Figure 2: HyperBK Bookmark menu (from [3]).

4. CLASSIFYING WEB PAGES TO BE BOOKMARKED

Each web page that is accessed is parsed using the Document Object Model (DOM) to extract the text components. Stop words, HTML and JavaScript tags are removed, and the remaining words are stemmed using Porter's Stemming Algorithm [11]. According to Bugeja, the five stems with the highest frequency are selected to be representative of the page, but only if they have a frequency of at least two, otherwise only three stems are selected. This helps to keep down computational costs. If a web page author has used META keywords in his or her document, then the five META keywords that most frequently occur in the document are used as the document representation instead.

In a future experiment, the keyword selection process will be modified. First, we will segment a document into its component topics, and extract keywords from the topic most likely to be relevant to the user. Although the results obtained by evaluation are good (see section 5), we think we can improve on them. Picking the top five ranking terms will not necessarily accurately capture the user's interest in the page. A web page is likely to consist of more than one topic, and it is unlikely that a page has been bookmarked because the user is interested in each topic. If only high ranking terms are selected, then potentially, the terms individually represent different topics that occur on the page, so some of the selected terms may, in fact, be irrelevant to the user. We will build upon HyperBK's ideas related to the SE Referrer to find out which keywords were used on a search engine query page which ultimately led to the current page. We can then use these keywords to represent the user's interest

in the page. Alternatively, we can examine the parent that was used to access the current page, extract the region surrounding the link and assume that terms occurring in that region accurately describe the potential interest in the to-be-bookmarked page. Finally, we can segment an accessed page into its component topics, merge similar topics, and generate a representation for each topic.

Ultimately, we want to recommend that the user saves the bookmark entry into a particular category. Fig. 3 shows the 'Add Bookmark' dialog box. The algorithm described below is used to select a candidate category in which to store the bookmark. A thumbnail of the page is shown in the top-left hand corner of the dialog box, and the candidate category is highlighted in the 'Bookmark Location'. If the user presses the 'OK' button, the bookmark will be stored in this location. Alternatively, the user can either search for a more appropriate location to store the bookmark; create a new category; or, save the bookmark to the last used location by pressing the 'Bookmark in...' button (in this case, the last location was 'University of Malta').

The algorithm used to select the candidate category is based primarily on simple keyword matches. As was described above, each time a web page is accessed, representative keywords are extracted and stored. If the page is bookmarked into a category, these terms are added to the set of terms that represent the category. The recommended category is the category that has the greatest number of keyword matches for the incoming web page. If no category scores highly enough (matches a high number of page terms), then the recommended category will be the category that contains another page from the same domain, as long as the category and incoming page share at least one keyword in common. Finally, if even this fails, then the title of page is compared to the category descriptions. The category with the highest number of matches is recommended.

The HyperBK approach does not automatically create categories and pick reasonable category names, meaning the user must be involved in creating categories and keeping the bookmark file organised by allocating pages to the correct bookmark category until there are a sufficient number of categories and bookmarked pages to make the recommendations accurate. A future experiment is planned to identify the smallest number of categories and category members to make recommendation feasible.

HyperBK includes a wizard to assist with the importation of bookmark files from other web browsers, and to assist with the automatic segmentation of categories if they become too large. Bugeja recommends that a category is split into subcategories once it reaches a membership of 20 pages, but he doesn't give reasons why 20 is a good number to split on. This setting is user changeable.

5. EVALUATION

We decided that the most appropriate way to evaluate the classification algorithm, apart from a longitudinal study in which HyperBK users would evaluate the system *in situ* over a period of time, would be to collect real user's bookmark files to see if HyperBK could assign bookmarked pages to categories in the same way that the users did. This means that we require bookmark files to be organised (and to contain some categories), and we assume that the user has assigned each bookmarked page to the correct category. Of course, this is a weak assumption, but we had insufficient time to conduct a longitudinal study. Bookmark file submission was conducted anonymously through a specially created portal.

Students following the BSc IT (Hons) degree programme at the University of Malta were invited by e-mail to submit their bookmark files (regardless of which Web Browsing software they used). Of approximately 200 students contacted, 30 submitted their bookmark files (a return of about 15%). Of these, 22 files were considered inappropriate for use because they did not contain more than one or two categories (the files were too loosely structured) and we felt that including them in the evaluation could unfairly bias the results in HyperBK's favour.

We randomly removed 10 URLs from categories of 5 of the remaining 8 bookmark files. We removed less than 10 URLs from the other three: in two cases because there were too few categories overall and in the third case (79231 in Table 5) because although there were many more categories, most of them contained bookmarks that appeared to be mostly unrelated. The challenge was to place the randomly selected bookmarks into the same categories that the users had placed them. The results are given in Table 5.

Figure 5 plots the precision against categories. We would hope for a generally high precision, perhaps dropping slightly as the number of categories grows, especially if categories become less distinguishable from each other (because only 5 terms are selected to describe a page). For two bookmark files containing 38 and 45 categories, precision drops to below 0.7, which is probably unacceptably low. However, one of the two bookmark files contains many similar categories, and the other had many categories each containing unrelated bookmarks.

6. SIMILAR SYSTEMS

Bugeja [3] has given a recent, short survey on bookmark management systems. Bugeja notes that bookmark management systems are usually offered as stand-alone systems - unlike HyperBK, none is integrated into a browser, and as discussed in [3], web browsers offer minimal bookmark management facilities. Most of the systems Bugeja looks at do not offer automatic web page classification features, although Abrams, Baecker, and Chignell [2] list some requirements for bookmark management systems. Among their requirements are improving the organisation of bookmarks on behalf of the user, possibly by automatically "filing" new bookmark entries, and integrating the bookmark management system with a web browser. Feng and Brner [6] use "semantic treemaps" to categorise bookmark entries. Nakajima *et. al.* use keywords that appear in a document's "context" to to automatically construct queries, rather than to classify a document, where a context is composed of the pages visited between a search engine's query page and a page that is bookmarked [10]. Li and Yamanishi [9] use a "finite mixture model" to classify documents, but as Bugeja points out, this requires the prior existence and standard description of categories in which to place documents. On the

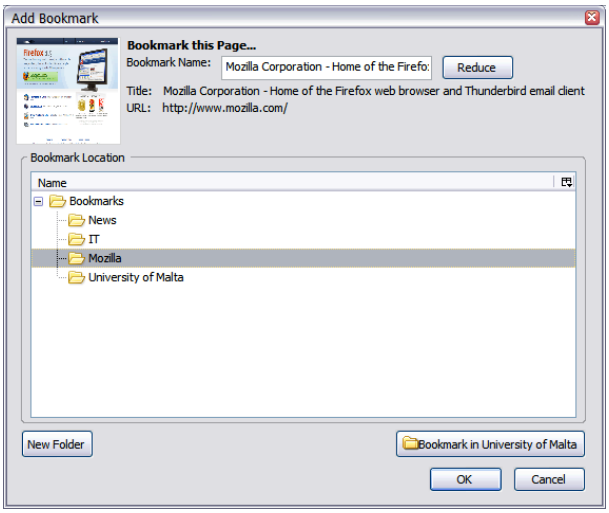

Figure 3: 'Add bookmark' dialog (from [3]).

Table 1: Classification Evaluation Results (from [3]) (Legend: 'BKs' = Total Bookmarks; 'Cat.' = Total Categories; 'Hits' = bookmarks allocated into correct category; 'Misses' = bookmarks allocated wrongly; 'Near Hits' = bookmark allocated to a parent category (excluding the Bookmarks Root); 'Approx Precision' = Near Hits+Hits/total; 'Precision' = Hits/total.)

ID	BKs	Cat.	Root BKs	Root Cat.	Hits	Misses	Near Hits	Approx. Precision	Precision
23740	425	105	49	33	7	2	1	0.80	0.70
24166	330	64	2	26	8	0	2	1.00	0.80
88014	240	53	21	12	7	2	1	0.80	0.70
23248	197	45	6	9	5	3	2	0.70	0.50
79231	158	38	18	18	4	3	0	0.57	0.57
58917	139	29	21	11	8	2	0	0.80	0.80
76243	38	11	3	11	5	0	0	1.00	1.00
80999	22	7	1	5	4	0	0	1.00	1.00
				Totals	48	12	6	6.67	6.07
				Averages	4.8	1.2	0.6	0.67	0.61

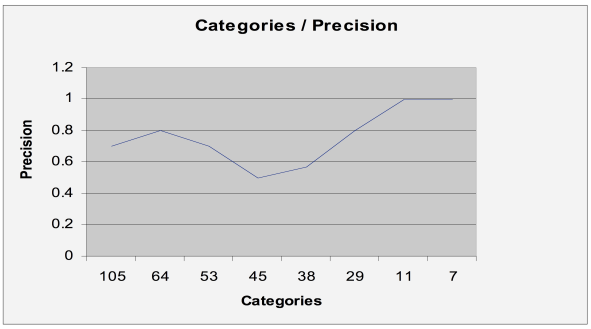

Figure 4: Categories v. Precision (from [3]).

other hand, Shen, *et. al.* [12] use a page summary on which to base a classification. Google also offers bookmark hosting[2], but whether they plan to offer automatic bookmark management remains to be seen.

7. CONCLUSION AND FUTURE WORK

Automatic bookmark file classification could be a useful extension to web browsers. Instead of just offering the last category used to store a bookmark, or dumping the newly created bookmark into a default location, HyperBK presents the user with a candidate category based on a simple Boolean matching algorithm, which has been extended to also consider the domain names of previously bookmarked pages and keyword extraction from titles. This simple algorithm gives reasonable accuracy. In an experiment, 67% of bookmarks were classified correctly. There is room for improvement. We intend to perform topic segmentation on web pages to extract keywords from the topic most likely to be of interest to the user, and we intend to carry out a more extensive evaluation. Instead of classifying a random selection of URLs from a user's bookmark file, we will attempt to allocate all the bookmarked pages to the user selected category in the order that they were really allocated. This will help us determine the average minimum category membership size required to consistently place a page in the correct category.

8. ACKNOWLEDGEMENTS

This paper is based on 'Managing WWW Browser's Bookmarks and History: A Firefox Extension', Ian Bugeja's Final Year project report [3]. Ian was a BSc IT (Hons) student under my supervision in 2005-06.

9. REFERENCES

[1] D. Abrams and R. Baecker. How people use WWW bookmarks. In *CHI '97: CHI '97 extended abstracts on Human factors in computing systems*, pages 341–342, New York, NY, USA, 1997. ACM Press.

[2] D. Abrams, R. Baecker, and M. Chignell. Information archiving with bookmarks: personal Web space construction and organization. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 41–48, New York, NY, USA, 1998. ACM Press/Addison-Wesley Publishing Co.

[3] I. Bugeja. Managing WWW broswer's bookmarks and history (a Firefox extension). Final year project report, Department of Computer Science & AI, University of Malta, 2006.

[4] A. Cockburn and B. McKenzie. What do web users do? an empirical analysis of web use. *Int. J. Hum.-Comput. Stud.*, 54(6):903–922, 2001.

[5] J. Conklin. A survey of hypertext. Technical Report 2, Austin, Texas, 3 1987.

[6] Y. Feng and K. Brner. Using semantic treemaps to categorize and visualize bookmark files. In *Proceedings of SPIE - Visualization and Data Analysis*, volume 4665, pages 218–227, January 2002.

[2]http://www.google.com/bookmarks

[7] J. Gemmell, G. Bell, R. Lueder, S. Drucker, and C. Wong. MyLifeBits: fulfilling the Memex vision. In *MULTIMEDIA '02: Proceedings of the tenth ACM international conference on Multimedia*, pages 235–238, New York, NY, USA, 2002. ACM Press.

[8] E. Herder. *Forward, Back, and Home Again - Analysing User Behavior on the Web*. PhD thesis, University of Twente, 2005.

[9] H. Li and K. Yamanishi. Document classification using a finite mixture model. In *Proceedings of the 35th annual meeting on Association for Computational Linguistics*, pages 39–47, Morristown, NJ, USA, 1997. Association for Computational Linguistics.

[10] S. Nakajima, S. Kinoshita, and K. Tanaka. Context-dependent information exploration. In *Proceedings of the the 11th World Wide Web Conference (WWW2002)*, New York, NY, USA, 2002. ACM Press.

[11] M. F. Porter. An algorithm for suffix stripping. pages 313–316, 1997.

[12] D. Shen, Z. Chen, Q. Yang, H.-J. Zeng, B. Zhang, Y. Lu, and W.-Y. Ma. Web-page classification through summarization. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 242–249, New York, NY, USA, 2004. ACM Press.

[13] C. Staff. How did I find that? Automatically constructing queries from bookmarked web pages and categories. In *CSAW'06: Proceedings of the third Computer Science Annual Workshop.*, to appear.

A Brief Comparison of Real-Time Software Design Methods

Tony Spiteri Staines
Dept. of Computer Information Systems
University of Malta
staines@cis.um.edu.mt

ABSTRACT
This paper briefly attempts to compare several mainstream methods/methodologies that are used for the analysis and design of real time systems. These are i) CORE, ii) YSM, iii) MASCOT, iv) CODARTS, v) HOOD, vi) ROOM, vii) UML, viii) UML-RT. Methods i-iii are use a data driven approach, whilst methods iv-vii use an object-oriented approach. All these methods have their advantages and disadvantages. Thus it is difficult to decide which method is best suited to a particular real-time design situation. Some methods like YSM, MASCOT and CODARTS are more oriented towards designing event driven systems and reactive behavior. Object oriented methods like the UML have many diagrams obtained from other methods. In the first part of the paper each method is briefly presented and its main features are explained. In the second part a score based ranking is used to try to identify which method has the best overall characteristics for real time development. The final results are presented in a tabular form and using a bar chart. In addition to this it is explained how each method fits in the SDLC. Both the score of each method and how it fits in the SDLC must be considered when selecting methods. To conclude some other issues are explained, because the selection of one method does not automatically imply that there will not be any problems.

Categories and Subject Descriptors
D.2.1 [**Software Engineering**]: Requirements/ Specifications – *Elicitation methods, Languages, Methodologies, Tools.*

General Terms
Design, Measurement

Keywords
Real Time, software design methods, methodologies, systems modeling, comparison of methods, data driven approach, object-oriented approach, software development lifecycle, RT – real time

1. INTRODUCTION
Real Time systems are more complex than traditional systems. Some real time systems demonstrate reactive behavior. They have timing, communication and reliability requirements that are not easily accounted for [10]. RT development involves not only software engineering but also hardware engineering, control engineering and communication engineering. Minor changes to specifications could be very costly. Software programs embedded directly into hardware controller and devices might require entire rewriting. There are hardware configuration problems when software engineers do not understand why specific processors, devices and operating systems are needed. Specific methodologies have been developed for helping the analysis and design of RT systems. These were different from normal methods because they had to focus on behavior and not just static system properties. Methodologies do not guarantee solving all software development problems although they attempt to structure the analysis & development of RT systems applying design techniques and rules. RT methodologies make use of the basic concepts from structured analysis and design [3,4]. First methodologies like MASCOT, JSD, DARTS were data-driven based on principles from traditional structured analysis and design [4,7]. Later work was to use object oriented notations. These offer several advantages over the former methodologies like reuse and a neater approach [2,9]. The OMG boosted the use of object oriented methods for analysis and design through the UML and UML-RT. Early versions of object oriented modeling lacked the dynamic aspects of behavior and focused mainly on static and structural aspects which were insufficient for real time. Later models tried to combine object oriented notations with state charts, activity diagrams, interaction diagrams or message sequence charts now known as sequence diagrams in the UML.

Some authors compare software design methods with software modeling notations. This is incorrect because a proper design method should encompass the entire software development lifecycle process. This is not the case with the UML where the focus is on modeling a system rather than on managing the entire software development. The OMG created USDP (Unified Software Development Process) based on the UML notations. The USDP is a proper methodology. Even the COMET (Concurrent Object Modeling architectural design method) in [8] combines the UML notations within a special lifecycle model. On the other hand software notations are more generic and focus on particular aspects of the design process. In system specifications there could be the use of models or system views. These could be singular or multiple, formal, semi-formal or informal, graphical or language based [1]. Good specifications using UML constructs could be used to derive specifications in the HDL (hardware description

language or ESDL (embedded systems description language) as in [6].

2. OVERVIEW OF SOME METHODS

2.1 Controlled Requirements Expression

The CORE (controlled requirements expression) method [4,5,12] is based on block diagrams, viewpoint diagrams and written statements of requirements. It was designed in the UK for the requirements analysis phase and was widely used in avionics. It is suitable for the informal process of gathering the systems requirements that are expressed using informal notations like block diagrams, viewpoint diagrams etc. It is very simple to use and understand not involving any complex notations etc. This approach is mainly data driven at a very high level but still could be used in conjunction with object oriented analysis. There is top-down decomposition into smaller parts. The CORE makes use of viewpoint diagrams, thread/ dataflow diagrams and block diagram notations. Control loops are available for use in the final diagram. The idea of view points could prove to be important to other methods and also for situations where requirements specification proves to be difficult. The CORE method tries to take a practical approach to problem identification at the earliest possible stage. The diagrams used are quite simple and some form of support could be obtained using modern case tools. The results that are produced using this method can be used as the input for another method. Some limitations are that: i) There is no reference to timing, concurrency, synchronization, ii) Unsuitable for Architectural design iii) No simulation model is produced.

2.2 Yourdon Structured Method

The YSM (Yourdon structured method) in [4] is based on the classic DFDs and structured methods used for traditional data design. It has been adapted and combined with many diagrams for RT design. It has been developed and refined over the years and many modern CASE tools can be used to support the notation. YSM starts off from a high-level and decomposes the system into lower levels ending up with complete program specifications. Two embedded design Methodologies have been derived from YSM. These are Ward-Mellor, Hatley-Pirbhai. This method can being used in conjunction with diagrams like PEM(Processor Environment Model) which is a hardware based design to help decide on the hardware configuration. There is also the SEM (Software-Environment Model). There are many different data driven methods that make use of the principles in YSM and add other diagrams. The PEM model and SEM are important because as pointed out RT systems are highly dependant on the available hardware which is normally ignored. YSM also uses DFDs, STDs, E-R diagrams, textual specifications, structure charts etc for design purposes. DFDs can be combined with STDs to represent both continuous and discrete actions. The behavioral model consists of DFDs, STDs & ERDs together with textual support describing the requirements but having no implementation details. The PEM covers the physical processors deciding which processor should do which work and HCI details. The COM involves translating the SEM units into structure charts and refining them so that this can be translated into program code. One advantage is that YSM is a highly structured data analysis method. Some limitations are i) it is unsuitable for prototyping. ii) it must be followed in logical sequence or sequential ordering for successful implementation iii) It is possible to take a long time to implement the complete system iv) user must have familiarity with certain constructs. v) there is specific reference to timing issues, concurrency etc although the diagrams can be altered to support time.

2.3 Modular Approach to Software Construction Operation and Test

MASCOT (Modular approach to software construction operation and test) was first issued in 1970s by the Royal Signals and Radar Establishment UK and successive versions MASCOT 3 exist [11]. It is mainly used for avionics and in the military field. It is a highly modular rigorous approach based on hierarchical decomposition to lower levels. MASCOT is based on processes or activities in a system and aims at designing complex interactive RT applications in a highly structured approach. Mascot focuses on communication between different components and enforces that a specification must be complete at every level. Interfacing between modules is extremely well represented, thus even concurrency and synchronization can be dealt with. The main steps are i) Describe the overall internal functions of the system, together with its external connections. This is known as the Network Diagram. ii) The network is decomposed into lower-level components iii) Define the structure of single thread processes (transform). iv) Design and code simple components in terms of algorithms and data structures. There are the following rules i) processes cannot send data directly to other processes ii) communication between different components can only take place through channels or windows. iii) Intercommunication Data Areas (IDAs) must be used for data exchange, information storage and communication. Some limitations of Mascot are i) it does not directly support requirements analysis and goes directly into building a model ii) it is not widely supported via many case tools iii) it is not suitable for prototyping or RAD iv) it is expensive to apply.

2.4 Concurrent Design Approach for RT Systems

CODARTS (Concurrent design approach for RT systems) is a modified form of DARTS (Design approach for RT systems) [7]. CODARTS implements concepts from DARTS for an object-oriented perspective. ADARTS was mainly aimed for use with the ADA language. CODARTS uses notations from RTSAD (Real-Time structured analysis and design). The diagrams used in CODARTS are similar to control flow diagrams that use special symbols for different types of message passing e.g. loosely-coupled message communication, tightly-coupled message. Possible diagrams are task architecture diagrams, software architecture diagrams and STDs. CODARTS classifies message passing into several types not normally found in other methods. These are supposed to be easily implemented in ADA. Some limitations of CODARTS are i) Designed mainly for the ADA language. ii) Notations used are not well understood iii) Message communication even though well identified still does not account for concurrency, synchronization, mutual exclusion. iii) uses a limited number of views.

2.5 Hierarchical Object Oriented Design

HOOD (Hierarchical Object Oriented Design) method covers some parts of the software development lifecycle. It is mainly aimed at the ADA language taking an object oriented approach. It can be useful for prototyping. The idea behind HOOD is to

identify objects in a parent to child relationship and their operations. Finally a graphical system description is to be produced in a control/ dataflow diagram that shows the flow of information between a set of objects. The diagrams can be decomposed to the required levels. The Top-Level Object is an active object because it uses the lower-level ones but not vice-versa. Rules distinguish passive Objects from active Objects. Certain flows are not permitted like cyclic flows. Limitations of HOOD are : i) does not distinguish Data Flows between Objects from Event Signals ii) Not so simple and straightforward to use iii) Has just one main diagrammatic type thus just one model structure is given.

2.6 Real time Object Oriented Modeling

ROOM (Real time object oriented modeling) is similar to HOOD in principle but is more oriented to RT design and focuses on proper communication between objects. ROOM introduces the concept of Actors in 'ROOMcharts' which are a variation of StateCharts (ROOMcharts define the state of an actor, signals that initiate transitions between states, and actions performed at transitions. There is a strong focus on this actor behavior. The actor is anything that initiates a sequence of events. There is the use of 'ports' for information exchange and threads that can have a priority. Some limitations of ROOM are : i) Closely Tied with one particular CASE tool called 'ObjecTime' which can generate C++ code ii) It has a limited number of diagrams that show only certain views of the system i.e. actor view. iii) Its diagrams need to be supported with temporal analysis for complex systems.

2.7 The Unified Modeling Language

The UML can be considered to be a repository of notations existing in methods like ROOM, HOOD, YSM, MASCOT, etc. The name 'unified' implies a unification of modeling constructs. E.g. UML state diagrams are simplified STDs, communication diagrams are found elsewhere as interaction diagrams, sequence diagrams are derived from MSC (Message sequence charts). It contains notations that are lacking in other methodologies and tries to standardize them and it is set to improve upon previous notations. It is well supported by a variety of CASE tools when compared to other methods and can be used by anyone without formal knowledge. The main system views can be categorized into i) static ii) behavioral. The UML is not a proper software development method and can be combined with almost any development method. Diagrams and notations used are Informal. It is possible to use the OCL (Object Constraint Language) to formalize the diagrams used. When a class uses operations by a second class a control flow is set up. The UML does not distinguish between the spatial distribution of objects and the logical object distribution. Code generation can be done from some UML diagrams like a class diagram. There are projects like the ECLIPSE open source tool that supports many UML constructs. There is a lack of standardization amongst the UML CASE tools and UML versions giving rise to confusion about which notations should be used. Some CASE tools providers have created their own notations that differ from those in the UML. Some limitations of the UML are : i) studies show that maintaining UML diagrams can become a complex process ii) UML lacks formal verification iii) the same thing can be modeled in several different ways, all could be correct. So there is a lack of consistency.

2.8 The UML-RT

UML-RT is based on extensions to the UML specifically aimed at RT. The most important 'new' notations are mainly capsules, ports, connectors and protocols. UML-RT implements some ideas from HOOD, ROOM and MASCOT adding them to the normal UML notations. E.g. the idea of capsule diagrams embedding child objects is similar to HOOD Parent-Child object relationships. The idea of active and passive ports already exists in ROOM. The idea of using capsules to model complex objects that are usually spatially distributed is similar to that of MASCOT where components / devices are connected using windows, ports and IDAs. Some limitations of UML-RT are i) not widely used and supported. UML-RT includes all the modeling capabilities of ROOM.

3. PRACTICAL ASSESSMENT OF THE METHODS

These methods were measured on the attributes in table 1.and 2. The final classification results are in table 3. i) Consistency between notations refers to the consistency between the diagrams used. The more notations there are the more difficult it becomes to keep consistency. ii) Support for communication constructs includes support for concurrency, synchronization, mutual exclusion, signaling, communication control, the use of ports and abstraction. iii) Support for resource control refers to the handling of different system components with processing loops and activity management, possibly used for performance management. iv) Support for temporal requirements indicates the need to show the different states the system or components can be in. Other issues like CASE tool support, abstraction and also ease of use were also considered.

Table 1. Method Comparison 1

METHOD	CONSISTENCY BETWEEN NOTATIONS	SUPPORT FOR COMUNICATION CONSTRUCTS	SUPPORT FOR RESOURCE CONTROL	DIFFERENT SYSTEM VIEWS
CORE	Very Good	Poor	Poor	Average
YSM	Very Good	Poor	Poor	Average
MASCOT	Very Good	Excellent	Very Good	Poor
CODARTS	Very Good	Good	Good	Average
HOOD	Very Good	Average	Good	Poor
ROOM	Very Good	Good	Very Good	Poor
UML	Poor	Average	Average	Very Good
UML-RT	Good	Good	Good	Good

score method (poor = 1, average = 2 , good = 3, very good =4, excellent = 5

Table 2. Method Comparison 2

METHOD	CASE TOOL SUPPORT	ABSTRACTION / INFO. HIDING & COMPOSITION	SUPPORT TEMPORAL REQUIREMENTS	EASE OF USE
CORE	Very Good	Average	Poor	Good
YSM	Very Good	Average	Poor	Good
MASCOT	Poor	Very Good	Average	Poor
CODARTS	Good	Average	Average	Average
HOOD	Good	Average	Average	Average
ROOM	Good	Good	Very Good	Average
UML	Excellent	Good	Average	Very Good
UML-RT	Average	Good	Good	Good

score method (poor = 1, average = 2 , good = 3, very good =4, excellent = 5)

Table 3. Final Method Score

Method	Ranking Score
ROOM	24
UML	23
UML-RT	23
MASCOT	22
CODARTS	21
HOOD	19
CORE	18
YSM	17

Fig. 1 below depicts the final results for the methods/methodologies commonly used for real time software development. The results are obtained from the data in table 3.

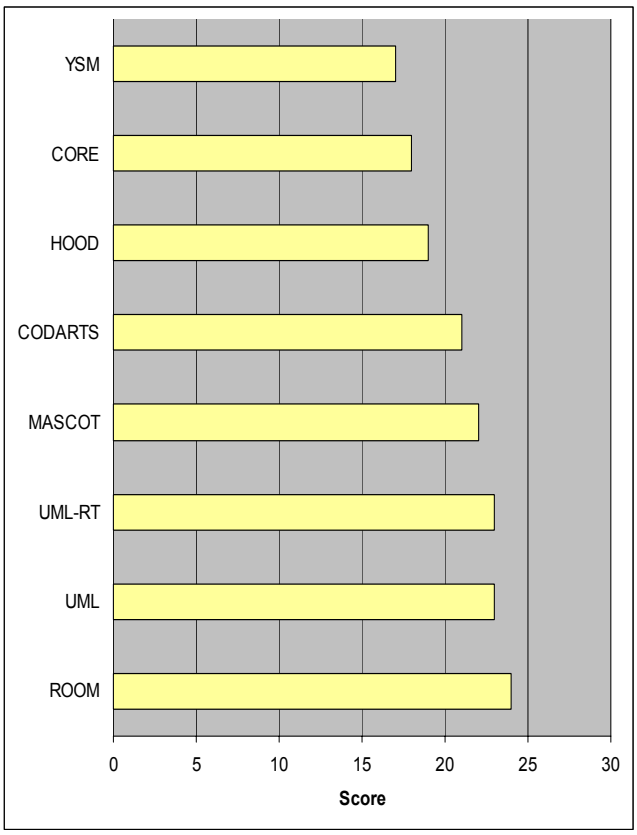

Figure 1. Real Time Method Ranking Bar Chart

Fig. 2 depicts how each method would actually fit in the SDLC (systems development lifecycle) process. The main steps included are requirements analysis, requirements specification, logical design, physical design, coding and testing. Obviously coding would imply integrating the components through interfaces etc. This is based on my own observations with reference to [3,4,7,8,9,12] and also practical use of some of these methods.

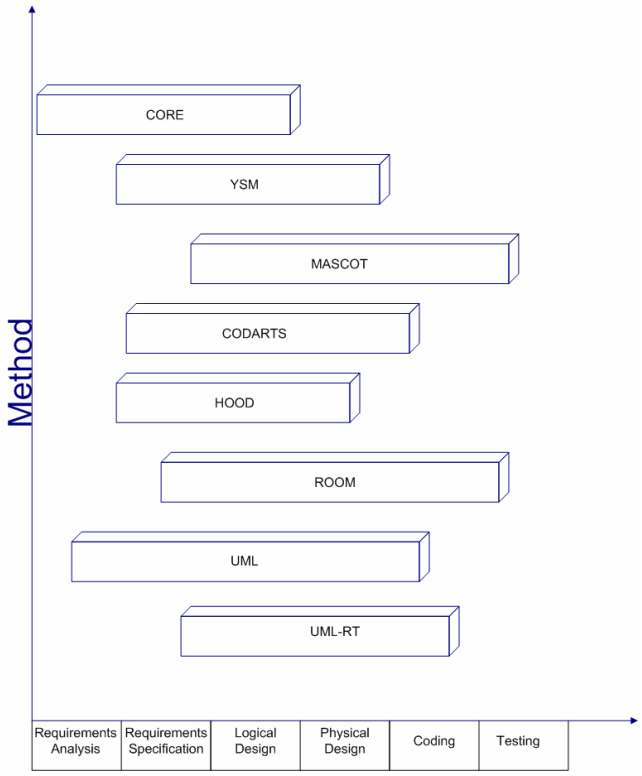

Figure 2. Methods vs SDLC phases

4. DISCUSSION

The results of this comparison in fig. 1 show that ROOM, UML and UML-RT rank as the three best methods for the development of RT systems. UML has the advantage of gaining widespread use and a lot of work is being done to improve UML continuously. ROOM and UML-RT whilst being suitable for describing complex RT systems, unfortunately lack widespread support of many CASE tools and require time to master. Another advantage of UML is that some UML diagrams are applied in a MDA approach and used to create PIM [6]. It is not justifiable that only one particular method is used. E.g. other methods like MASCOT embody principles that are still valid today and have been implemented in part in ROOM. UML does not have proper control flow diagrams similar to those found in YSM and CODARTS. These are important for designing command and control and embedded system tasks. UML instead uses activity diagrams or communication diagrams. Activity diagrams are more adequate for business analysis, communication diagrams lack some detail and need modification on the other hand control flow diagrams are oriented to task management, reactive behavior and control. This could indicate that UML is more oriented towards building soft- real time systems like those used in e-commerce , agent architectures, workflow systems, etc. On the other hand CODARTS and YSM would be more suitable for things like avionics, a cruise control description etc. Another problem with the UML is that there are so many notations that it is often difficult to select what is really needed. E.g. Sequence diagrams and communication diagrams are semantically equivalent. When should one use one rather than the other? Also there are several ways in UML how to represent the same thing. Thus it is possible

to have different diagrams of the same type representing the same scenario. In methods like the UML, ROOM, HOOD the messaging topology between objects is often 'loosely defined' with the possibility of having confusion.

It is obvious that what is lacking in one method might exist in another method. Object oriented methods are not a complete guarantee that there will be reusable components that will be available at a cheaper price especially if the interface needs to be rewritten. RT systems depend heavily on available hardware and might be operating system specific. This would imply that the design pattern is already biased from the onset of the project.

All the methods mentioned do not use proper formal verification techniques. Formal verification could be very important for checking that a design is free from deadlock. A lot of work is being done to try to formalize the UML like in [13]. There are also issues of performance analysis and task scheduling that need to be accounted for. CODARTS notations have already been used for performance analysis and task scheduling. The UML lacks performance analysis and does not take time into account. Actually the timing problem for many methods can be partially solved by translating dynamic UML diagrams into timed Petri Nets or using timed automata. The UML has been criticized by various authors. Note that even though ROOM is bound to a particular case tool its diagrams can easily be supported with other conventional case tools thus it has good case tool support.

The diagram in fig. 2 simply describes how each method fits in the systems development lifecycle process. These issues need to be considered when using these methods. If the focus is more on requirements engineering CORE could prove to be better than the others. CORE is the most adequate for requirements analysis and specification. HOOD, CODARTS and YSM cover part of the requirements analysis up to coding. MASCOT is more oriented towards the design, implementation phase and testing. The UML can be used for requirements analysis and can cover a wide aspect of the systems development lifecycle but it needs to be used in COMET or the USDP process. It could also be possible to combine CORE with UML. ROOM can cover up to testing depending on how it is used but it is not focused on the initial requirements analysis. What is evident is that no method covers all the required steps. This illustrates that for RT development one never be restricted to using a single method.

5. CONCLUSION

This paper has compared several methods for the analysis and design of real time systems. Although ROOM , UML, UML-RT stand out clearly as being the best on a number of attributes, in real life it is better not to be restricted to a single method. E.g. when students are using a method like the UML for their APT s it is always suggested that other notations from another method can be used. This is especially the case if there is a problem that requires some explanation. If using another notation or diagram would help then why not use it. A synonymous approach could be considered for industrial use. There are also other specific factors that need to be considered when selecting a method, like i) the type of industry involved, ii) user specialization, iii) if formal verification is required iv) reliability and safeness. It must be kept in mind that the results established in this paper are based on the set of attributes in table 1 & 2 might not be fully agreed upon by everybody.

6. REFERENCES

[1] Bennet, D. *Designing Hard Software*. Manning, 1996.

[2] Booch, G. *Object-Oriented Design with Applications*, 2d ed. Reading, Mass.: Addison-Wesley, 1991.

[3] Burns, A., Wellings, *A.. Real-Time Systems and Programming Languages*.Addison-Wesley, 2001.

[4] Cooling, J. *Software Design for Real-Time Systems*. Chapman & Hall, U.K.,1995.

[5] CORE, Controlled Requirements Expression (1986). *System Designers plc*, Fleet Hampshire, GU13 8 PD, UK document no.1986/0786/500/PR/0518.

[6] Coyle, F.P., Thornton, M.A. From UML to HDL: a Model Driven Architectural Approach to Hardware-Software Co-Design, *Information Systems: New Generations Conference (ISNG)*, Apr 2005, pp. 88-93.

[7] Gomaa, H. *Software Design Methods for Concurrent and Real-Time Systems*.Addison-Wesley, USA, 1996.

[8] Gomaa, H. *Designing Concurrent, Distributed and Real-Time Applications with UML*. Addison-Wesley, USA, 2004

[9] Graham, I. *Object Oriented Methods*. Addison-Wesley, USA 2000.

[10] Liu, J.W.S. *Real-Time Systems*, Pretence Hall, 2000.

[11] MASCOT, *The Official Handbook of MASCOT*. Joint IECCA and MUF Committee, 1987.

[12] Mullery, G.P., CORE - a method for controlled requirement specification. *ACM International Conference on Software Engineering Proceedings of the 4th international conference on Software engineering*, Munich Germany 1979 , pp.126 – 135.

[13] Saldhana, J.A., Shatz, S.M., Hu, Z.Formalization of Object Behavior and Interaction From UML Models. *International Journal of Software & Knowledge Engineering*, 11(6), 2001, pp. 643-673.

Automatic Interface Generation for Enumerative Model Checking

Computer Science Annual Workshop 2006

Sandro Spina
Dept. of Computer Science and A.I.
New Computing Building
University of Malta, Malta
sandro.spina@um.edu.mt

Gordon Pace
Dept. of Computer Science and A.I.
New Computing Building
University of Malta, Malta
gordon.pace@um.edu.mt

ABSTRACT

Explicit state model checking techniques suffer from the state explosion problem [7]. Interfaces [6, 2] can provide a partial solution to this problem by means of compositional state space reduction and can thus be applied when verifying interestingly large examples. Interface generation has till now been largely a manual process, were experts in the system or protocol to be verified describe the interface. This can lead to errors appearing in the verification process unless overheads to check the correctness of the interface are carried out. We address this issue by looking at automatic generation of interfaces, which by the very nature of their construction can be guaranteed to be correct. This report outlines preliminary experiments carried out on automatic techniques for interface generation together with their proofs of correctness.

1. INTRODUCTION

Computer systems (both software and hardware) have over the past few decades been introduced into almost every piece of machinery. Real-time systems such as controllers for avionics, cars and medical equipment have become ubiquitous. Model checking techniques are used to algorithmically verify these finite state systems formally. It is becoming increasingly popular by many hardware/software manufacturers to verify that their systems actually implement the required specifications. This is achieved by verifying if the model of the system satisfies some logical specification. Suppose we want to verify, for example, that every request for service is eventually acknowledged, or that there are no deadlock states in our systems. This sort of verification can be carried out using model checking techniques.

CSAW '06 CSAI Department, University of Malta

Processes can be described using some formal process calculi such as CCS [8], CSP [3] or LOTOS [4]. Properties are then expressed as temporal logic formulas. Computational Tree Logic (CTL) [1] is one such temporal logic used to express properties of a system in the context of formal verification. It uses atomic propositions as its building blocks to make statements about the states of a system. CTL then combines these propositions into formulas using logical and temporal operators. Referring to the previous example, one would for example want to verify that every computation path after a service request is met, will eventually encounter a service acknowledgment state.

The state space explosion problem occurs with systems composed of a large number of interacting components using data structures which can potentially store many different values. The problem is clearly that of traversing the entire search space which would typically grow exponentially with the addition of new system components. One possible solution is that of decreasing the number of states in the computational graph while still maintaining an equivalent graph. In order to do so one would need to combine equivalent states (thus decreasing states) in the computational graph. We adopt the technique used by Krimm and Mounier in [6], namely *interfaces*.

We start this report with some preliminary definitions. We then focus on the theory behind our method of interface generation. Two interface generators are then explained in some detail. We conclude this report by describing how these implementations work in the verification of a reliable multicast protocol.

2. PRELIMINARY DEFINITIONS

The behaviour of a sequential process can be modeled by a labeled transition system, consisting of a set of states and a labeled transition relation between states. Each transition describes the execution of the process from a current state given a particular instruction (label).

In what follows $\mathcal{A}$ is the global set of labels, τ a particular label representing a hidden or unobservable instruction ($\tau \notin$

$\mathcal{A}$). Given a set of labels A ($A \subseteq \mathcal{A}$) we will write A_τ to denote $A \cup \{\tau\}$ and A^* to represent the set of finite sequences over A.

Definition 1 *A Labeled Transition System (LTS, for short) is a quadruplet $M = (Q, A, T, q_0)$ where Q is a finite set of states, $A \subseteq \mathcal{A}$ is a finite set of actions, $T \subseteq Q \times A_\tau \times Q$ is a transition relation between states in Q and $q_0 \in Q$ is the initial state.*

We write $q \xrightarrow{a} q'$ to denote a transition between states q and q' using $a \in A$, i.e. $(q, a, q') \in T$. We shall also use $q \stackrel{s}{\Rightarrow} q_n$ (where s is a string in A_τ^*) to indicate that there exist states $q_1 \dots q_n$ following string s.

We now define the set of possible actions from a state $q \in Q$.

Definition 2 *Given an LTS $M = (Q, A, T, q_0)$ and $q \in Q$, the actions possible from state q is defined as actions$(q) = \{a : A \mid \exists q' \cdot q \xrightarrow{a} q'\}$.*

We now define the language generated by a LTS from a particular state $p \in Q$.

Definition 3 *Given an LTS $M = (Q, A, T, q_0)$ and $q \in Q$, the (observable) language starting from q in M is defined as follows:*

$$\mathcal{L}_{\mathcal{M}}(q) = \{\sigma \mid \sigma = a_1 a_2 \dots a_n \wedge \exists q_1, \dots, q_n \cdot q \stackrel{\tau^* a_1}{\Rightarrow} q_1 \dots \stackrel{\tau^* a_n \tau^*}{\Rightarrow} q_n\}$$

The (observable) language generated by LTS M is defined as the language starting from the initial state of M: $\mathcal{L}_M(q_0)$.

3. REFINEMENTS OF LTSS

In this section we introduce the binary operator $\sqsubseteq$, that compares two LTSes.

Definition 4 *We say that M_2* is refined by *M_1 ($M_2 \sqsubseteq M_1$) if for some* total *function* $\text{eq} \in Q_1 \to Q_2$ *the following holds:*

i) $A_1 = A_2$

ii) $Q_2 \subseteq Q_1$

iii) $q \xrightarrow{a}_1 q'$, *implies that,* $eq(q) \xrightarrow{a}_2 eq(q')$

iv) $q_0 = eq(q_0)$

Lemma 1 *Given two LTS $M_i = (Q_i, A_i, T_i, q_{0i})$, where $i \in \{1, 2\}$, related with a function eq, then $q \stackrel{s}{\Rightarrow}_1 q'$ implies that $eq(q) \stackrel{s}{\Rightarrow}_2 eq(q')$*

Proof: We prove this lemma by string induction over s.

The base case, taking s to be the empty string is trivially true.

For the inductive case, we start by assuming that: $\forall q, q' \cdot q \stackrel{t}{\Rightarrow}_1 q'$ implies that $eq(q) \stackrel{t}{\Rightarrow}_2 eq(q')$.

We now need to prove that:

$$\forall q, q' \cdot q \stackrel{at}{\Rightarrow}_1 q' \text{ implies that } eq(q) \stackrel{at}{\Rightarrow}_2 eq(q')$$

But, if $q \stackrel{at}{\Rightarrow}_1 q'$ then $\exists q'' \cdot q \xrightarrow{a}_1 q'' \wedge q'' \stackrel{t}{\Rightarrow}_1 q'$.

By the inductive hypothesis, it follows that $\exists q'' \cdot q \xrightarrow{a}_1 q'' \wedge eq(q'') \stackrel{t}{\Rightarrow}_2 eq(q')$.

Since we know that every transition in M_1 is mirrored in M_2 on equivalent states, then we know that $\exists q'' \cdot eq(q) \xrightarrow{a}_2 eq(q'') \wedge eq(q'') \stackrel{t}{\Rightarrow}_2 eq(q')$.

Hence, we conclude that:

$$eq(q) \stackrel{at}{\Rightarrow}_2 eq(q')$$

The result follows by string induction. □

Theorem 1 *Given two LTSs $M_i = (Q_i, A_i, T_i, q_{0i})$, with $i \in \{1, 2\}$, if M_2* is refined by *M_1 ($M_2 \sqsubseteq M_1$), then the language generated by M_1, $\mathcal{L}(M_1)$, is a subset of the language generated by M_2, $\mathcal{L}(M_2)$.*

Proof: To require to show that if $s \in \mathcal{L}(M_1)$ then $s \in \mathcal{L}(M_2)$.

If $s \in \mathcal{L}(M_1)$, then, by definition of $\mathcal{L}(M)$:

$$\exists q_1 : Q_1 \cdot q_{01} \stackrel{s}{\Rightarrow}_1 q_1$$

By applying lemma 1, we can conclude that $eq(q_{01}) \stackrel{s}{\Rightarrow}_2 eq(q_1)$. But since, we know that $q_{02} = eq(q_{01})$, it follows that $s \in \mathcal{L}(M_2)$. □

4. INTERFACES

Interfaces [6] exploit the use of a compositional approach for state space generation. Essentially an interface represents the envorinment of a sub-expression E' in E. This interface, usually a LTS, represents the set of authorised execution sequences that can be performed by E' within the context of E. Using a projector operator one can generate a restricted LTS of E' such that useless execution sequences are cut off according to the corresponding interface.

In [6] a new projection operator is defined. This is the *semi-composition* operator which ensures that :

1. it restricts the behaviour of E' according to its environment

2. it preserves the behaviour of the initial expression when a sub-expression E' is replaced by its corresponing reduced expression.

3. it can be computed on-the-fly, i.e. can be obtained without generating the LTS of E' first.

The definition of the semi-composition operator is given in [6]. $M_1 \rceil\!\rceil_G M_2$ denotes the LTS resulting from the semi-composition of M_1 by M_2. M_2 is the interface with which M_1 is semi-composed. One should note that if M_2 is manually generated by an expert of the system or protocol being verified then semi-composition is probably going to be much more effective in reducing the states of S_1. Our work explores the possibility of *automatically* creating effective interfaces. We can then guarantee their correctness by construction.

If M_1 is composed with M_2 (not necessarily locally) and M_2' is an LTS such that $\mathcal{L}(M_2) \subseteq \mathcal{L}(M_2')$, then we can reduce M_1 to $M_1 \rceil\!\rceil_G M_2'$ without altering the overall behaviour of the overall system. We would clearly be altering the behaviour of M_1 but this is exactly what we want in terms of state reduction. Since the complexity of the semi-composition operator increases as the the number of states of the interface increases, sometimes being impossible to calculate due to the size of the interface, our aim is to balance these requirements — taking an LTS M we want to produce a smaller LTS M' satisfying $\mathcal{L}(M) \subseteq \mathcal{L}(M')$. Clearly, many solutions exist satisfying this loose requirement. In this paper we present two initial experiments in this direction.

5. AUTOMATIC GENERATION OF INTERFACES

We have so far implemented two interface generators. This section describes these algorithms. In what follows we refer to the original LTS as M_1 and its reduced LTS, the interface, as M_2

5.1 Chaos State Partition

The first interface implementation is the chaos state partition interface. The main idea is that we keep a number of states from M_1, collapsing the rest into a *chaos state*. So for example if we have M_1 with 20 states and we want to generate an interface taking in consideration only the first 10 states (traversing the LTS in breadth-first order, starting from the inital state), its reduced version M_2 would have 10+1 states. The extra state is the chaos state (we call χ) in which all the other states are grouped. Figure 1 illustrates this reduction.

Let $\bar{Q}$ be a subset of states of M_1 which will be kept in M_2. M_2 is the LTS resulting from the reduction of M_1. $\bar{Q} = \{0,1,2,3,4,5,6,7,8,9\}$ in the example given here.

Definition 5 *Given an LTS* $M = (Q, A, T, q_0)$ *and* $\bar{Q} \subseteq Q$, *we define the reduction of* M *to states* $\bar{Q}$ *to be* $M \rhd \bar{Q} = (Q', A', T', q_0')$ *such that* $Q' = \bar{Q} \cup \{chaos\}$, $A' = A$, $q_0' = \text{chaos}_\chi^{\bar{Q}}(q_0)$ *and* $T' = \{(\text{chaos}_\chi^{\bar{Q}}(q), a, \text{chaos}_\chi^{\bar{Q}}(q')) \mid (q, a, q') \in T_1\} \cup \{(\chi, a, \chi) \mid a \in A\}$, *where* chaos *is defined as follows:*

$$\text{chaos}_\chi^Q(q) = \begin{cases} q & \text{if } q \in Q \\ \chi & \text{otherwise} \end{cases}$$

This construction yields an LTS M' which is refined by the original LTS M.

Proposition 1 $M \rhd \bar{Q} \sqsubseteq M$

This can be shown by taking $eq(q)$ to be $chaos_\chi^{\bar{Q}}(q)$.

By theorem 1 we are guaranteed that $M \rhd \bar{Q}$ accepts a superset of the language accepted by M, and hence can be used as a replacement interface.

The Chaos Partition algorithm has been implemented using the CADP toolkit [5] traversing the LTS in breadth-first order, starting from the inital state.

Algorithm 1 Calculate $M_2 = M_1 \rhd \bar{Q}$

Require: m ≤ n
  n ← number of states in M_1
  m ← depth in breath first order of last state in M_1 to keep before chaos
  for $i = 0$ to n **do**
    if i < m **then**
      copy state Q_i from M_1 to M_2
      copy outgoing tranisitions of state Q_i from M_1 to M_2
    else
      join state Q_i to the chaos state in M_2
      copy outgoing transitions of Q_i in M_1 to M_2
    end if
  end for

5.2 Node Behaviour State Partition

Our second implementation abandons the idea of creating a chaos state and instead moves in the direction of creating a set partition which groups together states in M_1 which can perform exactly the same set of strings of length n. With, for example, the length being 1, and there two states in M_1 which can only perform actions a and b then these two states are grouped together in one state in M_2. We currently cater only for length 1 but will deal with the general case in future work. Figure 2 illustrates the same LTS shown earlier on but this time reduced with the Node Behaviour state partition.

The state partitions created with this reduction are the following:

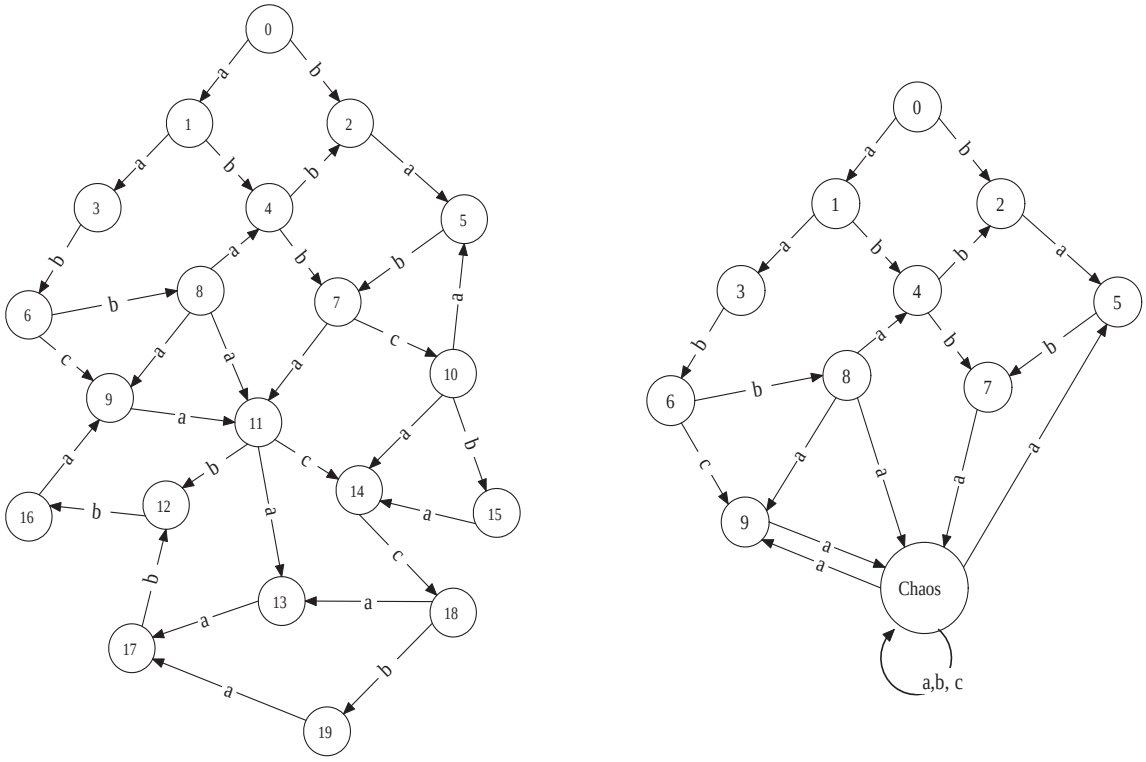

Figure 1: State reduction of a 20 state LTS using the chaos state partition

State Actions	State Partition
a,b	{0,1,10,18}
a	{2,8,9,13,15,16,19}
b	{3,4,5,12,17}
b,c	{6}
a,c	{7}
a,b,c	{11}
c	{14}

The partitions created will form the new states in M_2.

Definition 6 *Given an LTS* $M = (Q, A, T, q_0)$*, the reduction of* M *looking at trails of length 1 is defined to be* $M^{[1]} = (Q', A', T', q'_0)$ *where* $Q' = 2^A$*,* $A' = A$*,* $q'_0 = actions(q_0)$ *and* $T' = \{actions(q), a, actions(q')) \mid (q, a, q') \in T\}$.

This construction yields an LTS M' which is refined by the original LTS M.

Proposition 2 $M^{[1]} \sqsubseteq M$

With depth = 1 the possible states of M_2 is equal to the powerset of A. Hence function eq is defined as follows:

$$eq(q) = actions(q)$$

Interface generation using the Node Behaviour partition algorithm has also been implemented within the CADP toolkit [5]. As explained in algorithm 2 we first go through all the states in M_1 and group them in state partitions according to their outgoing transitions. These state partitions become the new states in M_2. We then create the transitions between the new state partitions to reflect the transitions present in M_1. The number of states and number of transitions in M_2 is clearly always smaller or equal to that in M_1.

6. CASE STUDIES OF COMPOSITIONAL VERIFICATION

In this section we describe some experiments carried out with the automatically generated interfaces. In order to come up with a good comparison of how these interfaces work we make use of an example listed as demo of the CADP toolkit which uses the *projector* operator discussed earlier.

We first give a very brief introduction to the CADP toolkit, then describe the rel/REL multicast protocol. We illustrate the results achieved when the sub-expressions of the protocol are restricted with our automatically generated interfaces.

6.1 Construction and Analysis of Distributed Processes (CADP)

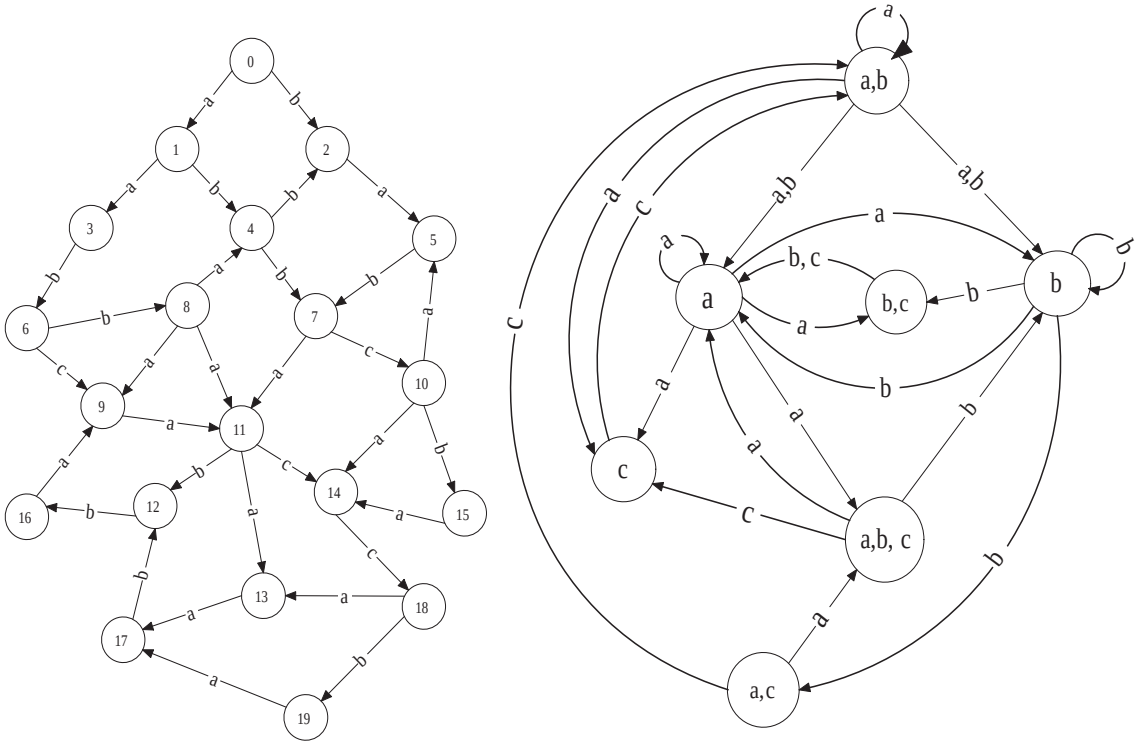

Figure 2: State reduction of a 20 state LTS when n=1

Algorithm 2 Calculate $M_2 = M_1^{[1]}$

```
n ← number of states in M1
for i = 0 to n do
  if state partition already exists which maps the behaviour of Qi then
    add Qi from M1 to this state partition
  else
    create new state partition
    add Qi from M1 to the newly created state partition
  end if
end for
X,Y ← set of state partitions in M2
for all partitions Xj in X do
  for all partitions Yk in Y do
    if there are states in Xj and Yk such that these states were connected in M1 then
      connect with the same transition as in M1 partitions Xj and Yk
    end if
  end for
end for
```

The CADP toolkit [5] consists of a set of verification tools developed by the VASY research group[1]. The toolkit is especially useful in the design of communication protocols and distributed systems. Processes are specified in LOTOS [4] which is essentially a language for specifying CCS and CSP like processes. CADP contains amongst other tools, compilers for translating LOTOS descriptions into C code and LTSs which is then used for simulation, verification and testing purposes. The implementation of our interfaces was carried out using the CADP tools for explicit state manipulation, namely the BCG. Binary Coded Graphs (BCG) is a package which uses highly optimised representation of LTSs together with various other tools for the manipulation of these LTSs. Further information on CADP can be obtained by visiting the VASY website.

The *projector* operator within CADP implements the semi-composition operator defined in [6]. This operator is used to constrain processes within their environment. We make use this semi-composition operator to contrain processes with our automatically generated interfaces before actually applying the parallel composition on them.

6.2 The rel/REL Multicast Protocol

The rel/REL protocol [9] provides a reliable atomic multicast service between stations connected by a network. The

[1]http://www.inrialpes.fr/vasy/cadp/

LOTOS specification of the multicast protocol can be found in [6]. The rel/REL multicast service should ensure that a message sent by a transmitter is appropriately broadcasted to all receivers taking in consideration the fact that these stations may suffer from failures (i.e., they can crash).

In this paper we do not give details of how the rel/REL protocol works. We are only interested in the service provided by the protocol which need to be verified. The first property is atomicity, meaning that either all receivers get the message, or none of them will. This property is verified using temporal logic. The second property is that of causality meaning that the order of messages is preserved. This property is verified using comparison modulo safety equivalences. By using our automatically generated interfaces we would like to observe a decrease in the number of states in the intermediate LTSs composing the whole protocol.

The example considered here is composed of one transmitting station (Trans) and three receiver stations (Rec). Their parallel composition denoted by expression E (without using interfaces) is as follows:

$$((Rec2||_A Rec3)||_B Rec1)||_C Trans$$

$$\begin{array}{rcl} \text{where A} & = & \{R23, R32\} \\ \text{B} & = & \{R12, R13, R21, R31\} \\ \text{C} & = & \{RT1, RT2, RT3\} \end{array}$$

In [6] manually generated interfaces $\mathcal{I}$ are used to restrict the receiver nodes. The transmitter LTS is then used to further restrict the composed receiver stations. Figure 6.2 shows how the projector operator is used when generating the LTS of the whole system. Note that the symbol $-||$ in figure 6.2 refers to the projector opertor.

In order to calculate the decrease in number of states imposed by our interfaces we need to split up expression E in various sub-expressions. [6] splits expression E listed above in the following intermediate LTSs each of which corresponding to a generation step. Here we add $S0$ as a further intermediate step to analyse.

$$\begin{array}{lcl} S1_i & = & sem(Rec_i) \rceil\!\lfloor \mathcal{I}_i \\ S0 & = & S1_2 \|_{\{R23,R32\}} S1_3 \\ S2 & = & S0 \rceil\!\lfloor_{\{RT2,RT3\}} sem(Tx) \\ S3 & = & (S1_1 \|_{\{R12,R21,R13,R31\}} S2) \ \rceil\!\lfloor_{\{RT_n\}} \ sem(Tx) \\ sem(E) & = & S3 \|_{\{RT_n\}} sem(Tx) \end{array}$$

The following table illustrates the various reductions obtained when using our interfaces. Each row indicates the number of states of the subexpression together with the number of states remaining when the subexpression is reduced up to strong bisimulation [8].

The first column shows the reductions when manually generated interfaces are used. The reductions obtained with this

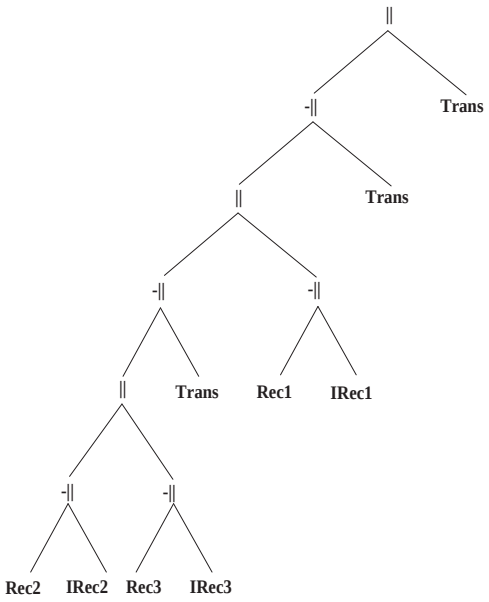

Figure 3: Computational tree of the rel/REL protocol using interfaces

method are clearly the best possible. The second column shows the sizes of the intermediate LTSs when no interfaces are used. We could only generated $S0$ in this case since the composition of $S2$ with $S1_i$ could not be generated due to the exponential growth of the composed LTSs. The third and fourth column illustrate the reductions obtained when *BFReduction* (Chaos State Partition implementation) and *TEReduction* (Node Behaviour n=1) were used.

The first row is indicative of the reductions obtained by the interfaces. When using a manually generated interface (user written) the receiver node is reduced to 1086 states. The original size of the receive node LTS is 2882 as indicated in the second column. Both BFReduction and TEReduction exploit the Transmitter LTS which is eventually composed with the Receiver nodes. An interface is generated out of this LTS and immediately applied to the Receiver node.

Both interfaces obtain a reduction in state space of the receiver node. When the receiver LTS is projected with the interface created by the BFReduction algorithm the receiver node is reduced to 2106 states. With the TEReduction algorithm we only manage to reduce the receiver LTS to 2700. It is interesting to note however that when both LTSs are reduced up to strong bisimulation both interface reduced receiver nodes get much smaller. Further work needs to be carried out in order to understand why this is the case.

The composition of two receiver nodes yields an LTS of 3,789,678 states when no interfaces are used. When using our interfaces we manage to get a reduction to 1,621,640 states with the BFReduction interface and a reduction to

Expression	Manual Interfaces	No Interfaces	BFReduction	TEReduction
$S1_2$	1086/369	2882/127	2106/149	2700/127
$S0$	229767/31020	3789768/15139	1621640/20939	3313694/15139
$S2$	149366/30171	na	1621640/20939	3313694/15139

3,313,694 states with the TEReduction interface. The composition if the receiver nodes on which the manual interface was used goes up only to 229,676 states. This clearly shows the difference a manually generated interface has on the composition of these LTS.

With these interfaces we can only generate up to $S2$. Without interfaces not even $S2$ could be generated. We are currently investigating some of the results obtained with these interfaces, whilst trying to improve on their implementations. The two implementations which we curently have serve only as proof of concept for our work. We clearly need to come up with more 'intelligent' interface generators.

7. FUTURE RESEARCH

This paper discusses only our initial attempts at interface generation. Through these interface we want to reduce the intermediate state space needed for the verification of systems. So far we have implemented two interface generators which only give us some indications on the way forward. The current interfaces have some pitfalls which we shall be addressing shortly. For example, TEReduction with a depth of 1, groups together in one state all those states which are able to perform only the internal action τ. This can be partially avoided simply by increasing the depth to some other value greater than 1.

Further experiments also need to be carried out in order to better asses the behaviour of the current interfaces. We shall be working on the verification of various other protocols using interfaces. Ultimately we would like to obtain a general purpose interface generator which is able to effectively reduce the number of intermediate states necessary.

8. REFERENCES

[1] E.M.Clarke, O.Grumberg, and D.A.Peled. *Model Checking*. The MIT Press, Cambridge, Massachusettes, 1999.

[2] Frederic Lang. Refined interfaces for compositional verification. In E. Brinksma, editor, *International Conference on Formal Techniques for Networked and Distributed Systems FORTE'2006*, pages 239–258, Enschede, The Netherlands, 2006. Springer Verlag, LNCS 1217.

[3] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, August 1978.

[4] ISO/IEC. LOTOS. A formal description technique based on the temporal ordering of observational behaviour. international standard 8807, international organisation for standardization - information processing systems. *Open Systems Interconnection*, 1989.

[5] J. -C. Fernandez, H. Garavel, A. Kerbrat, L. Mounier, R. Mateescu, and M. Sighireanu. CADP: a protocol validation and verification toolbox. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the Eighth International Conference on Computer Aided Verification CAV*, volume 1102, pages 437–440, New Brunswick, NJ, USA, / 1996. Springer Verlag.

[6] J.P. Krimm and L. Mounier. Compositional state space generation from Lotos programs. In E. Brinksma, editor, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 239–258, Enschede, The Netherlands, 1997. Springer Verlag, LNCS 1217.

[7] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic Model Checking: 10^{20} States and Beyond. In *Proceedings of the Fifth Annual IEEE Symposium on Logic in Computer Science*, pages 1–33, Washington, D.C., 1990. IEEE Computer Society Press.

[8] R. Milner. *A Calculus of Communicating Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[9] P D Ezhilchelvan and S K Shrivastava. rel/rel: a family of reliable multicast protocols for distributed systems. *Distributed Systems Engineering*, 1(5):323–331, 1994.

EUMEDGRID

Grid Computing in Malta and the Mediterranean

Kevin Vella
University of Malta, Malta

Robert Sultana
University of Malta, Malta

Reggie Cushing
University of Malta, Malta

Federico Ruggieri
INFN Rome, Italy

Roberto Barbera
Catania University/INFN Catania, Italy

Ognjen Prnjat
GRNET, Greece

Federica Tanlongo
GARR, Italy

Giuseppe Andronico
Catania University/INFN Catania, Italy

Fotis Karayannis
GRNET, Greece

ABSTRACT

In this paper we introduce EUMEDGRID, an EU-funded project with the objectives of building the first high performance computing grid extending across the Mediterranean, and fostering national grid initiatives in the Mediterranean region. Grid computing is an emerging technology which has generated significant global interest due to its potential to usher in a new era in computing: the provision of computing as a service on demand, akin to a utility such as the electricity grid. Grid computing aims to provide transparent access to computing and storage facilities distributed over a wide area. The European Commission sees grid computing as a crucial component in improving collaboration across the European Research Area. It is branded a key technology in the European Information Society 2010 initiative, whose aim is to maximize economic growth and social inclusion, thus making the EU "the most competitive, knowledge-based economy in the world by the year 2010" [29]. A brief overview of grid computing, applications, standards and architecture opens the paper. This is followed by an account of EUMEDGRID's objectives and activities, and a discussion of Malta's participation in grid initiatives. Finally, we consider the future of grids in Malta and the Mediterranean.

Keywords

Parallel and Cluster Computing, Grid Computing, Networking, Collaborative Systems, e-Infrastructures, e-Science, Human Factors, Standardisation.

1. INTRODUCTION

EUMEDGRID [9] is an initiative funded through the European Commission's 6th Framework programme which aims to build the first high performance computing grid extending across the Mediterranean region. EUMEDGRID is currently in its pilot phase, and will eventually form part of EGEE, the European e-Science grid [6].

EUMEDGRID kicked off in 2006 with a meeting in Malta [28] and to date has secured EU funding to operate until December 2007. The project is coordinated by INFN (Italian National Institute for Nuclear Physics), and the project partners besides the University of Malta are

- GRNET (Greek Research and Technology Network),
- CERN (The European Organisation for Nuclear Research),
- DANTE (Delivery of Advanced Network Technology to Europe, UK),
- Consortium GARR (Italian Academic and Research Network),
- CYNET (Cyprus Research and Academic Network),
- RED.ES (Entitad Publica Empresarial, Spain),
- CERIST (Centre De Recherche sur l'Information Scientifique et Technique, Algeria),
- CNRST (Centre National Pour la Recherche Scientifique et Technique, Morocco),
- EUN (Egyptian Universities Network),
- HIAST (Higher Institute of Applied Sciences and Technology, Syria),
- MSRTDC (Ministry of Scientific Research, Technology and Competency Development, Tunisia),
- TUBITAK (Scientific and Technological Research Council, Turkey).

Additionally, a number of third parties from Greece, Italy, Israel (IUCC), Jordan (JUNET), Palestine (PADI2), Tunisia, and Turkey are participating.

2. GRID COMPUTING

Grid computing is an emerging technology which has generated significant global attention. This is because grids have the potential to usher in a new era in computing: the age of computing and storage resources on demand, somewhat similar to a utility model such as the electricity grid. From a researcher's viewpoint, grid computing enables remote interactive access to

specialised and exclusive scientific equipment, large scale computing facilities and repositories of experimental data, as well as collaboration with geographically dispersed research groups. This can revolutionise the way research is conducted and empower researchers in remote areas with limited facilities.

The European Commission sees grid computing as a crucial component in improving collaboration across the European Research Area. It is branded a key technology in the European Information Society 2010 initiative, whose aim is to maximize economic growth and social inclusion, thus making the EU "the most competitive, knowledge-based economy in the world by the year 2010" [29].

2.1 Grid Architecture

The concept of the Grid is driven by the need for "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organisations" [16]. This includes direct access to computers, software, data and other resources. Clear constraints on what resources may be shared and who can utilise them are defined by dynamic virtual organisations constituted by multiple individuals or institutions.

Grid technologies complement existing distributed technologies, rather than replacing them, by extending the distribution across organisational boundaries. Since interoperability is key to sharing, The Grid architecture must be open standards-based, defining protocols through which virtual organisation members and resources flexibly and dynamically negotiate, manage and utilise sharing relationships. These protocols govern interaction between components rather than their implementation, hence existing structures within institutions may be preserved.

The service abstraction is central to grid operations, and a number of standard services are defined to provide computation, data, resource discovery and other functionality that virtual organisation members may provide and utilise, while abstracting away the internal heterogeneity within specific resources. It is worth noting that web services are being utilised for this purpose in next generation Grid implementations.

Finally, programming interfaces must be built on top of grid services to enable developers to create applications for this dynamic environment.

The Grid architecture as specified in [16] contains the following layers, moving upwards:

- The **Fabric Layer** exposes the local resources which will be shared across the Grid. In general, more sophisticated actions on existing resources, such as advance reservation, will involve more work in the fabric layer in order to extend the resources' existing (possibly limited) functionality. Enquiry (to establish current state, capabilities and structure) and management mechanisms are required at the very least. The types of resources exposed by the fabric layer may include computational resources, storage resources, network resources and databases.
- The **Connectivity Layer** provides communication and authentication protocols to enable the controlled sharing of resources in the fabric layer. Standard Internet protocols for networking and security are generally used. Authentication for virtual organisations should enable single sign-on, delegation of permissions (the ability of a program to perform actions on a user's behalf), user-based trust (combined use of resources from multiple locations without further administrator intervention, provided usage permissions for the individual resources are set), and integration with existing local security mechanisms.
- The **Resource Layer** builds on the connectivity layer to provide sharing of individual resources. This layer defines protocols and programming interfaces for negotiation, initiation, monitoring, control and accounting for the sharing of a resource. The resource layer also utilises the local fabric layer to provide shared access to local resources. Resource layer protocols may be classified into information and management protocols. The problem of synchronisation between multiple resources is left to the following layer.
- The **Collective Layer** deals with the coordination of multiple resources by providing protocols, services and programming interfaces for directories, resource allocation, scheduling and brokering, monitoring and diagnostics, data replication, workload management and collaboration, software discovery and accounting, amongst others. Some collective layer services may be specific to a particular virtual organisation. Persistence of state is a key issue in this layer.
- The **Application Layer** consists of the applications that execute within a virtual organisation, and make use of services at any underlying layer, which are exposed through well-defined protocols.

2.2 Grid Applications

While the most widely publicised applications for grids to date are in high energy physics (HEP) and bioinformatics, any application requiring substantial computational power, storage resources, and/or cooperation between users across geographically dispersed locations is a valid candidate.

In this section we will focus on a selection of applications running on the European e-Science Grid (EGEE and EGEE-II [6]), due to the close association of EUMEDGRID with these activities. The applications mentioned here are treated in further detail in [7].

- **High Energy Physics** was an initial pilot area for grid applications. One of the main tasks of European research grids will be to handle the processing of vast amounts of data that will be generated by the Large Hadron Collider experiments (ATLAS, CMS, ALICE, LHCb) at CERN. Other HEP experiments, such as BaBar, CDF, DØ, H1 and ZEUS are currently making use of the EGEE infrastructure.
- **Biomedical Applications.** The biomedical community was also involved in initial grid pilots from the outset. At this stage, several applications are running on the EGEE infrastructure. Amongst these are GPS@ and WISDOM. GPS@ is a bioinformatics portal providing protein sequence similarity searches, sites and signatures detection, multiple alignment, secondary structure prediction and primary structure analysis. The WISDOM Drug Discovery application aims to speed up the process of finding new drugs against malaria, H5N1 and other diseases.
- **Astrophysics Applications.** The European Space Agency is utilising EGEE e-infrastructure to simulate the forthcoming Planck satellite mission and test the data pipelines, thus

providing input to the mission's hardware requirements. The EGEE e-infrastructure is also processing data from MAGIC, an imaging atmospheric telescope located on the Canary Islands that is used for astro-particle physics research.

- **Earth Science and Geophysics Applications.** The Grid is being used to analyse ozone profiles from the GOME satellite and oil spill data from the ERS/SAR satellite, facilitating data sharing within the earth observation community. Earthquake simulations using 3D geological models, and Montecarlo simulations for seawater intrusion in a coastal aquifer of the Mediterranean basin are also being carried out. In addition, a flood forecasting application has been ported for grid execution. Finally, GeoCluster, a seismic processing solution, is the first industrial application running on the EGEE production service.
- **Computational Chemistry.** A number of applications have been deployed under the computational chemistry virtual organisation. The GEMS and DI-Poly applications are running on EGEE to simulate the reaction dynamics of complex chemical systems. RWAVEP computes chemical reactive quantum probabilities. COLOMBUS performs *ab initio* molecular electronic structure calculations, while GAMESS calculates SCF wave functions for molecular quantum chemistry.
- Other areas such as **Financial and Economic Research**, **Digital Libraries**, **Fusion Research**.

3. THE EUMEDGRID PROJECT

3.1 European and Mediterranean Scenarios

Recent developments in European Research Infrastructures have led to the creation of a pan-EU high-speed research network with full administrative and operational support. This network has paved the way for the development of an overlying European Grid infrastructure in order to enable the sharing of distributed computing power and storage across geographical and administrative domains. This e-infrastructure provides a platform for new methods of global collaborative research: e-science. The leading initiatives in European e-infrastructures are the GÉANT /GÉANT2 [18][19] and EGEE/EGEE-II [6] projects.

The e-Infrastructure is bound to broaden in the next few years, as a result of a number of initiatives taken in this area between Europe and developing countries all around the world. Implementing a Grid-enabled infrastructure would be of strategic relevance to foster collaboration between researchers from Europe and other countries and to contribute towards bridging the digital divide among areas with different levels of technological development.

The EUMEDCONNECT project [8] has succeeded in establishing a Mediterranean research network (see Figure 1). At this very moment, several non-EU Mediterranean countries are taking the first steps towards adopting Grid computing and a number of cooperative research projects exist thanks to the collaboration between scientists from the Mediterranean and Europe. These projects would strongly benefit from the availability of a stable regional Grid-enabled infrastructure.

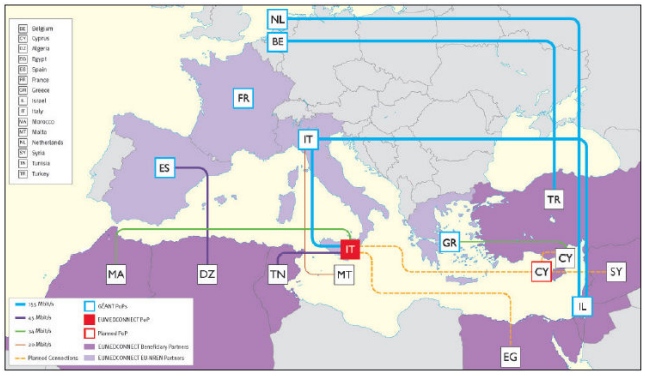

Figure 1. The EUMEDCONNECT network, a basis for EUMEDGRID

3.2 The Objectives of EUMEDGRID

The objective of the EUMEDGRID project is to bring the less experienced and less resourced countries of the Mediterranean region to the level of European developments with regards to e-infrastructures [26]. With the Mediterranean networking infrastructure reaching stability through the EUMEDCONNECT project, the focus of the EUMEDGRID will be on Grid infrastructure and related e-science applications. The objectives of EUMEDGRID are focused on two main areas:

- creating a human network in e-science, assessing its needs and transferring Grid expertise to the constituent communities; and
- driving the technological developments that will enable the commencement of Grid operations in the region.

A broad range of activities will focus on dissemination and outreach, hands-on workshops and close collaboration with related projects such as EUMEDCONNECT, GÉANT, EGEE and SEE-GRID [27]. Furthermore, a pilot Grid infrastructure will be created, including Grid cluster deployment and the development and enforcement of operational and organisational schemes. Finally, a range of regional e-science applications will be investigated, with candidate applications deployed on the regional infrastructure.

This approach will help bridge the digital divide between Mediterranean countries and the EU and will facilitate the participation of these countries in European and worldwide e-infrastructures and e-science activities, thus expanding and supporting the European Research Area (ERA).

EUMEDGRID will pave the way towards the provision of a transparent and ubiquitous common market of computing resources to the various research communities in the region. Integration of EUMEDGRID into the wider European e-infrastructure will provide the research and education community of the region with access to a much larger pool of storage and computing resources than currently available.

EUMEDGRID will utilise the networks built through the EUMEDCONNECT and GÉANT projects, as well as through strong liaisons with the EGEE project and the experience of other regional initiatives such as SEE-GRID.

We now provide a detailed list of specific objectives for EUMEDGRID as specified in the project proposal's technical

annex. The first set of objectives focus on soft actions, with the overall aim of creating a human network in the area of Grids, e-science and e-infrastructures in the Mediterranean, and promoting regional and international cooperation:

O1 Stimulate the formation of national Grid infrastructures (NGIs) in the Mediterranean countries, thus contributing to the creation of a "virtual Grid-based research space".

O2 Promote awareness in the region regarding Grid developments through the organization of a number of dissemination and outreach events, which will promote the project results to the private and public sector, ultimately reaching the general public.

O3 Establish a dialogue regarding policy developments for research and education networking and provide input to the agenda of national funding bodies and if possible governments.

The second set of objectives focus the technical aspects and are intended to support, on the basis of an in-deep analysis of local requirements, the implementation of a pilot Grid infrastructure across the Mediterranean and the deployment of a set of test applications on it:

O4 Capture local e-science user requirements in terms of resources needed, Grid services, and application software.

O5 Provide guidelines and technical cookbooks to guide regional integration in the Euro-Mediterranean infrastructures

O6 Support the establishment of pilot Grid resource centres at each country in the region, and adapt and implement operational and organisational management procedures, bringing the region up to speed with production-level operations. Pilot Grid resource centres are intended to be the major vehicle of this process, becoming the seeds of national Grid infrastructures in the Mediterranean countries.

O7 Build upon and exploit the network infrastructure provided by GÉANT and EUMEDCONNECT in the region. The amalgamation of national grid initiatives into a Mediterranean infrastructure will take advantage of the existing human network created within the EUMEDCONNECT project. It is envisaged that this human network will trigger the creation of a physical backbone connecting directly all EUMEDGRID actors, to maximize the effectiveness of the pilot grid infrastructure.

O8 As a proof of principle, support the deployment of EGEE applications (high energy physics, biomed) and other Grid applications of regional interest on the pilot infrastructure, with the involvement of local user community. New user communities will be actively encouraged to join the EUMEDGRID community and deploy their own applications on the pilot infrastructure.

3.3 Expected Impact of EUMEDGRID

This section outlines the expected strategic impact of the EUMEDGRID project, focusing on standards and policy.

3.3.1 Standards

One of the key objectives of EUMEDGRID is to enable the implementation of a pilot Grid interoperable with wider European (EGEE) and worldwide Grids, thus ensuring a smooth integration on the infrastructure, core middleware and service levels. This will in turn enable resource and service access and sharing between the Mediterranean region and rest of Europe and the world, allowing the local user communities to access resources and collaborate with partners across Europe and the world.

In this context, an inherent activity within EUMEDGRID will be active use of the middleware developed by EGEE, its validation and certification, and feedback regarding its functionality and performance, based on its use. In this way, EUMEDGRID will actively participate in requirements capture and feedback on the middleware, standards and protocols, but also Grid management tools. Via this feedback through EGEE, EUMEDGRID will indirectly contribute to international standards bodies such as the Open Grid Forum (OGF) [25].

3.3.2 Policy

The Barcelona Euro-Mediterranean Conference of 27-28 November 1995 [14] stressed that support for the development of the Mediterranean scientific and technological community, together with the upgrading and modernisation of local telecommunications infrastructure, are two pivotal elements for the success of the Euro-Mediterranean partnership.

The EC has since approved initiatives for the development of the EUro-MEDiterranean Information Society (EUMEDIS) specifically designed to reduce the region's informational and technological gap as compared to the neighbouring countries. The Programme is also complementary to a regional telecom regulatory framework project, New Approaches to Telecom Policy, launched around the same time. In coming years it is foreseen that all current financial assistance instruments will be integrated into the framework of the new European Neighbourhood Policy.

The European Neighbourhood Policy (ENP) [15] is an overall policy aiming to share the benefits of the EU's enlargement with neighbouring countries in strengthening stability, security and welfare, in order to prevent the emergence of new dividing lines between the enlarged EU and its neighbours. Amongst the ENP's key issues are the integration of infrastructures, and scientific and cultural cooperation.

The EUMEDGRID project will contribute to policy developments via a number of its activities:

- Enhance collaboration between scientific communities, both within the region and with the rest of Europe and the world, thus paving the way for educational, scientific and cultural exchanges, initiatives and projects across borders.
- Stimulate the sharing of the know-how in terms of Grid computing within the region and between the region and the rest of Europe and the world.
- Stimulate the use of the regional networking infrastructure (GÉANT, EUMEDCONNECT) and trigger the connection of the Mediterranean national research and education networks with each other in order to create a powerful regional backbone.
- Become a catalyst for the development of a common computing and storage infrastructure, thus paving the way for a unified regional e-Infrastructure.

- Facilitate research activities over a number of scientific sectors, by providing access to the nascent e-Infrastructures to a wide spectrum of applications.
- Encourage the interaction of research communities and governmental bodies in the region with the aim of gaining long-term support for e-infrastructures and e-science.
- Foster dialogue on common policies among governments by means of a targeted communication strategy, including the organization of a dedicated Policy workshop.
- Take advantage of the experience of the e-Infrastructure Reflection Group [4], involve in it actors from Mediterranean and propose its model at a regional level. EUMEDGRID aims to trigger these actions by involving the countries of the region in latest developments and e-infrastructure practices in networking, computing and e-science. e-Europe aims to accelerate the development of the information society in Europe and ensure its availability to all citizens. In this context, the impact of the project will hopefully be beyond the immediate scientific community, building towards an all-inclusive knowledge society.

The project is expected to improve the following knowledge-based society indicators along three distinct lines:

1. Providing cheaper, faster, secure and immediate access to resources;
2. Investing in people and skills;
3. Exploiting the network infrastructures provided by GÉANT and EUMEDCONNECT.

These are seen as directly linked to economic development of the region.

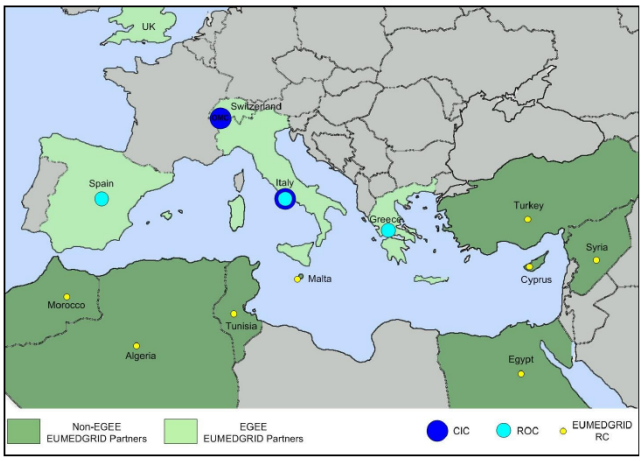

Figure 2. The strategic impact of EUMEDGRID on the Mediterranean region

Figure 2 depicts the strategic impact of EUMEDGRID on the Mediterranean region. A distinction is made between European EGEE countries and the less-resourced Mediterranean countries: the EGEE countries already operate Grid management structures such as Regional Operations Centres (ROCs) and Core Infrastructure Centres (CICs), which will help launch and operate the Grid Resource Centres (RCs) in the Mediterranean countries thus creating a unified Grid infrastructure across the region.

3.4 EUMEDGRID Activities

There are four support activities within EUMEDGRID, dealing with requirements capture and analysis (work package 2), Grid middleware and establishment and operations of the regional pilot Grid infrastructure (work package 3), supporting applications deployment (work package 4), and outreach and dissemination (work package 5). Project administration and technical management is encapsulated a separate work package. We will provide a brief overview of the activities within each work package, as outlined in the project proposal [26].

3.4.1 Requirements capture and analysis (WP2)

This work package, under the management of the University of Malta, is responsible for the gathering and analysis of participant information through various sources, and the formulation of a strategy and technological roadmap for the implementation of the EUMEDGRID initiatives on the basis of this analysis.

A questionnaire was designed to establish the status in each of the participant countries as regards grid technology awareness, current use of computing or grid facilities, current and future grid application requirements, and existing hardware infrastructure (computing power, storage capacity, connectivity). A system for the online submission of questionnaire responses was developed and deployed [12]. In total, ninety-six questionnaire responses were gathered from researchers and institutions in eleven participant countries and analysed, though we are still accepting responses at this stage. Furthermore, the current state of play in the IT and telecommunications areas including research and academic network initiatives in each of the participant countries was established through various sources. The information collected was analysed and a detailed report on the findings presented in a project deliverable document [10], which has provided EUMEDGRID with a clear view of the state of the current Mediterranean grid computing, IT and telecommunications landscape.

The second activity of this work package, coordinated by CERN, utilised as its principal input the above-mentioned results to formulate a strategy and a technical roadmap for EUMEDGRID. A model for the promulgation of Grid technologies to the Mediterranean region was established to guide the formation of sustainable National Grid Initiatives and the development of regional infrastructure for integration purposes. Technical solutions were proposed, taking into consideration the middleware and hardware infrastructure requirements. Additionally, a regional deployment strategy starting from site installation to operations was developed, giving due consideration to the regional perspective in terms of financial, political, technical, managerial, and time-scale concerns. The experience gained in other European projects for the promotion of regional Grid initiatives, such as SEE-GRID, was a primary guiding force in this activity. The strategy and technical roadmap was also documented in a project deliverable [11], and serves as a guide for the infrastructure and operations activities in work package 3.

This work package was formally completed in summer 2006, though the questionnaire is still open for further responses from interested researchers and institutions.

3.4.2 Infrastructure and operations (WP3)

Taking as input the regional requirements and technical roadmap produced by work package 2, this work package, coordinated by

GRNET, is currently supporting the successful deployment and operation of the regional pilot Grid infrastructure. This involves the selection and adaptation of middleware solutions produced by the leading Grid projects, with special focus on the latest EGEE middleware based on web services.

The aim during the first year of the project is to ensure the establishment of one stable pilot cluster in each beneficiary country, and to achieve interoperability of the pilot infrastructure. This task is already in an advanced state of completion, with clusters forming part of EUMEDGRID in various Mediterranean countries including Malta. A catch-all Certification Authority (CA) for the region, to issue user and machine certificates to entities in the countries which do not have an established CA, has been established. In the meantime, related CA and Registration Authority guidelines are being established. By the end of the first project year, all the participant countries should have gained some hands-on experience with Grid middleware and clusters.

As the regional teams gain experience, studies will be carried out to determine the operational requirements for the pilot infrastructure. Solutions for the operations centres will be proposed, with the special emphasis on the relationship with EGEE production-level operations and operational structures such as existing Regional Operations Centres (ROCs) and Core Infrastructure Centres (CICs). GRNET will lead this effort, benefiting from its experience in running a distributed operations centre in the EGEE and SEE-GRID projects.

Solutions will also be proposed for helpdesks and monitoring. Although the project does not intend to deliver production-level Grid services (with full manageability, robustness, resilience to failure, scalability, 24/7 production operations, etc.), all available effort will be invested to adopt as many practices as possible from EGEE. The second year of the project will focus on this aspect and it is envisaged that at the end of the project most countries will gain enough expertise and experience to be able to support full-fledged production Grid operations.

This work package aims to integrate the existing and nascent National Grid Initiatives through enabling an interoperable infrastructure. Moreover, by promoting the deployment of standard solutions, this infrastructure will aim to be interoperable with wider European and worldwide Grid services. Finally, this work package is responsible for providing network support and co-ordinating this activity with corresponding network projects such as EUMEDCONNECT and GÉANT.

3.4.3 Application selection and deployment (WP4)

Three application classes will be considered for deployment on the EUMEDGRID infrastructure under this work package which is coordinated by INFN:

- **A well-established set of Grid applications, such as the EGEE applications**, which will be used to verify the effectiveness of the pilot infrastructure. The application areas (and associated user communities) in this class are:
    - LHC (Large Hadron Collider) experiments in the field of High Energy Physics. Experiments at LHC, currently under development at CERN, will start taking data in 2007 and groups of Mediterranean scientists are actively participating in these experiments. The data collected by each experiment will be of the order of 100 Mbytes per second and will be widely available to all the collaborators in Europe and worldwide, provided that good connectivity and a compatible grid infrastructure is available.
    - Biomedicine applications, as this is the second pilot scientific field in EGEE. EUMEDGRID is currently contributing resources to the WISDOM Data Challenge [4], which is utilising Grid technology to assist in drug design.
    - Earth sciences applications, which are emerging as the third-level scientific field in EGEE. A typical application that could be deployed is SPECFEM3D, numerical simulation of earthquakes in complex three-dimensional geological models.
- **A set of applications to be chosen between the Mediterranean partners of the project as regional pilot applications**. The selected applications will be a sample of those most demanding in terms of producing and processing large quantities of data. The GILDA test bed will be used to validate and integrate candidate applications.
- **Applications intended to attract new communities**. A preliminary segmentation of the user base, obtained in collaboration with the Mediterranean partners, will identify this audience, and a number of public relations activities will be undertaken in order to contact the new potential users and facilitate their participation in the EUMEDGRID community. The intention is to support to the communities contacted as part of the dissemination by illustrating the benefits of Grid computing and providing them with an opportunity to use the Grid infrastructure for their research activities.

An applications questionnaire is currently being hosted at the University of Malta [13], and potential grid users in the Mediterranean region are encouraged to submit their responses in order to indicate their Grid application requirements.

3.4.4 Dissemination and outreach (WP5)

Dissemination and outreach activities are a crucial component of EUMEDGRID's strategy to introduce state of the art grid technologies to users from a broad range of disciplines in the Mediterranean region. The aims of this work package, coordinated by GARR, are to:

- disseminate knowledge to communities who have already expressed an interest in grids through the requirements analysis questionnaire or other means.
- disseminate information about the benefits of grid technology to a wider audience, in order to attract new communities which have not yet evaluated this technology. This includes not just researchers, but parties who could potentially support further development in this field, such as governments, funding bodies and private industry.
- create a human network in the Mediterranean to spread both Grid awareness and know-how among researchers.

The tools at EUMEDGRID's disposal to achieve these aims include the project web site, press releases to general and specialist media, bulletins, and events organised for general and specialist audiences. A number of events have already been organised, such as the February 2006 kick-off meeting and

information event in Malta, the September 2006 conference in Rome and various Grid computing tutorials across the Mediterranean. Over the past few months EUMEDGRID has been the focus of a number of media reports in the region. Additionally, through this work package, EUMEDGRID is liaising with a number of other projects such as EGEE and SEE-GRID and ensuring that information on EUMEDGRID is disseminated at related events.

Training events are being hosted at a steady rate in a number of participant countries, including Greece, Turkey and Morocco. Training sessions ensure that users understand the characteristics of the offered grid services and that they have enough technical knowledge to properly use the infrastructure. Besides the basic induction training, two other complementary training services will be offered, covering the needs of advanced users who require deeper technical knowledge, and regular updates to keep the users informed about new services and functionalities.

Specialised training is also provided to persons involved in the technical aspects of the project. These include developers, virtual organisation managers, resource centre administrators, resource centre security managers and other support personnel. The INFN GILDA grid computing infrastructure for dissemination and training [17] will be an enabler and a key tool for the transmission of advanced knowledge and will allow new users to gain first-hand, on-the-job experience of a real grid infrastructure, including typical services and applications.

4. GRID COMPUTING IN MALTA

Malta, through the University's Computing Services Centre, has been involved in e-Infrastructure initiatives such EUMEDCONNECT and GÉANT for several years, with the associated benefit of substantially improved international Internet connectivity for academics and researchers.

The need for a Mediterranean Grid project was discussed amongst EUMEDCONNECT participants in late 2004, and an editorial board consisting of members from Italy, Greece and Malta was appointed to pen a proposal for EU funding. The bid was successful, and EUMEDGRID commenced in January 2006 with INFN, Italy leading the project and Malta's participation coordinated by the University's Computing Services Centre in collaboration with the Department of Computer Science and Artificial Intelligence.

The project kick-off meeting was hosted by Malta in February 2006 [28]. A public information session on Grid computing was organised, at which the Maltese Minister for Education, the Hon. Louis Galea, addressed the audience. A number of interviews with the intention of further raising awareness about Grid computing were aired on Maltese television stations during 2006.

Malta, as overall coordinator of the requirements gathering and analysis work package which ran through project months one to seven, was responsible for devising a Grids requirements questionnaire, deploying an online system for the submission of responses, and liaising with the Mediterranean partners to ensure that the call for questionnaire responses reached the intended target audience in the region. By April 2006, ninety-six responses were received from eleven countries. On the basis of these responses, and further information gathered from the partners and other sources regarding the IT and telecommunications infrastructure in the Mediterranean countries, a requirements analysis document for EUMEDGRID was produced at the University of Malta and submitted to the European Commission in July 2006 [10]. This document served as an input to the technical roadmap prepared by CERN within the same work package [11].

Progress has been also made on the deployment of the Maltese Grid e-Infrastructure. The Department of Computer Science and Artificial Intelligence's compute cluster, hosted in the Computing Building, has been a part of the EUMEDGRID e-Infrastructure since summer 2006, and jobs are being successfully submitted from other countries.

The current effort is focused on identifying Maltese grid computing users and applications. Maltese researchers and academics are being solicited to provide EUMEDGRID with information on their possible applications for Grid technology. We are looking for individuals or teams who:

- would benefit from accessing the existing EGEE applications (High Energy Physics, Biomed, Earth Sciences), or other existing Grid applications that could be made available on EUMEDGRID;
- currently use parallel applications running on clusters (using MPI [23], PVM [1], etc.) and would like to 'gridify' them for vastly improved execution speed and capacity for larger data sets;
- require large amounts of compute and storage resources to solve a research problem, but do not have experience with parallel or Grid computing.

A principal target for Malta by the end of the first EUMEDGRID project is to have a number of local users making use of the newly created e-infrastructure to solve research problems on a pilot basis. This will hopefully lead to the Grid service being offered as a standard computing service to academic and research institutions in Malta in the coming years.

A goal of EUMEDGRID is to stimulate the formation of locally funded National Grid Initiatives. In the Maltese context, this is a long term activity which commences with educating the Maltese public, researchers and policy makers about the Grid philosophy that is considered so crucial in the European context.

5. CONCLUSION

This paper has provided its reader with a brief overview of Grid computing and of the EUMEDGRID project. The objectives of EUMEDGRID were discussed, along with an analysis of the project's strategic impact, and the principal actions that are being carried out within EUMEDGRID. Furthermore, an overview of Malta's involvement in Grid initiatives was provided.

The EUMEDGRID initiative hopes to influence the Mediterranean region from the social as well as the technological perspective. While the Grid deployment process is a necessary component of the EUMEDGRID approach, its value would be severely diluted without a corresponding effort to stimulate the creation and strengthening of human networks in e-science, a thriving user community of researchers spanning the Mediterranean, and indeed, the entire world. The nascent e-infrastructure is purely an enabler for this vision.

EUMEDGRID will formally close at the end of 2007. With the project activities well on track, it is the partners' intention to seek

further funding for the continuation of the project, while developing plans to ensure its long-term sustainability.

6. REFERENCES

[1] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V. *Parallel Virtual Machine: a users' guide and tutorial for networked parallel computing.* MIT Press, 1994.

[2] Jones, B. *The EGEE project: a technical overview*. 1st EGEE Conference, Cork, Ireland, April 2004.

[3] West, D. *Seamless research networking for the Mediterranean region – the EUMEDCONNECT initiative*. TERENA Networking Conference 2004, Rhodes, Greece, June 2004.

[4] e-Infrastructure Reflection Group, Austrian Chairmanship. *White paper*. June 2006.

[5] EGEE Press Release. *EGEE battles Malaria with Grid WISDOM.* August 2005.

[6] EGEE-II Project Web Page. *http://www.eu-egee.org*

[7] EGEE User and Application Portal (NA4). *http://egeena4.lal.in2p3.fr*

[8] EUMEDCONNECT Project Web Page. *http://www.eumedconnect.net*

[9] EUMEDGRID Project Web Page. *http://www.eumedgrid.org*

[10] EUMEDGRID Deliverable. *User and e-infrastructure requirements capture and analysis*. July 2006.

[11] EUMEDGRID Deliverable. *Proposed technical roadmap.* September 2006.

[12] EUMEDGRID WP2 Questionnaire Web Page. *https://secure.um.edu.mt/eumedgrid/questionnaire/wp2*

[13] EUMEDGRID WP4 Questionnaire Web Page. *https://secure.um.edu.mt/eumedgrid/questionnaire/wp4*

[14] European Commission. *Europe and the Mediterranean: towards a closer partnership – an overview of the Barcelona process in 2002*. March 2003.

[15] European Commission. *European Neighbourhood Policy: strategy paper.* May 2004.

[16] Berman, F., Fox, G., Hey, T. *Grid computing: making the global infrastructure a reality*. Wiley Publishers, 2003.

[17] Andronico, G. et al. *GILDA: the grid INFN virtual laboratory for dissemination activities*. TRIDENTCOM 2005, 1st International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, February 2005.

[18] GÉANT Project Web Page. *http://www.geant.net*

[19] GÉANT 2 Project Web Page. *http://www.geant2.net*

[20] GILDA Testbed Web Page. *http://gilda.ct.infn.it*

[21] Foster, I., Kesselman, K., Tuecke, S. *The anatomy of the Grid: enabling scalable virtual organisations*. In [16].

[22] Sanchez-Papaspiliou, J.A. *SEE-GRID: expanding the e-infrastructure inclusion into South-East Europe*. 1st EGEE Conference, Cork, Ireland, April 2004.

[23] Message Passing Interface Forum. *MPI-2: extensions to the Message Passing Interface.* November 2003.

[24] Appleton, O. *The EGEE-II project: overview paper*. June 2006.

[25] Open Grid Forum Webpage (previously the Global Grid Forum). *http://www.ogf.org*

[26] Andronico, G., Barbera, R., Karayannis, F., Prnjat, O., Ruggieri, F., Tanlongo, F., Vella, K. *The EUMEDGRID project proposal/technical annex*. December 2005.

[27] SEE-GRID and SEE-GRID-2 Project Web Page. *http://www.see-grid.eu*

[28] University of Malta. *e-Science and the EU – University of Malta hosts first EUMEDGRID project meeting*. Press Release, January 2006.

[29] Reding, V. *Pay-as-you-use computing power for all – Europe and the key role of Grid technology.* European Grid Technology Day 'Building a European Research Area for Grids and a Competitive Market Place for Grid services', Brussels, May 2005.

Runtime Validation Using Interval Temporal Logic

Karlston D'Emanuele
kema001@um.edu.mt
Dept. of Computer Science and AI
University of Malta

Gordon Pace
gordon.pace@um.edu.mt
Dept. of Computer Science and AI
University of Malta

ABSTRACT

Formal specifications are one of the design choices in reactive and/or real-time systems as a number of notations exist to formally define parts of the system. However, defining the system formally is not enough to guarantee correctness thus the specifications are used as execution monitors over the system. A number of projects are around that provides a framework to define execution monitors in Interval Temporal Logic (ITL), such as Temporal-Rover [3], EAGLE Flier [1], and D^{3}CA [2] framework.

This paper briefly describes the D^{3}CA framework, consisting in the adaptation of Quantified Discrete-Time Duration Calculus [7] to monitoring assertions. The D^{3}CA framework uses the synchronous data-flow programming language Lustre as a generic platform for defining the notation. Additionally, Lustre endows the framework with the ability to predetermine the space and time requirements of the monitoring system. After defining the notation framework the second part of the paper presents two case studies - a mine pump and an answering machine. The case studies illustrate the power endowed by using ITL observers in a reactive or event-driven system.

Categories and Subject Descriptors

[**Formal Methods**]: Validation

General Terms

Formal validation, duration calculus, QDDC, D^{3}CA , interval temporal logic.

1. INTRODUCTION

The question "Does this program do what it is supposed to do?" and "Will the program work under any environment changes?" are frequently asked when developing a software. A number of techniques have been proposed and adopted during the years, one of which is formal validation.

The major validation tools around concentrate on temporal logics as they provide a means for time measurement. The temporal logic branch of Interval Temporal Logic (ITL) provide means to measure correctness in intervals, which facilitates the scoping of tests and reduce the total impact of the monitors on the system.

We illustrate this by building a basic framework in which the user can, alongside the program, specify properties using interval temporal logic which are automatically woven as monitors into the source code, thus producing the single monitored application. The monitored application at certain intervals checks whether the application state is valid according to the mathematical model, enabling runtime validation of ITL properties.

In this paper we concentrate on showing how the framework can be used in a normal application using two simulated environments.

The rest of the paper is organised as follows: the next section briefly describes some of the validation tools around. Section 3 outlines the syntax and semantics of the interval temporal logic notation used in the paper. In section 4 we describe a framework for the Interval Temporal Logic (ITL) monitors generation and weaving. Finally we conclude by presenting two scenarios where the framework was applied.

2. VALIDATION

The size and complexity of software developed today result in debugging and testing not being sufficiently effective. Formal methods go some way towards tackling this problem. Using model-checking, one test all execution paths of a system to be checked for correctness. Nevertheless, model-checking is expensive to run and does not scale up, even when systems are reduced via abstraction. Validation is a light-weighed formal method, which checks the system correctness for a single path of execution. The path verification is performed by checking at runtime that the formal specifications weaved into the system source code constantly hold. A number of projects have been undertaken in order to find a suitable validation technique for different logics and scenarios. Some projects are Temporal Rover [3], Java-MaC [6], EAGLE [1] and RTMAssertions [8].

Temporal Rover. is a proprietary validation tool by Time-Rover. It integrates properties specified in Linear Temporal Logic (LTL) and Metric Temporal Logic (MTL) into an annotated source code [3]. The integration is performed by a pre-compiler. The formal properties are checked for consistency on every cycle, determined by a call to a method

"assign". On a false evaluation of a property an exception is raised, which the system handles in order to return to a safe state. One of the aims of Temporal Rover is to validate embedded system, which are limited in resources. Hence, the system provides another tool, DBRover, which allows properties to be checked remotely on a host PC.

Java-MaC. is a prototype implementation of a more generic architecture named Modelling and Checking (MaC). The MaC architecture consists of three modules, a filter, event recogniser and run-time checker. The properties used by the filter and event recogniser are written in Primitive Event Definition Language (PEDL), which provides a means of defining the formal specifications in relation to the implementation level. When an event (state transition) is identified by the event recogniser, the new state information is sent to the run-time checker, whose properties are written in Meta Event Definition Language (MEDL). The run-time checker in MaC is executed remotely and the information is transmitted using TCP sockets, leading to less strain on the monitored system resources.

Eagle. framework distinguishes from the other validation projects mentioned in this report. EAGLE performs validation over an execution trace and on a state-by-state basis that frees the monitoring system from using additional memory for storing the trace. The framework provides its own logic, named EAGLE, which is enough powerful to allow other logic notation to be specified in the EAGLE logic. A disadvantage in EAGLE is that the type of properties, that is whether it is a liveness or safety property, has to be explicitly specified.

RTMAssertions. uses Aspect-Oriented Programming (AOP) to separate between LTL properties and the program implementation. The LTL properties are first converted into an Abstract Syntax Tree (AST) which together with the RTM framework is weaved into the code by the aspect weaver. The LTL properties are evaluated by subdividing the LTL formula into a number of atomics, the tree leaves, which are evaluated first. Then by following the nodes path to the root, the actual formula result, is calculated recursively.

This section overviewed some of the projects in the validation field. Next is the logic notation used in the paper

3. DISCRETE AND DETERMINISTIC DURATION CALCULUS

Discrete and deterministic Duration Calculus is a descendant of Quantified Discrete-Time Duration Calculus by Pandya [7]. The logic assumes that the future is determined by the past events thus it can be easily implemented using any programming language.

3.1 Syntax and Semantics

The most basic expression in discrete and deterministic Duration Calculus are propositions, referred to as state variables. Let P be a state variable than its syntax is

$$P ::= \mathbf{false} \mid \mathbf{true} \mid p \mid P\ op\ P \mid \neg P$$

where p is a propositional variable and
$op \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow, \otimes \text{ (xor)}\}$.

On the assumption that system states have finite variability. Let σ be a non-empty sequence of state variables

$$\sigma =_{df} (\text{state variable} \mapsto \mathbb{B})^+$$

where, the length of the sequence is given by $\#(\sigma)$.

Discrete and deterministic Duration Calculus is an Interval Temporal Logic, in other words the expressions defined using this notation require an interval. Defining first the concept of time as

$$\mathbb{T} =_{df} \mathbb{N}$$

Then an interval is defined as

$$Intv =_{df} \{(b, e) \mid b, e \in \mathbb{T}\}$$

Let $\sigma_{\mathcal{I}} \models D$ mean that the finite sequence σ satisfies the duration formula D within the interval, $\mathcal{I} \in Intv$.[1].

σ_i	$\models P$	**iff**	$i \in \mathbb{T} \wedge P(i) = \mathbf{true}$
σ_i	$\models P_1 \lessdot P_2$	**iff**	$i \in \mathbb{T} \wedge \sigma_{i-1} \models P_1 \wedge \sigma_i \models P_2$
$\sigma_{\mathcal{I}}$	$\models \text{begin}(P)$	**iff**	$\sigma_{\mathcal{I}_b} \models P$
$\sigma_{\mathcal{I}}$	$\models \text{end}(P)$	**iff**	$\sigma_{\mathcal{I}_e} \models P$
$\sigma_{\mathcal{I}}$	$\models \lceil P \rceil$	**iff**	$\forall i \in \mathcal{I} \cdot i < \mathcal{I}_e \wedge \sigma_i \models P$
$\sigma_{\mathcal{I}}$	$\models \lceil\lceil P \rceil\rceil$	**iff**	$\forall i \in \mathcal{I} \cdot \sigma_i \models P$
$\sigma_{\mathcal{I}}$	$\models \eta \leqslant c$	**iff**	$(\#(\sigma) = \mathcal{I}_e - \mathcal{I}_b) \leqslant n$
$\sigma_{\mathcal{I}}$	$\models \Sigma(P) \leqslant c$	$=_{df}$	$\sum_{\mathcal{I}_b}^{\mathcal{I}_e - 1} P(i)$
$\sigma_{\mathcal{I}}$	$\models \text{age}(P) \leqslant c$	$=_{df}$	The state variable P is true for the last part of the interval and it is not constantly true for more than c time units.
$\sigma_{\mathcal{I}}$	$\models D_1 \text{ then } D_2$	**iff**	$\exists m \in \mathcal{I} \cdot \mathcal{I}_b \leqslant m \leqslant \mathcal{I}_e \wedge$ $\sigma_{[\mathcal{I}_b, m-1]} \models D_1 \wedge$ $\sigma_{[\mathcal{I}_b, m]} \models \neg D_1 \wedge$ $\sigma_{[m, \mathcal{I}_e]} \models D_2$
$\sigma_{\mathcal{I}}$	$\models D_1 \overset{\delta}{\hookleftarrow} D_2$	**iff**	The duration formula D_2 must be true for the first δ time units that formula D_1 is true.
$\sigma_{\mathcal{I}}$	$\models D^*$	$=_{df}$	$\exists n \in \mathbb{N} \cdot D_1 \text{ then } D_2 \ldots \text{ then } D_n$

The next section describes the design of a system that translates formulae written using the notation introduced into run-time monitors for .NET systems.

4. THE TOOL: D^{3}CA

D^{3}CA is a prototype implementation of a validation engine for properties defined in deterministic QDDC. The programming language used for implementation is C#.

D^{3}CA consists of two modules: the validation engine and the weaving of validation with the system. The validation engine grabs a collection of properties and using a simulated Lustre environment checks the each property with the current state. The weaving process can be performed using Aspect Oriented programming (AOP) tools. Nevertheless, for better understanding of the communication process between the validation engine and the monitored system, a weaver is discussed.

Figure 1 illustrates the architecture of a D^{3}CA monitored system.

[1]Due to lack of space, some operators are defined informally. For full formal definitions, refer to [2].

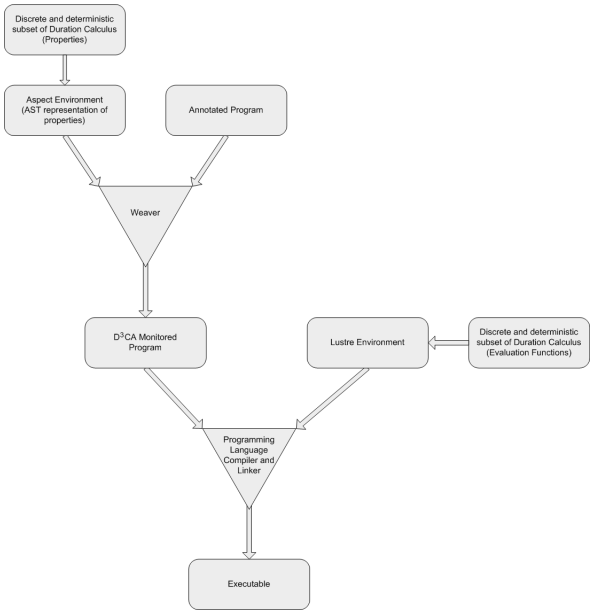

Figure 1: D^{3}CA Architecture Overview.

4.1 Lustre

Lustre is a dataflow synchronous language [4], that is it applies the synchrony hypothesis over execution of code. The hypothesis states that all computations are performed atomically and take no time to execute. Lustre implements a restricted subset of dataflow primitives and structures, that makes it applicable to reactive and real-time systems.

Lustre provides three data types: integer, real and boolean. The variables used inside Lustre must be explicitly declared and initialised to the appropriate data type. The expression variables have the form "$X = E$" where X is associated with the expression E. Therefore, E is taken to be the only possible value for X.

The Lustre operators set consists in the basic data type operators, conditional branching and two special operators, `pre` and "followed by" ($\rightarrow$). The `pre` operator enables the access to a variable history. While the "followed by" operator is used to concatenate two streams together, where at time zero the second stream is equal to the first value of the first stream.

In D^{3}CA the Lustre environment is simulated, that is, rather than using a Lustre compiler the data types are implemented as classes. Using boolean variables with deterministic QDDC leads the value of **false** to be ambiguous. The Lustre environment is extended with 3-valued logic data type, which extends the boolean data type with the additional value of **indeterminate** for expressions whose truth value has yet to be determined.

The use of a Lustre environment in D^{3}CA allows the space and memory requirements of the specified properties to be predetermined. Hence, providing a place for optimisation and a knowledge on the side-effects of the validation engine on the monitored system.

4.2 Weaver

The weaver module consists in transforming annotated control instructions to the actual validation code. The annotated control instructions can be of five types:

1. {*validation_engine*: **bind** *variable_name variable_value*}
2. {*validation_engine*: **unbind** *variable_name*}
3. {*validation_engine*: **start** *assertion_name*}
4. {*validation_engine*: **stop** *assertion_name*}
5. {*validation_engine*: **synchronise** $\mathbb{B}$}

In validation the state variables are mapped to the system variables. In software, variables are typically placed in their local scope, hence, they cannot be accessed from outside code. This raises a problem with interval temporal logic assertions that have crosscutting semantics. To surmount the problem the two variable un/binding annotations are provided. A variable can be bound to either a system variable while the variable is in scope or to a numeric value. The unbind annotation instructs the weaver that the value of the variable must be kept constant.

An important characteristic of the D^{3}CA is that it uses interval temporal logic. That is, the properties are expected to hold for intervals. To control the interval of an assertion the start and stop annotations are provided. When checking a stopped assertion, the **indeterminate** value is considered to be false as the property has not been full satisfied during the interval.

The last and most important annotation is "synchronise". This annotation instructs the weaver that the properties has to be checked for consistency. Nevertheless, before performing the runtime checking the variables are updated. The update also includes the reassignment of constant variables as to reflect their values according to the current state, Figure 2.

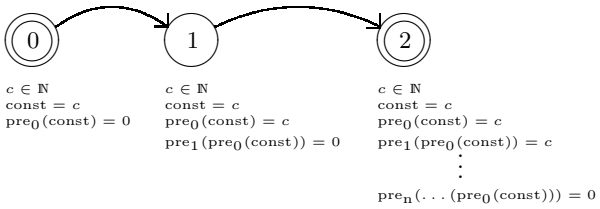

Figure 2: Lustre constant to system state relation

The "synchronise" annotation takes a boolean value. This value instructs the synchronisation method whether to abandon execution on trapping an error. However, independently of the value passed the runtime checker still reports the errors that has been encountered.

4.3 Monitoring

The monitoring process consists in two interdependent tasks. The core task is the evaluation of the properties, which is performed using the Lustre environment. The direct use of the evaluation process is cumbersome. Hence, the validation task is used to abstract the evaluation process and perform the repetitive task of evaluating each property.

4.3.1 Initialisation

The initialisation process consists in converting the monitoring properties from the mathematical notation to a collection Abstract Syntax Trees (ASTs). The conversion process

is performed using a parser as the mathematical notation provides a suitable grammar representation.

The AST data structure is adopted since it provides a suitable visualisation of how complex properties are decomposed into smaller properties. Evaluating the smaller properties is assumed to be simpler, hence, by evaluating the lower nodes in the tree facilitates the process of obtaining the satisfiability value.

4.3.2 Evaluation

The evaluation process is a bottom-up traversal of the AST structure. The simplest nodes to evaluate are the leaf nodes that are either propositions or numeric variables. After mapping the system state value to the leaf nodes, the nodes at a higher level can be evaluated. The evaluation of the non-leaf nodes consists in calling the function related to the operator[2]. The property satisfiability value is then determined by the value obtained by evaluating the root node.

EVALUATE(*Symbolic Automaton*)

```
1  for each node starting from the leaf nodes
2        do expression variable ← evaluation node expression
3  Symbolic automaton validity result ← root node value
```

4.3.3 Validation

The evaluation process described above is encapsulated in the validation process. The property ASTs are evaluated bottom-up, hence, the state variables has to be updated before the starting the evaluation. In algorithm VALIDATE line 1 suspends the system execution to perform the validation process. Therefore, ensuring that the system state is not corrupted during the validation process. When the system is stopped, line 2 updates the state variables to reflect the system variables. That is, performs a transition from the current state to the new state.

The actual validation process consists in performing the evaluation process on the collection of ASTs, lines 3–6. Each AST is checked for validity and one of the 3 logic states is returned. When a property is violated the system reports the error together with a the property trace. Note that, the monitoring system uses symbolic automata to represent the system, hence, it is not possible to depict the entire state according to the execution path, without keeping a history.

VALIDATE

```
1  Stop system execution. // Required for variable integrity
2  Update non-expression variables
3  for each symbolic automaton
4        do Valid ← Evaluate(Symbolic automaton)
5           if Valid == false
6              then Error(Symbolic automaton)
7  Resume system execution. // On the assumption that
                               the system was not
                               aborted due to errors.
```

[2]Refer to [2] for the actual deterministic QDDC execution semantics.

4.4 Design review

D^{3}CA implements a solution for monitoring systems using interval temporal logic. The validation process is performed on a Lustre environment, which allows memory and space requirements to be predetermined.

Properties are cleaner if written in mathematical notation. The validation mechanism provided by D^{3}CA includes a parser that on initialisation of a property the mathematical notation is converted into symbolic automata. The symbolic automata are then stored in an Abstract Syntax Tree data structure as it provides a suitable representation for the evaluation process.

The architecture of the monitored program during runtime is illustrated in Figure 3.

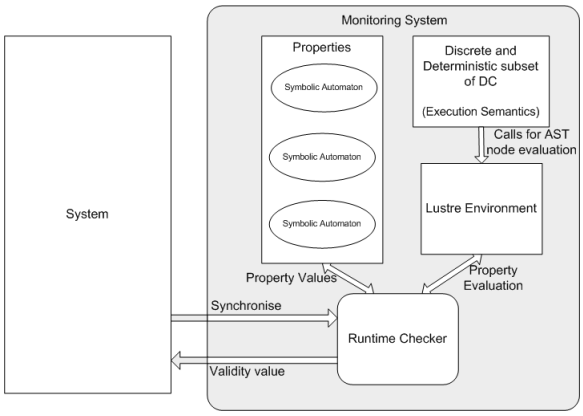

Figure 3: System composition diagram

5. CASE STUDIES

The framework is applied to two simulators, a mine pump and an answering machine.

5.1 Mine Pump

The first case study consists in the adaptation of a commonly used example in Duration Calculus literature [7, 5]. The case study consists in simulating the behaviour of a water extraction pump employed in a mine to lower the level of the water collected in a sump.

A mine has a water (H_2O) and methane (CH_4) leakage. The water leaked is collected in a sump which is monitored by two sensors signalling when the water level is high or dangerous. When the water level is high a pump is started to pump water out. Nevertheless, the pump cannot be started is the methane level is high. Using the notation introduced earlier the property for starting the pump can be defined as,

$$(\llbracket LowH_2O \rrbracket \text{ then } (\text{age}(HighH_2O \wedge \neg HighCH_4) <= \delta \text{ then } \llbracket PumpOn \rrbracket))^*$$

where, δ is the time required for the pump to start operating.

When the pump is operating it takes ϵ time to lower the water level to acceptable level. This property is defined as

$$(\llbracket PumpOn \rrbracket \wedge \eta \leqslant \epsilon \text{ then begin}(LowLowH_2O))^*$$

The last property related to the pump operation is to check that when the water level is low or the methane level

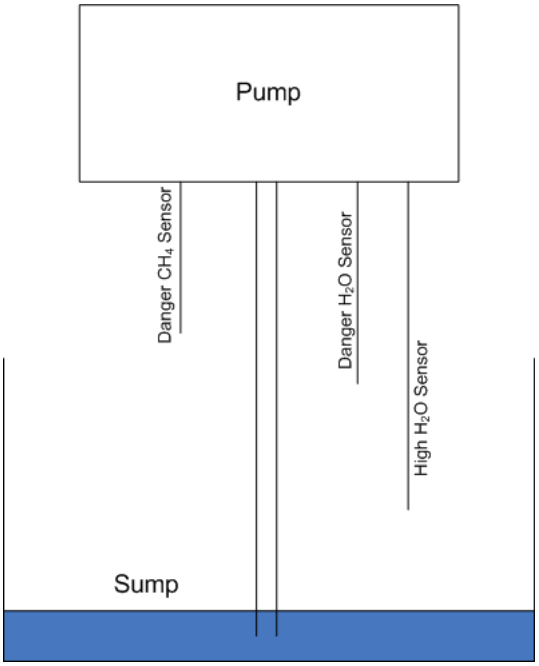

Figure 4: Mine Pump Environment

has rose to the level where it is dangerous to operate machines.

$$(\text{age}(LowH_2O \vee HighCH_4) \leqslant \delta \text{ then } [\![PumpOff]\!])^*$$

The discrete and deterministic Duration Calculus notation allows environment assumptions to be defined. The water level sensors are expected to report the water level in ascending or descending order. That is the water cannot go to dangerous level before it reaches the high level. The first sensor assumption is that the water is at high-level for some time before it reaches the dangerous level.

$$[\![HighH_2O]\!] \overset{\omega}{\hookleftarrow} [\![\neg DangerousH_2O]\!]$$

We can also say that the water is in dangerous levels if it is also at the high level.

$$[\![DangerousH_2O \Rightarrow HighH_2O]\!]$$

The other environment assumptions are related to the methane release. For the mine operations not to be interrupted on frequent intervals, the methane release occurs only at least ζ time after the last methane release. More formally,

$$(\text{age}(\neg HighCH_4) \leqslant \zeta \text{ then } \mathbf{true}) \lessdot\!\!- HighCH_4$$

When deploying the mine pump an assumption is made that methane releases are of short burst thus allowing the pump to be operated.

$$\text{age}(HighCH_4) \leqslant \kappa$$

where κ is the maximum time a methane release can take for the pump to be operatable.

Finally we equip the mine pump with an alarm system to notify the workers that there is possibility of danger. The alarm will go off when either the water reaches the dangerous level or there is a high level of methane in the mine.

$$(\text{age}(DangerousH_2O) \leqslant \delta \text{ then } [\![AlarmOn]\!])^*$$

$$(\text{age}(HighCH_4) \leqslant \delta \text{ then } [\![AlarmOn]\!])^*$$

$$(\text{age}(\neg DangerousH_2O \wedge \neg HighCH_4) \leqslant \delta \text{ then } [\![AlarmOff]\!])^*$$

5.2 Mine Pump Scenario

The scenario presented here consists in simulating a long methane release, which violates the methane release assumption.

The constant variables are initialised as follows: $\delta = 2$, $\epsilon = 7$, $\kappa = 2$, $\omega = 17$, $\zeta = 25$.

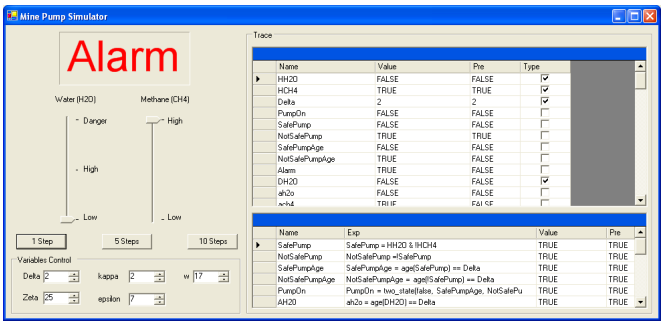

Figure 5: Mine Pump Screen Shot

The simulator has two controls, one for the water level and another for the methane level, which allow the simulation of environment changes. The simulation interface also provide three buttons one to signal a clock tick, another to simulate 5 clock ticks using the same system state and the last button to simulate 10 clock ticks.

To simulate the scenario where a methane release breaks the assumption

$$\text{age}(HighCH_4) \leqslant \kappa$$

the methane controller is set on the high mark. Then since $\kappa = 2$ the 5 clock ticks button is pressed. On the third clock tick the assumption breaks and an assumption violated error is reported back to the user. In order to track back the origin of the error the user is presented back with the assertion that failed and the system state values. In our case the system state is ($\text{age}(HighCH_4) = 3$, $\eta = 3$, $\kappa = 3$, $\text{age}(HighCH_4) \leqslant \kappa =$ **false**). When analysing the system state it is immediately clear that that the methane release was longer than what is expected.

5.3 Answering Machine

This section describes a simulated answering machine on which the D^{3}CA is applied. Figure 6 illustrates the answering machine states and the possible transition between the states.

The answering machine depicted above has four different interval measurements. The time spent in the "idle" and "receiver up" states cannot be determined. Therefore, when specifying the system in Duration Calculus the intervals can be considered as open. While the "ringing" and "recording" intervals that are fixed in length. The ringing interval is set to 10 rings, therefore, the answering machine must start playing the recorded message only if in the meantime the receiver has not been pulled up. The message "recording" interval allows the callee to leave a message for about 3 minutes and then the line is dropped. The last interval measure is determined by the length of the message recorded

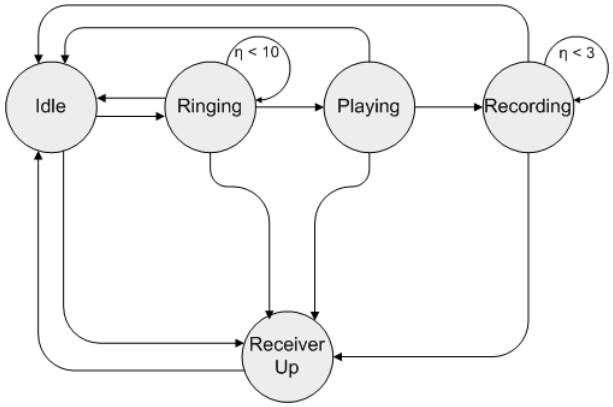

Figure 6: Answering Machine State Diagram

by the answering machine owner. This interval measurement is applied to the "playing" state. Therefore, the specification of the system is

$$\big((\llbracket \text{idle} \rrbracket \text{ then } \text{age}(\text{ringing}) \leqslant 10 \text{ then } \llbracket \text{playing} \rrbracket \text{ then } \text{age}(\text{recording}) \leqslant 3)^* \vee \text{end}(\text{receiver up})\big)^*$$

It can be noted that although the intervals are measured using different units, the specifications consider the length of interval independently of the measuring units. In the case of D^3CA and of this particular case study the different measuring units are handled by the placing of "synchronise" annotation in the system implementations.

The "idle" state reflects that the answering machine is doing no other operation. Hence, when the receiver is lifted up the answering machine is expected to be idle. The "receiver up" state in Figure 6 is included to show that the idle state of the answering machine and the idle state when the receiver is up are different.

The "idle" state property can be specified in terms of the answering machine states as

$$\text{idle} = \neg\text{ringing} \wedge \neg\text{playing} \wedge \neg\text{recording}.$$

The simplest assumption properties to verify are related to the transition from one state to the next, where the next state has only one entrance path.

$$\text{idle} \prec\!\!\!\!\!\!- \text{ ringing}$$

$$\text{age}(\text{ringing}) \leqslant 9 \prec\!\!\!\!\!\!- \text{ playing}$$

$$\text{playing} \prec\!\!\!\!\!\!- \text{ recording}$$

The answering machine under design assumes that when the receiver is up then no other calls can come in. That is, there is no multiplexing between different lines. When the receiver is up, the answering machine should be idle.

$$\text{end}(\text{receiver up}) \implies \text{end}(\text{idle})$$

The answering machine is then simulated in relation to the above operation properties and assumption properties. The properties specified above were able to trigger all the errors inserted in the simulation. Hence, they forced the behaviour of the system to the specifications.

5.4 Answering Machine Scenario

The mine pump scenario showed how the framework can trigger violations to state properties. In this scenario we show that the framework is also capable of detecting violations to state transitions. This is illustrated by a small simulation that violates the transition from ringing to playing the recorded message.

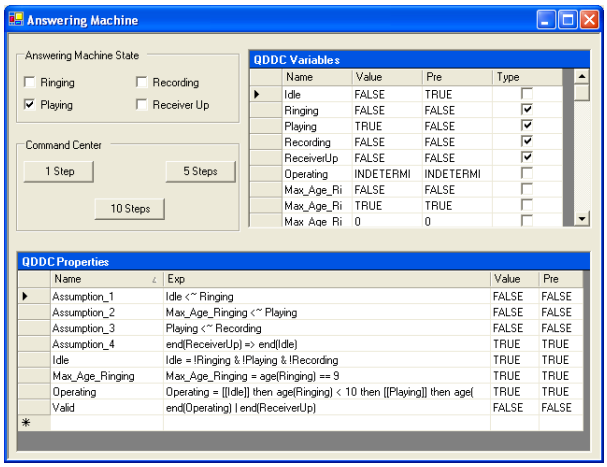

Figure 7: Answering Machine Screen Shot

The answering machine simulator has a number of control buttons, one for every state except for "idle". The state "idle" is represented by unmarking all the other controls. When the system starts the state is immediately set to "idle". A number of steps are performed to leaving the state as "idle". Then the phone starts ringing, so the "ringing" control is marked and a number of clock ticks are simulated. However, after the 5 ringing tone the "playing" state is marked. This violates the transition assumption

$$\text{age}(\text{ringing}) \leqslant 9 \prec\!\!\!\!\!\!- \text{ playing}$$

as only 5 ringing tones has been performed. The simulator immediately reports the error, figure 8. The error shows the property violated together with the values of each subexpression.

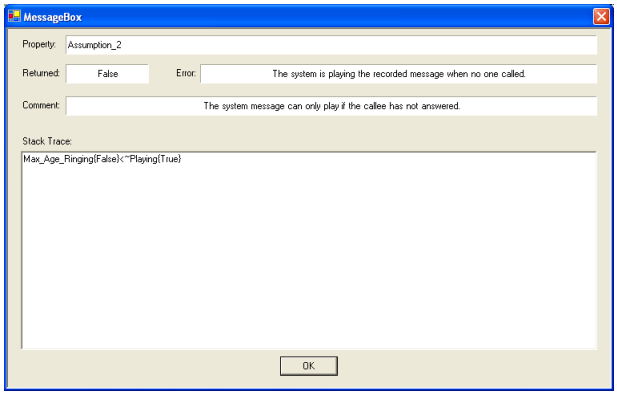

Figure 8: Answering Machine Error Report

From the two simple scenarios presented in this section it was shown that the framework has the potential to define

different properties of the system. The framework hides all the complexities related to notation interpretation in programming language and in defining the monitoring system.

The benefit of using Interval Temporal Logic monitors is the increase in reliability of the system without the need to overwhelm the system with point-logic assertions.

6. CONCLUSION

The use of validation for testing software correctness is a well applied concept, and different scenarios lead to the use of different validation approaches. In this paper we showed how Interval Temporal Logic validation can be integrated with normal applications and in real-life scenarios. The integration is obtained through the use of a framework that allows to predetermine the space and time requirements for computing state satisfiability. The framework presented in this paper simplifies the migration from one scenario to another by freeing the validation part from environment and platform dependencies.

7. REFERENCES

[1] H. Barringer, A. Goldberg, K. Havelund, and K. Sen. Eagle monitors by collecting facts and generating obligations. Technical Report Pre-Print CSPP-26, University of Manchester, Department of Computer Science, University of Manchester, October 2003.

[2] K. D'Emanuele. Runtime monitoring of duration calculus assertions for real-time applications. Master's thesis, Computer Science and A.I. Department, University of Malta, 2006. To be submitted.

[3] D. Drusinsky. The temporal rover and the ATG rover. In *SPIN*, pages 323–330, 2000.

[4] N. Halbwachs. Synchronous programming of reactive systems. In *Computer Aided Verification*, pages 1–16, 1998.

[5] M. Joseph, editor. *Real-time systems specification, verification and analysis*. Tata Research Development and Design Centre, Jume 2001.

[6] M. Kim, S. Kannan, I. Lee, O. Sokolsky, and M. Viswanathan. Java-MaC: a run-time assurance tool for Java prgrams. In *1st Workshop on Runtime Verification (RV'01), volume 55 of ENTCS*, 2001.

[7] P. Pandya. Specifying and deciding quantified discrete-time duration calculus formulae using DCVALID. Technical Report TCS00-PKP-1, Tata Institute of Fundamental Research, 2000.

[8] S. Thaker. Runtime monitoring temporal property specification through code assertions. Department of Computer Science, University of Texas at Austin, 2005.

Author Index