

INVITED SPEAKERS

Corpus-Based Knowledge Representation

Alon Y. Halevy

University of Washington
Seattle, Washington, U.S.A.
alon@cs.washington.edu

Jayant Madhavan

University of Washington
Seattle, Washington, U.S.A.
jayant@cs.washington.edu

Abstract

A corpus-based knowledge representation system consists of a large collection of disparate knowledge fragments or schemas, and a rich set of statistics computed over the corpus. We argue that by collecting such a corpus and computing the appropriate statistics, corpus-based representation offers an alternative to traditional knowledge representation for a broad class of applications. The key advantage of corpus-based representation is that we avoid the laborious process of building a (often brittle) knowledge base. We describe the basic building blocks of a corpus-based representation system and a set of applications for which such a paradigm is appropriate, including one application where the approach is already showing promising results.

1 Introduction

Declarative representation of knowledge has long been acknowledged as a key building block of AI systems. A knowledge base (KB) with associated reasoning mechanisms can serve as a backbone for tasks such as query answering, learning and diagnosis. One of the key challenges in deploying knowledge representation (KR) systems is the cost and the complexity of building the knowledge base, especially in cases where it needs to cover a broad domain. Building a KB is highly labor intensive for several reasons. First, the acquisition of domain knowledge and articulating it in a formal language can be incredibly difficult. Second, since the KB needs to form a coherent logical whole, it is hard to distribute its creation of a knowledge base over multiple experts or knowledge engineers. Finally, no matter how broad the KB, it will be brittle at the edges of its knowledge – it can only be used for tasks whose knowledge needs have been anticipated in advance. While all of the aforementioned difficulties have received significant attention, together they still remain a fundamental challenge for knowledge representation technology.

This paper argues that there is an alternative approach to knowledge representation that has the potential of providing competitive functionality for a broad class of applications. In a nutshell, this approach, which we term *corpus based representation*, is based on collecting a large corpus of disparate fragments of knowledge, and building a set of tools that are based on analyzing properties of the corpus. The

fragments of knowledge in the corpus can include individual KBs, database schemas with or without data instances of the schema, queries written over KBs and databases, and any form of meta-data associated with them. Unlike a KB that needs careful ontological design, the corpus is a set independent uncoordinated contributions. The intuition is that if the corpus is large enough, then the patterns we identify in it can be of great use for knowledge intensive tasks.

Corpus-based knowledge representation is an outgrowth of our work on *schema and ontology matching* [Doan *et al.*, 2001; 2002]. The matching problem (which we elaborate on in Section 2), is to find a semantic mapping between two disparate representations, be they database schemas or ontologies. While a detailed KB about the domain of the representations being matched could be of great value for the matching task, such KBs rarely exist, and their construction is not cost effective. Instead, our approach to matching is based on analyzing the variations of representations in a corpus of schemas. This paper extends the idea of using a corpus to a general approach to knowledge representation. Specifically, we identify other tasks in which corpus-based representation can be useful, and then define preliminary version for the contents of a corpus and the interfaces it should support.

We emphasize that corpus-based representation is not a replacement for traditional knowledge representation. There are many tasks in which very finely tuned reasoning is required, and such reasoning can only be done with a very well designed knowledge base (e.g., medical diagnosis, monitoring spacecraft, and making sense of tax laws). Our goal is to explore the space of problems in which the laborious construction of knowledge bases can be avoided.

2 A motivating application: matching disparate representations

The discussion in this paper spans many kinds of representations, ranging from database schema to ontologies in expressive KR formalisms. Collectively, we use the term *domain model* to refer to representations in any of these formalisms. The term *schema* refers to intensional knowledge, such as the names of database tables and their respective columns, or to the terminological component of a KB. The ground facts (or, rows in a database) are called *instance data*.

Sharing data among multiple data sources and applica-

tions is a problem that arises time and again in large enterprises, B2B settings, coordination between government agencies, large-scale science projects, and on the World-Wide Web. The problem has received significant attention in both the AI and Database communities, and over the years, several paradigms for data sharing have been implemented, each appropriate for certain scenarios. Examples include data warehousing oriented architectures, data integration systems [Lenzerini, 2002; Draper *et al.*, 2001], message passing systems (e.g., [MQSERIES, 2003]), web services, and peer-data management systems [Halevy *et al.*, 2003b; Kalnis *et al.*, 2002; Bernstein *et al.*, 2002].¹

No matter what paradigm we employ to share the data, a key problem is the semantic heterogeneity between the domain models of data sources that were originally designed independently. Thus, to obtain meaningful interoperation, one needs a *semantic mapping* between the schemas. A semantic mapping is a set of expressions that specify how the data in one database corresponds to the data in another database [Madhavan *et al.*, 2002]. While languages for specifying semantic mappings have been developed and are well understood (see [Lenzerini, 2002; Halevy, 2001] for surveys), the creation of semantic mappings has become a key bottleneck as it is labor intensive and error-prone.

The goal of schema matching is to assist a human to relate two domain models. Complete automation of the process is unlikely to be possible, but the goal is to significantly increase the productivity of human experts. The matching problem is difficult because it requires understanding the underlying semantics of the domain models being matched. While a domain model (with its instance data) provides many clues on its intended semantics, it does not suffice in order to relate it to a different domain model.

The process of generating a semantic mapping has traditionally been divided into two phases. The first phase finds a *match* between the two schemas. The match result is a set of *correspondences* between elements in the two schemas, stating that these elements are *somehow* related. For example, a correspondence may state that *agent* in one domain model corresponds to *contact* in another. The second phase builds on the correspondences by creating the mapping expressions. The mapping expressions, often expressed as queries or rules, enable translating data from one data source to another, or reformulating a query over one data source into a query on the other. A plethora of techniques have been proposed for schema matching: see [Rahm and Bernstein, 2001] for a survey, and [Noy and Musen, 2002; Doan *et al.*, 2002; Do and Rahm, 2002] for some work since then. Collectively, these techniques mirror the heuristics that a human designer may follow. For example, techniques have considered exploiting relationships between names of elements in the domain models, structural similarities between them, similarities in data values, and even correlations between values in different attributes. Several recent works on schema matching are based on *combining* multiple techniques in a principled fashion [Madhavan *et al.*, 2001; Doan *et al.*, 2001;

Do and Rahm, 2002].

Corpus-based matching

Conceivably, detailed knowledge about the domain in which the matching is being performed can be an important resource in a schema matching system. However, creating an appropriate KB is often hard, and furthermore, the result may be brittle in the sense it only helps on its domain of coverage, and only provides a *single* perspective on the domain.

We are pursuing an alternate approach in which knowledge is gleaned by analyzing a large corpus of database schemas and previously validated mappings. There are two types of knowledge that we can glean from such a corpus. First, we can learn the different ways in which words (or terms) are used in database structures (i.e., as relation names, attribute names and data values). Second, the validated mappings show how variations in term usages correspond to each other in disparate structures.

Although such a corpus is not easy to construct, it is a very different kind of activity than building a detailed and comprehensive knowledge base. It does not require the careful ontological design as a knowledge base does, nor the careful control on its contents, thereby removing the key bottlenecks present in the design of knowledge bases. The corpus offers *multiple perspectives* on modeling a particular domain, including different coverages of the domain. Thereby, it is more likely to provide knowledge that is useful for matching two disparate schemas.

In our initial work on the LSD system [Doan *et al.*, 2001], we investigated the benefit of learning from previously validated mappings. In [Doan *et al.*, 2001] we considered the case where multiple data sources are mapped to a single *mediated schema*, on which users pose queries. We provided LSD with the mediated schema and a set of *training matches* for some data sources. LSD used these matches to learn models of the elements of the mediated schema. Since no single learning algorithm captures all the cues from the domain, we used a multi-strategy approach that combined the predictions of several learners. We then asked LSD to predict matches between the mediated schema and a set of test schemas. Our experiments showed that (1) it is possible to achieve high accuracy with multi-strategy learning, and (2) additional accuracy is obtained by considering domain constraints (i.e., a simple form of domain knowledge). Overall, LSD achieved matching accuracy of 75-90% on small to medium sized schemas of data sources on the Web. We extended LSD to consider simple taxonomies of concepts in [Doan *et al.*, 2002].

In recent work [Madhavan *et al.*, 2003], we investigate the benefit of a corpus of schemas and matches, and the ability to use such a corpus to predict mappings between a *pair* of schemas that have not been previously seen. Like in LSD, we learn models for elements in the corpus, using both the information available in the schema and validated matches that are provided in the corpus. Given two schemas, S_1 and S_2 , we calculate for each element in them a *similarity vector* w.r.t. the corpus, i.e., how similar each element in S_i is to each element in the corpus. Very roughly speaking, if the similarity vectors of two elements $a_1 \in S_1$ and $a_2 \in S_2$ are similar to each other, then we predict that a_1 matches a_2 . The results of

¹This paper accompanies an invited talk at the conference that surveys the successes and challenges in data integration to date.

our experiments show that (1) even with a modest corpus of 10 schemas we are able to achieve good accuracy, and (2) the correct matches found by using the corpus and those found by other previously known techniques overlap, but have significant differences. Hence, the use of the corpus is finding matches that would not have been predicted by other techniques.

3 The corpus-based representation principle

This paper argues that using a corpus of domain models is a general technique that can be applied in many applications as an alternative or complement to traditional knowledge representation. Section 3.1 describes a few such applications, and Section 3.2 presents the components of a corpus-based representation system.

3.1 Applications of corpus-based representation

The following are classes of applications that can benefit from corpus-based representation.

Web search and query answering: the first class of applications concerns various information finding tasks on the Web (or intranet). The first application is *query answering* on the web (e.g., [Kwok *et al.*, 2001; Radev *et al.*, 2002]): a natural language query is posed to a web search interface, and rather than finding relevant pages, the search engine tries to find the answers to the query. A second, very related application, is *focused crawling* (e.g., [Sizov *et al.*, 2003]), where the search engine is given a particular topic, and it tries to find pages relevant to it by starting at a set of initial pages, and crawling from them in a focused fashion. In both of these applications, the presence of additional domain knowledge has been shown or argued to be useful. However, the cost of constructing knowledge bases with such wide domain coverage would be prohibitive.

Creating and querying structured knowledge: one of the greatest impediments to using database and knowledge base technology is the conceptual difficulty of dealing with structured data versus unstructured text (the so called *structure chasm* [Halevy *et al.*, 2003a]). It is well known that creating domain models (whether for database schema or knowledge bases) is a conceptual effort. Querying requires a different kind of effort – understanding someone else’s domain model. Hence, a significant challenge for the KR and Database communities is to create tools that facilitate the creation and querying of structured data. [Halevy *et al.*, 2003a] argues that a corpus of schemas enables building several tools for this purpose.

One such tool would be a *schema design advisor*, which assists in the authoring of structured data, much in the spirit of an auto-complete feature. A user of the such a tool creates a schema fragment and some data in a particular domain, and the tool then proposes extensions to the schema using the corpus. The user may choose a schema from the list and modify it further to fit the local context. Note that besides time savings, such a tool has other advantages, such as possibly resulting in better designs and helping users conform to

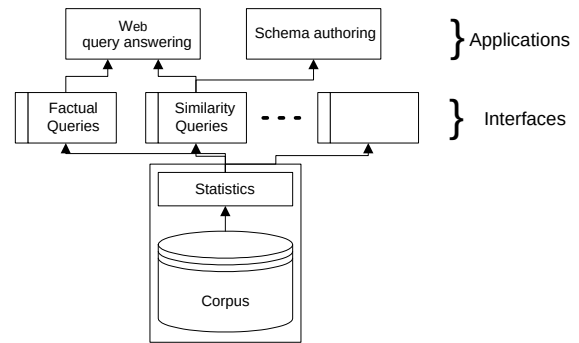


Figure 1: A corpus-based representation system contains a large collection of disparate domain models, instance data, validated mappings between domain models and meta-data. Applications access the corpus and the statistics computed on it through a set of interfaces. The interfaces support different types factual queries and similarity queries.

certain standards, when these are applicable. One can also imagine using corpus-based representation during the phase of verifying the correctness of a knowledge base. Hence, corpus-based representation can be an aid in creating traditional knowledge-based systems.

On the querying side, corpus-based representation can facilitate the querying of unfamiliar data. Specifically, consider a tool that enables you to pose a query using *your own terminology* to any database. The tool would then use the corpus to propose reformulations of the your query that are well formed w.r.t. the schemas of the database at hand. The tool may propose a few such queries (possibly with example answers), and let you choose among them or refine them.

General properties of candidate applications: the exact characterization of which applications lend themselves to corpus-based representation is a subject for future research. One property of such applications that immediately stands out is that they use the knowledge base as an aid in resolving ambiguity. That is, the applications use a set of techniques to propose plausible answers to a task, and then use the knowledge base to help *rank* the answers, *rule out* certain ones, or *guide* the search for answers. Hence, beyond the applications mentioned above, corpus-based representation can be useful in learning tasks, natural language processing and interfaces [Popescu *et al.*, 2003], and information retrieval. In fact, as we explain in Section 4, corpus-based representation was inspired by the use of corpora (of different kinds!) in natural language processing and information retrieval. In contrast, corpus-based representation would be of limited use in applications that require intricate logical inference on carefully designed domain models (e.g., medical diagnosis, control of spacecraft, or reasoning about income-tax rules).

3.2 Corpus-based representation systems

We now offer some specific details on how corpus-based representation systems can be built and used. We describe the types of contents we may put in a corpus, and the interfaces that the corpus provides to an application (see Figure 1).

Contents of a corpus

A corpus can include any kind of information related to structured data. In particular, it includes the following.

1. Various forms of domain-model information: on the less expressive end, the corpus can include relational and object-oriented database schema or entity/relationship diagrams, XML DTDs or schemas, possibly with associated integrity constraints (e.g., functional dependencies). On the more expressive end, the corpus can include terminologies.

2. Instance data: we can include the actual rows of tables (or representative rows), XML documents, or the assertion (A-Box) component of a terminological knowledge base or Horn theory. In fact we can include data sets that do not have a schema (e.g., certain file formats). Note that it is very often the case that elements in the schema of one model may be instance data in another model. Hence, the distinction between schema and instance data is not clean cut.

3. Validated mappings: mappings can be given directly between pairs of domain models, or go through an intermediate domain model [Madhavan *et al.*, 2002].

4. Queries: queries (posed by users or applications) provide important information about how certain data is used. For example, when a database query performs a join over attributes of two different tables, that indicates that the columns are modeling the same domain (and this often not evident from the schema that only specifies the data type).

5. Other meta-data: there are many forms of meta-data that accompany domain models. They range from text descriptions of the meaning of different fields to statistics about table cardinalities or histograms on the set of possible values within a column.

It is important to emphasize that a corpus is *not* a logically coherent universal database. Rather, it is a collection of disparate structures, and there is relatively little control on the logical design of elements of the corpus. We expect that the domain-model information of the corpus will be stored and accessed using tools for model management [Bernstein, 2003], which attempt to provide a set of operators for manipulating models of data (as opposed to the data itself).

Statistics on the corpus

There is a plethora of possible analyses that can be performed on such a corpus, and finding the most effective ones is a long term research challenge. Below we describe certain kinds of statistics that can be computed over the corpus. We classify them according to whether they apply to individual words or terms, partial structures, or to elements of particular schemas.

Word and term statistics: these statistics are associated with individual words (in any language) and with noun or verb phrases. These statistics indicate how a word is used in different roles in structured data. For each of these statistics, we can maintain different versions, depending on whether we take into consideration only word stemming, synonym tables, inter-language dictionaries, or any combination of these three. The statistics include:

1. Term usage: How frequently the term is used as a relation name, attribute name, or in data (as a percent of all of its uses or as a percent of structures in the corpus).

2. Co-occurring schema elements: For each of the different uses of a term, which relation names and attributes tend to appear with it? What tend to be the names of related tables and their attribute names? What tend to be the join predicates on pairs of tables? Are there clusters of attribute names that appear in conjunction? Are there mutually exclusive uses of attribute names?

3. Similar names: For each of the uses of a term, which other words tend to be used with similar statistical characteristics?

Composite statistics: the same statistics can be applied to *partial structures*. Examples of partial structures are sets of data instances, relations with associated attribute names, a relation with some data (possibly with missing values).

Clearly, we need to significantly limit the number of partial structures for which we keep statistics (e.g., use techniques for discovering partial structures that occur frequently e.g., [Polyzotis and Garofalkis, 2002]). Given statistics for certain partial structures, we can estimate the statistics for other related structures.

Statistics for schema elements: the same word, used in different structures, can have different meanings. Hence, we may want to characterize the specific usages of terms in structures, and relate them to usage of terms in other structures. For example, in [Madhavan *et al.*, 2003] we learn a classifier for every relation and attribute name in the corpus. Following [Doan *et al.*, 2001], we use meta-strategy learning. The training data used for learning is gleaned from the schema to which the element belongs and the training data of elements that have been mapped to it by a validated mapping in the corpus. Intuitively, the classifier is meant to recognize the particular usage of the term, even if it appears differently in another structure.

Application interfaces to the corpus

Applications built on a corpus should be able to access directly the statistics computed on the corpus, and perform the obvious lookups and queries on the corpus. In addition, just as knowledge based systems provide a set of higher-level interfaces (e.g., in the spirit of [Chaudhri *et al.*, 1998]), the same could be done for corpus-based representation systems. The following are examples of functions such an interface could support. Note that the implementation of these interfaces poses many research challenges.

We classify the interfaces on whether they ask for *factual* queries, or *similarity* queries. In all cases, we expect that the answers returned will be a ranked list of answers. In addition, as in traditional knowledge bases, we should be able to obtain an *explanation* for the answers. In the notation used below, variables are prefaced by \$.

Factual information: in the simplest case, a query is a non-ground formula, and the corpus should return the possible groundings of the formula. As a simple example, we can ask for the values of \$x, where Killed(\$x, Elvis). A different example would be a higher-order query, e.g., find all the \$P, such that \$P(Toyota, Honda).

A more complicated function would take as input a set of constants and return formulas that include all of these constants. Here, the goal is to find how two terms are re-

lated without specifying exactly what role they play in a formula. For example, the input may be *GPA* and *studentID*, and the answers could be *GPA(studentID, \$value)* and *Student(studentID, GPA, address)*. Note that the two terms play different roles in the two answers, and that the answers are templates for formulas (i.e., schema description), rather than particular sets of formulas.

In both of these cases, one can also complement the query with background knowledge. The knowledge may eliminate some answers or result in a different ranking of answers.

Similarity information: one of the key imports of the corpus is that it contains *different* ways of saying similar facts. Similarity queries try to get at these different perspectives. In the simplest case, a query can be a ground fact, and the corpus will return similar ways of saying the same thing. For example, the input can be *Class(Lexus, luxury)*, and the output could be *LuxuryCar(Lexus, Toyota)*, or even *CarReview(Lexus, LuxuryClass, VeryGood, goodTires)*.

In a more complex case, the query would consist of two terms, and the answer would be sets of facts in which these terms tend to play similar roles. For example, the input could be *Review* and *Referee*, and the answer would be *IjcaiReview(paper1, reviewer2, accept)* and *AIJournalReferees(round2, paper3, reviewer4, reject)*. The answer could also specify where the terms are used differently, such as refereeing a soccer game and reviewing a movie.

In summary, we note some of the fundamental challenges in building an interface to a corpus. First, unlike logical knowledge/databases, answers are ranked rather than defining a boolean condition on data. Second, we are often interested in similarity querying, looking for different ways of saying the same thing². Third, the answers returned often summarize sets of facts or describe schema fragments. Finally, it is often useful to tell the user what is *not* an answer.

4 Related work

The corpus-based representation approach stands in stark contrast to an approach like the Cyc Project [Lenat and Guha, 1990] that attempts to build a comprehensive knowledge base of all common sense. A corpus is a collection of any set of small domain models that are not centrally designed or coordinated. Importantly, the corpus is likely to include several models of the same domain. One of the key engineering challenges that arose in building Cyc was how to enable several knowledge engineers to add to such a large knowledge base without having to understand the entire knowledge base. *Contexts* (e.g., [Guha, 1991; Nayak, 1994; Giunchiglia and Ghidini, 1998]) are a mechanism proposed to address this complexity, where each knowledge engineer can focus on a small knowledge base, and then a set of lifting rules makes sure the entire knowledge base is semantically related. Unlike a corpus, contexts are still logically coordinated through a set of lifting rules.

The use of a corpus of structured data was first proposed in [Halevy *et al.*, 2003a]. There, the goal was to create tools

²There is a large literature on similarity querying in databases and content based image retrieval, but these focus on similarities between objects represented as points in multi-dimensional space.

that facilitate the creation, querying and sharing of structured data. That work was partially inspired by the use of large corpora in information retrieval and in statistical natural language processing (e.g. [Charniak, 1997]).

Several works have considered creating knowledge bases by a large set of independent contributors. The Open Minds Project [Singh *et al.*, 2002] collects knowledge from contributors on the web and builds a KB of common sense. Although the contributors are independent, the knowledge entry is guided by a set of templates, and hence the creation of a coherent KB is possible. [Richardson and Domingos, 2003] describe how to create a feedback loop by which consumers of knowledge provide indirect feedback to contributors of knowledge as to the usefulness of their contributions, thereby ultimately filtering out the bad contributions.

5 Conclusions

The analysis of large corpora is the key to the success of Information Retrieval techniques and to recent progress on Natural Language Processing. Corpus-based representation is predicated on the idea that large corpora of domain models can be leveraged to create novel tools in service of knowledge representation. The key advantage of corpus-based representation is that it avoids the need for careful logical design of a single comprehensive ontology. In addition, since a corpus will contain multiple ways of representing the same information, it opens up opportunities for exploring transformations on data and addressing problems related to representational heterogeneity. This paper laid the foundation for corpus-based representation, by describing preliminary versions of the contents of a corpus, the interfaces to it, some applications in which it would be a powerful representational tool, and an application from which the approach was inspired.

Corpus-based representation offers many exciting research challenges, not the least of which is actually collecting a large enough corpus to be of interest. In addition, there is a question about how focused the corpus needs to be in order to be useful in a particular domain. That is, can it only have domain models in that domain? Can domains models in other (possibly related) domains be useful, or do they only introduce noise that degrades the performance? Finally, no matter how a corpus is constructed, it can probably be manually *tuned* to perform even better. What form does such tuning take? While the challenges for corpus-based representation are enormous, we believe the payoffs could be huge, and the results can profoundly impact how we create and use structured knowledge.

Acknowledgments

Many colleagues have contributed to the ideas that led to corpus-based representation. We would like to thank Phil Bernstein, Anhui Doan, Pedro Domingos, Oren Etzioni, Zack Ives, Pradeep Shenoy, and Igor Tatarinov. Funding was provided by an NSF CAREER/PECASE Grant IIS-9985114, ITR Grant IIS-0205635, a Sloan Fellowship, and gifts from Microsoft Research Intel, NEC and NTT.

References

- [Bernstein *et al.*, 2002] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing : A vision. In *Proceedings of the WebDB Workshop*, 2002.
- [Bernstein, 2003] Philip A. Bernstein. Applying Model Management to Classical Meta Data Problems. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2003.
- [Charniak, 1997] Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of AAAI*, pages 598–603, 1997.
- [Chaudhri *et al.*, 1998] Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James Rice. OKBC: a programmatic foundation for knowledge base interoperability. In *Proceedings of AAAI*, pages 600–607, 1998.
- [Do and Rahm, 2002] Hong-Hai Do and Erhard Rahm. COMA - a system for flexible combination of schema matching approaches. In *Proc. of VLDB*, 2002.
- [Doan *et al.*, 2001] Anhui Doan, Pedro Domingos, and Alon Halevy. Reconciling schemas of disparate data sources: a machine learning approach. In *Proc. of SIGMOD*, 2001.
- [Doan *et al.*, 2002] Anhui Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Learning to map between ontologies on the semantic web. In *Proc. of the Int. WWW Conf.*, 2002.
- [Draper *et al.*, 2001] Denise Draper, Alon Y. Halevy, and Daniel S. Weld. The nimble integration system. In *Proc. of SIGMOD*, 2001.
- [Giunchiglia and Ghidini, 1998] Fausto Giunchiglia and Chiara Ghidini. Local models semantics, or contextual reasoning = locality + compatibility. In *Proceedings of KR*, pages 282–291, 1998.
- [Guha, 1991] Ramanathan V. Guha. *Contexts: A Formalization and Some Applications*. PhD thesis, Stanford University, Stanford, CA, 1991.
- [Halevy *et al.*, 2003a] Alon Halevy, Oren Etzioni, Anhui Doan, Zachary Ives, Jayant Madhavan, Luke McDowell, and Igor Tatarinov. Crossing the structure chasm. In *Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR)*, 2003.
- [Halevy *et al.*, 2003b] Alon Y. Halevy, Zachary G. Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In *Proc. of ICDE*, 2003.
- [Halevy, 2001] Alon Y. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4), 2001.
- [Kalnis *et al.*, 2002] P. Kalnis, W. Ng, B. Ooi, D. Papadias, and K. Tan. An adaptive peer-to-peer network for distributed caching of olap results. In *Proc. of SIGMOD*, 2002.
- [Kwok *et al.*, 2001] Cody Kwok, Oren Etzioni, and Dan Weld. Scaling question answering to the web. In *Proc. of the Int. WWW Conf.*, pages 150–161, 2001.
- [Lenat and Guha, 1990] Douglas B. Lenat and R.V. Guha. *Building Large Knowledge Bases*. Addison Wesley, Reading Mass., 1990.
- [Lenzerini, 2002] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS*, 2002.
- [Madhavan *et al.*, 2001] Jayant Madhavan, Phil Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *Proceedings of the International Conference on Very Large Databases (VLDB)*, 2001.
- [Madhavan *et al.*, 2002] Jayant Madhavan, Philip Bernstein, Pedro Domingos, and Alon Halevy. Representing and reasoning about mappings between domain models. In *Proceedings of AAAI*, 2002.
- [Madhavan *et al.*, 2003] Jayant Madhavan, Philip Bernstein, Kuang Chen, Alon Halevy, and Pradeep Shenoy. Matching schemas by learning from others. In *Working notes of the IJCAI-03 workshop on Data Integration on the Web*, 2003.
- [MQSERIES, 2003] <http://www-3.ibm.com/software/ts/mqseries/>, 2003.
- [Nayak, 1994] P. Pandurang Nayak. Representing multiple theories. In *Proceedings of AAAI*, pages 1154–1160, 1994.
- [Noy and Musen, 2002] Natalya Noy and Mark A. Musen. PROMPTDIFF: A fixed-point algorithm for comparing ontology versions. In *Proceedings of AAAI*, 2002.
- [Polyzotis and Garofalkis, 2002] Neoklis Polyzotis and Minos N. Garofalkis. Statistical synopses for graph-structured XML databases. In *SIGMOD '02*, 2002.
- [Popescu *et al.*, 2003] Ana-Marie Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. In *Proceedings of the conference on Intelligent User Interfaces*, 2003.
- [Radev *et al.*, 2002] Dragomir R. Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. Probabilistic question answering on the web. In *Proc. of the Int. WWW Conf.*, pages 408–419, 2002.
- [Rahm and Bernstein, 2001] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
- [Richardson and Domingos, 2003] Mathew Richardson and Pedro Domingos. Building large knowledge bases by mass collaboration. Technical Report 03-02-04, Department of CSE, University of Washington, Seattle, WA, 2003.
- [Singh *et al.*, 2002] P. Singh, L. Thomas, E. Mueller, G. Lim, T. Perkins, and W. zhu. Open mind common sense: Knowledge acquisition from the general public. In *Proceedings of ODBASE*, 2002.
- [Sizov *et al.*, 2003] S. Sizov, M. Biwer, J. Graupmann, S. Siersdorfer, M. Theobald, G. Weikum, and P. Zimmer. The BINGO! system for information portal generation and expert web search. In *Proceedings of the First Biannual Conference on Innovative Data Systems Research (CIDR), Asilomar(CA)*, 2003.

Challenges in Web Search Engines

Monika R. Henzinger
Google Inc.
2400 Bayshore Parkway
Mountain View, CA 94043
monika@google.com

Rajeev Motwani*
Department of Computer Science
Stanford University
Stanford, CA 94305
rajeev@cs.stanford.edu

Craig Silverstein
Google Inc.
2400 Bayshore Parkway
Mountain View, CA 94043
csilvers@google.com

Abstract

This article presents a high-level discussion of some problems that are unique to web search engines. The goal is to raise awareness and stimulate research in these areas.

1 Introduction

Web search engines are faced with a number of difficult problems in maintaining or enhancing the quality of their performance. These problems are either unique to this domain, or novel variants of problems that have been studied in the literature. Our goal in writing this article is to raise awareness of several problems that we believe could benefit from increased study by the research community. We deliberately ignore interesting and difficult problems that are already the subject of active research. An earlier version of this paper appeared in [Henzinger *et al.*, 2002].

We begin with a high-level description of the problems that we describe in further detail in the subsequent sections.

Spam. Users of web search engines tend to examine only the first page of search results. Silverstein *et al.* [Silverstein *et al.*, 1999] showed that for 85% of the queries only the first result screen is requested. Thus, inclusion in the first result screen, which usually shows the top 10 results, can lead to an increase in traffic to a web site, while exclusion means that only a small fraction of the users will actually see a link to the web site. For commercially-oriented web sites, whose income depends on their traffic, it is in their interest to be ranked within the top 10 results for a query relevant to the content of the web site.

To achieve this goal, some web authors try to deliberately manipulate their placement in the ranking order of various search engines. The result of this process is commonly called *search engine spam*. In this paper we will simply refer to it as *spam*. To achieve high rankings, authors either use a *text-based* approach, a *link-based* approach, a *cloaking* approach,

or a combination thereof. There are web ranking optimization services which, for a fee, claim to place a given web site highly on a given search engine.

Unfortunately, spamming has become so prevalent that every commercial search engine has had to take measures to identify and remove spam. Without such measures, the quality of the rankings suffers severely.

Traditional research in information retrieval has not had to deal with this problem of “malicious” content in the corpora. Quite certainly, this problem is not present in the benchmark document collections used by researchers in the past; indeed, those collections consist exclusively of high-quality content such as newspaper or scientific articles. Similarly, the spam problem is not present in the context of intranets, the web that exists within a corporation.

One approach to deal with the spam problem is to construct a spam classifier that tries to label pages as spam or not-spam. This is a challenging problem, which to the best of our knowledge has not been addressed to date.

Content Quality. Even if the spam problem did not exist, there are many troubling issues concerned with the quality of the content on the web. The web is full of noisy, low-quality, unreliable, and indeed contradictory content. A reasonable approach for relatively high-quality content would be to assume that every document in a collection is authoritative and accurate, design techniques for this context, and then tweak the techniques to incorporate the possibility of low-quality content. However, the democratic nature of content creation on the web leads to a corpus that is fundamentally noisy and of poor quality, and useful information emerges only in a statistical sense. In designing a high-quality search engine, one has to start with the assumption that a typical document cannot be “trusted” in isolation; rather, it is the synthesis of a large number of low-quality documents that provides the best set of results.

As a first step in the direction outlined above, it would be extremely helpful for web search engines to be able to identify the quality of web pages independent of a given user request. There have been link-based approaches, for instance PageRank [Brin and Page, 1998], for estimating the quality of web pages. However, PageRank only uses the link structure of the web to estimate page quality. It seems to us that a better estimate of the quality of a page requires additional

*Part of this work was done while the author was visiting Google Inc. Work also supported in part by NSF Grant IIS-0118173, and research grants from the Okawa Foundation and Veritas.

sources of information, both within a page (e.g., the reading-level of a page) and across different pages (e.g., correlation of content).

Quality Evaluation. Evaluating the quality of different ranking algorithms is a notoriously difficult problem. Commercial search engines have the benefit of large amounts of user-behavior data they can use to help evaluate ranking. Users usually will not make the effort to give explicit feedback but nonetheless leave implicit feedback information such as the results on which they clicked. The research issue is to exploit the implicit feedback to evaluate different ranking strategies.

Web Conventions. Most creators of web pages seem to follow simple “rules” without anybody imposing these rules on them. For example, they use the *anchor text* of a link to provide a succinct description of the target page. Since most authors behave this way, we will refer to these rules as *web conventions*, even though there has been no formalization or standardization of such rules.

Search engines rely on these web conventions to improve the quality of their results. Consequently, when webmasters violate these conventions they can confuse search engines. The main issue here is to identify the various conventions that have evolved organically and to develop techniques for accurately determining when the conventions are being violated.

Duplicate Hosts. Web search engines try to avoid crawling and indexing duplicate and near-duplicate pages, as they do not add new information to the search results and clutter up the results. The problem of identifying duplicates within a set of crawled pages is well studied. However, if a search engine can avoid crawling the duplicate content in the first place, the gain is even larger. In general, predicting whether a page will end up being a duplicate of an already-crawled page is chancy work, but the problem becomes more tractable if we limit it to finding duplicate *hosts*, that is, two hostnames that serve the same content. One of the ways that duplicate hosts can arise is via an artifact of the domain name system (DNS) where two hostnames can resolve to the same physical machine. There has only been some preliminary work on the duplicate hosts problem [Bharat *et al.*, 2000].

Vaguely-Structured Data. The degree of structure present in data has had a strong influence on techniques used for search and retrieval. At one extreme, the database community has focused on highly-structured, relational data, while at the other the information retrieval community has been more concerned with essentially unstructured text documents. Of late, there has been some movement toward the middle with the database literature considering the imposition of structure over almost-structured data. In a similar vein, document management systems use accumulated meta-information to introduce more structure. The emergence of XML has led to a flurry of research involving extraction, imposition, or maintenance of partially-structured data.

Web pages in HTML fall into the middle of this continuum of structure in documents, being neither close to free text nor to well-structured data. Instead HTML markup provides limited structural information, typically used to control layout but providing clues about semantic information. Layout information in HTML may seem of limited utility, especially compared to information contained in languages like XML that can be used to tag content, but in fact it is a particularly valuable source of meta-data in unreliable corpora such as the web. The value in layout information stems from the fact that it is visible to the user: Most meta-data which is not user-visible and therefore is particularly susceptible to spam techniques, but layout information is more difficult to use for spam without affecting the user experience. There has only been some initial, partly related work in this vein [Nestorov *et al.*, 1998; Chakrabarti *et al.*, 2001; Chakrabarti, 2001]. We believe that the exploitation of layout information can lead to direct and dramatic improvement in web search results.

2 Spam

Some web authors try to deliberately manipulate their placement in the rankings of various search engine. The resulting pages are called *spam*. Traditional information retrieval collections did not contain spam. As a result, there has not been much research into making search algorithms resistant to spam techniques. Web search engines, on the other hand, have been consistently developing and improving techniques for detecting and fighting spam. As search engine techniques have developed, new spam techniques have developed in response. Search engines do not publish their anti-spam techniques to avoid helping spammers to circumvent them.

Historical trends indicate that the use and variety of spam will continue to increase. There are challenging research issues involved in both detecting spam and in developing ranking algorithms that are resistant to spam. Current spam falls into following three broad categories: text spam, link spam, and cloaking. A spammer might use one or some combination of them.

2.1 Text Spam

All search engines evaluate the content of a document to determine its ranking for a search query. Text spam techniques are used to modify the text in such a way that the search engine rates the page as being particularly relevant, even though the modifications do not increase perceived relevance to a human reader of a document.

There are two ways to try to improve ranking. One is to concentrate on a small set of keywords and try to improve perceived relevance for that set of keywords. For instance, the document author might repeat those keywords often at the bottom of the document, which it is hoped will not disturb the user. Sometimes the text is presented in small type, or even rendered invisible (e.g., by being written in the page’s background color) to accomplish this.

Another technique is to try to increase the number of keywords for which the document is perceived relevant by a search engine. A naïve approach is to include (some subset

of) a dictionary at the bottom of the web page, to increase the chances that the page is returned for obscure queries. A less naïve approach is to add text on a different topic to the page to make it appear that this is the main topic of the page. For example, porn sites sometimes add the names of famous personalities to their pages in order to make these pages appear when a user searches for such personalities.

2.2 Link Spam

The advent of link analysis by search engines has been accompanied by an effort by spammers to manipulate link analysis systems. A common approach is for an author to put a *link farm* at the bottom of every page in a site, where a link farm is a collection of links that points to every other page in that site, or indeed to any site the author controls. The goal is to manipulate systems that use raw counts of incoming links to determine a web page's importance. Since a completely-linked link farm is easy to spot, more sophisticated techniques like pseudo web-rings and random linkage within a member group are now being used.

A problem with link farms is that they distract the reader because they are on pages that also have legitimate content. A more sophisticated form of link farms has been developed, called *doorway pages*. Doorway pages are web pages that consist entirely of links. They are not intended to be viewed by humans; rather, they are constructed in a way that makes it very likely that search engines will discover them. Doorway pages often have thousands of links, often including multiple links to the same page. (There is no text-spam equivalent of doorway pages because text, unlike links, is analyzed by search engines on a per-page basis.)

Both link farms and doorway pages are most effective when the link analysis is sensitive to the absolute number of links. Techniques that concentrate instead on the quality of links, such as PageRank [Brin and Page, 1998; Brin *et al.*, 1998], are not particularly vulnerable to these techniques.

2.3 Cloaking

Cloaking involves serving entirely different content to a search engine crawler than to other users.¹ As a result, the search engine is deceived as to the content of the page and scores the page in ways that, to a human observer, seem rather arbitrary.

Sometimes cloaking is used with the intent to “help” search engines, for instance by giving them an easily digestible, text-only version of a page that is otherwise heavy with multimedia content, or to provide link-based access to a database which is normally only accessible via forms (which search engines cannot yet navigate). Typically, however, cloaking is used to deceive search engines, allowing the author to achieve

¹A search engine crawler is a program that downloads web pages for the purpose of including them in the search engine results. Typically a search engine will download a number of pages using the crawler, then process the pages to create the data structures used to service search requests. These two steps are repeated continuously to ensure the search engine is searching over the most up-to-date content possible.

the benefits of link and text spam without inconveniencing human readers of the web page.

2.4 Defending against Spam

In general, text spam is defended against in a heuristic fashion. For instance, it was once common for sites to “hide” text by writing it in white text on a white background, ensuring that human readers were not affected while search engines were misled about the content. As a result, search engine companies detected such text and ignored it. Such reactive approaches are, obviously, not optimal. Can pro-active approaches succeed? Perhaps these approaches could be combined; it might be possible for the search engine to notice what pages change in response to the launch of a new anti-spam heuristic, and to consider those pages as potential spam pages.

Typically, link-spam sites have certain patterns of links that are easy to detect, but these patterns can mutate in much the same way as link spam detection techniques. A less heuristic approach to discovering link spam is required. One possibility is, as in the case of text spam, to use a more global analysis of the web instead of merely local page-level or site-level analysis. For example, a cluster of sites that suddenly sprout thousands of new and interlinked webpages is a candidate link-spam site. The work by [Kumar *et al.*, 1999] on finding small bipartite clusters in the web is a first step in this direction.

Cloaking can only be discovered by crawling a website twice, once using an HTTP client the cloaker believes is a search engine, and once from a client the cloaker believes is not a search engine. Even this is not good enough, since web pages typically differ between downloads for legitimate reasons, such as changing news headlines.

An interesting challenge is to build a spam classifier that reliably detects a large fraction of the currently existing spam categories.

3 Content Quality

While spams are attempts to deliberately mislead search engines, the web is replete with text that — intentionally or not — misleads its human readers as well. As an example, there is a webpage which claims (falsely!) that Thomas Jefferson was the first president of the United States. Many websites, purposefully or not, contain misleading medical information.² Other sites contain information that was once correct but is now out of date; for example, sites giving names of elected officials.

While there has been a great deal of research on determining the relevance of documents, the issue of document quality or accuracy has not been received much attention, whether in web search or other forms of information retrieval. For instance, the TREC conference explicitly states rules for when it considers a document to be relevant, but does not mention the accuracy or reliability of the document at all. This is understandable, since typical research corpora, including the

²One study showed many reputable medical sites contain contradictory information on different pages of their site [Berland *et al.*, 2001] — a particularly difficult content-quality problem!

ones used by TREC and found in corporate intranets, consist of document sources that are deemed both reliable and authoritative. The web, of course, is not such a corpus, so techniques for judging document quality is essential for generating good search results. Perhaps the one successful approach to (heuristically) approximating quality on the web is based on link analysis, for instance PageRank [Brin and Page, 1998; Brin *et al.*, 1998] and HITS [Kleinberg, 1998]. These techniques are a good start and work well in practice, but there is still ample room for improvement.

One interesting aspect of the problem of document quality is specific to hypertext corpora such as the web: evaluating the quality of *anchor text*. Anchor text is the text, typically displayed underlined and in blue by the web browser, that is used to annotate a hypertext link. Typically, web-based search engines benefit from including anchor-text analysis in their scoring function [Craswell *et al.*, 2001]. However, there has been little research into the perils of anchor-text analysis e.g. due to spam and on methodologies for avoiding the pitfalls.

For instance, for what kinds of low-quality pages might the anchor text still be of high quality? Is it possible to judge the quality of anchor text independently of the quality of the rest of the page? Is it possible to detect anchor text that is intended to be editorial rather than purely descriptive? In addition, many fundamental issues remain open in the application of anchor text to determination of document quality and content. In case of documents with multiple topics, can anchor text analysis be used to identify the themes?

Another promising area of research is to combine established link-analysis quality judgments with text-based judgments. A text-based analysis, for instance, could judge the quality of the Thomas Jefferson page by noting that most references to the first president of the United States in the web corpus attribute the role to George Washington.

4 Quality Evaluation

Search engines cannot easily improve their ranking algorithms without running tests to compare the quality of the new ranking technique with the old. Performing such comparisons with human evaluators is quite work-intensive and runs the danger of not correctly reflecting user needs. Thus, it would be best to have end users perform the evaluation task, as they know their own needs the best.

Users, typically, are very reluctant to give direct feedback. However, web search engines can collect implicit user feedback using log data such as the position of clicks for a search and the time spent on each click. This data is still incomplete. For instance, once the user clicks on a search result, the search engine does not know which pages the user visits until the user returns to the search engine. Also, it is hard to tell whether a user clicking on a page actually ends up finding that page relevant or useful.

Given the incomplete nature of the information, the experimental setup used to collect implicit user data becomes particularly important. That is: How should click-through and other data be collected? What metrics should be computed from the data?

One approach is to simply collect the click-through data from a subset of the users — or all users — for two ranking algorithms. The experimenter can then compute metrics such as the percentage of clicks on the top 5 results and the number of clicks per search.

Recently, Joachims [2002] suggested another experimental technique which involves merging the results of the two ranking algorithms into a single result set. In this way each user performs a comparison of the two algorithms. Joachims proposes to use the number of clicks as quality metric and shows that, under some weak assumptions, the clickthrough for ranking A is higher than the clickthrough for B if and only if A retrieves more relevant links than B.

5 Web Conventions

As the web has grown and developed, there has been an evolution of conventions for authoring web pages. Search engines assume adherence to these conventions to improve search results. In particular, there are three conventions that are assumed relating to anchor text, hyperlinks, and META tags.

- As discussed in Section 3, the fact that anchor text is meant to be descriptive is a web convention, and this can be exploited in the scoring function of a search engine.
- Search engines typically assume that if a web page author includes a link to another page, it is because the author believes that readers of the source page will find the destination page interesting and relevant. Because of the way people usually construct web pages, this assumption is usually valid. However, there are prominent exceptions: for instance, link exchange programs, in which web page authors agree to reciprocally link in order to improve their connectivity and rankings, and advertisement links. Humans are adept at distinguishing links included primarily for commercial purposes from those included primarily for editorial purposes. Search engines are less so.

To further complicate matters, the utility of a link is not a binary function. For instance, many pages have links allowing you to download the latest version of Adobe's Acrobat Reader. For visitors that do not have Acrobat Reader, this link is indeed useful, certainly more useful than for those who have already downloaded the program. Similarly, most sites have a terms of service link at the bottom of every page. When the user first enters the site, this link might well be very useful, but as the user browses other webpages on the site, the link's usefulness immediately decreases.

- A third web convention concerns the use of META tags. These tags are currently the primary way to include metadata within HTML. In theory META tags can include arbitrary content, but conventions have arisen for meaningful content. A META tag of particular importance to search engines is the so-called *Content META tag*, which web page authors use to describe the content of the document. Convention dictates that the content META tag contains either a short textual summary of the

page or a brief list of keywords pertaining to the content of the page.

Abuse of this META tags is common, but even when there is no attempt to deceive, there are those who break the convention, either out of ignorance or overzealousness. For instance, a webpage author might include a summary of their entire site within the META tag, rather than just the individual page. Or, the author might include keywords that are more general than the page warrants, using a META description of “cars for sale” on a web page that only sells a particular model of car.

In general, the correctness of META tags is difficult for search engines to analyze because they are not visible to users and thus are not constrained to being useful to visitors. However, there are many web page authors that use META tags correctly. Thus, if web search engines could correctly judge the usefulness of the text in a given META tag, the search results could potentially be improved significantly. The same applies to other content not normally displayed, such as ALT text associated with the IMAGE tag.

While link analysis has become increasingly important as a technique for web-based information retrieval, there has not been as much research into the different types of links on the web. Such research might try to distinguish commercial from editorial links, or links that relate to meta-information about the site (“This site best viewed with [start link]browser X[end link]”) from links that relate to the actual content of the site.

To some extent, existing research on link analysis is helpful, since authors of highly visible web pages are less likely to contravene established web conventions. But clearly this is not sufficient. For instance, highly visible pages are more, rather than less, likely to include advertisements than the average page.

Understanding the nature of links is valuable not only for itself, but also because it enables a more sophisticated treatment of the associated anchor text. A potential approach would be to use text analysis of anchor text, perhaps combined with meta-information such as the URL of the link, in conjunction with information obtained from the web graph.

6 Duplicate Hosts

Web search engines try to avoid crawling and indexing duplicate and near-duplicate pages, since such pages increase the time to crawl and do not contribute new information to the search results. The problem of finding duplicate or near-duplicate pages in a set of crawled pages is well studied [Brin *et al.*, 1995; Broder, 1997]. There has also been some research on identifying duplicate or near-duplicate directory trees [Cho *et al.*, 2000], called *mirrors*.

While mirror detection and individual-page detection try to provide a complete solution to the problem of duplicate pages, a simpler variant can reap most of the benefits while requiring less computational resources. This simpler problem is called *duplicate host detection*. Duplicate hosts (“duphosts”) are the single largest source of duplicate pages on the web, so solving the duplicate hosts problem can result in a significantly improved web crawler.

A *host* is merely a name in the domain name system (DNS), and duphosts arise from the fact that two DNS names can resolve to the same IP address.³ Companies typically reserve more than one name in DNS, both to increase visibility and to protect against domain name “squatters.” For instance, currently both `bikesport.com` and `bikesportworld.com` resolve to the same IP address, and as a result the sites `http://www.bikesport.com/` and `http://www.bikesportworld.com/` display identical content.

Unfortunately, duplicate IP addresses are neither necessary nor sufficient to identify duplicate hosts. Virtual hosting can result in different sites sharing an IP address, while round-robin DNS can result in a single site having multiple IP addresses.

Merely looking at the content of a small part of the site, such as the homepage, is equally ineffective. Even if two domain names resolve to the same website, their homepages could be different on the two viewings, if for instance the page includes an advertisement or other dynamic content. On the other hand, there are many unrelated sites on the web that have an identical “under construction” home page.

While there has been some work on the duphosts problem [Bharat *et al.*, 2000], it is by no means a solved problem. One difficulty is that the solution needs to be much less expensive than the brute-force approach that compares every pair of hosts. For instance, one approach might be to download every page on two hosts, and then look for a graph isomorphism. However, this defeats the purpose of the project, which is to not have to download pages from both of two sites that are duphosts.

Furthermore, web crawls are never complete, so any link-structure approach would have to be robust against missing pages. Specifically, a transient network problem, or server downtime, may keep the crawler from crawling a page in one host of a duphost pair, but not the other. Likewise, due to the increasing amount of dynamic content on the web, text-based approaches cannot check for exact duplicates.

On the other hand, the duphosts problem is simpler than the more general problem of detecting mirrors. Duphosts algorithms can take advantage of the fact that the urls between duphosts are very similar, differing only in the hostname component. Furthermore, they need not worry about content reformatting, which is a common problem with mirror sites.

Finally — and this is not a trivial matter — duphost analysis can benefit from semantic knowledge of DNS. For instance, candidate duphosts `http://foo.com` and `http://foo.co.uk` are, all other things being equal, likely to be duphosts, while candidates `http://foo.com` and `http://bar.com` are not as likely to be duphosts.

³In fact, it’s not necessary that they resolve to the same IP address to be duphosts, just that they resolve to the same webserver. Technically even that is not necessary; the minimum requirement is that they resolve to computers that serve the same content for the two hostnames in question.

7 Vaguely-Structured Data

While information retrieval corpora has tended to be very low in structure, database content is very well structured. This has obviously led to a major difference in how the two fields have evolved over the years. For instance, a practical consequence of this difference is that databases permit a much richer and complex set of queries, while text-based query languages are in general much more restricted.

As database content, or more generally structured data, started being exposed through web interfaces, there developed a third class of data called *semi-structured data*. In the web context, semi-structured data is typically the content of a webpage, or part of a webpage, that contains structured data but no longer contains unambiguous markup explicating the structure or schema. There has been considerable research on recovering the full structure of semi-structured data, for example, [Ahonen *et al.*, 1994] and [Nestorov *et al.*, 1998].

The three examples above cover three points on the continuum of structured data. However, most web pages do not fall into any of these categories, but instead fall into a fourth category we call *vaguely-structured data*. The information on these web page is not structured in a database sense — typically it's much closer to prose than to data — but it does have some structure, often unintentional, exhibited through the use of HTML markup.

We say that HTML markup provides unintentional structure because it is not typically the intent of the webpage author to describe the document's semantics. Rather, the author uses HTML to control the document's layout, the way the document appears to readers. (It is interesting to note that this subverts the original purpose of HTML, which was meant to be a document description language rather than a page description language.) To give one example, HTML has a tag that is intended to be used to mark up glossary entries. In common browsers, this caused the text to be indented in a particular way, and now the glossary tag is used in any context where the author wants text indented in that manner. Only rarely does this context involve an actual glossary.

Of course, often markup serves both a layout and a semantic purpose. The HTML header tags, for instance, produce large-font, bold text useful for breaking up text, but at the same time they indicate that the text so marked is probably a summary or description of the smaller-font text which follows.

Even when markup provides no reliable semantic information, it can prove valuable to a search engine. To give just one example, users have grown accustomed to ignoring text on the periphery of a web page [Faraday, 2001], which in many cases consists of navigational elements or advertisements. Search engines could use positional information, as expressed in the layout code, to adjust the weight given to various sections of text in the document.

In addition, layout can be used to classify pages. For instance, pages with an image in the upper-left of the page are often personal homepages. Pages with a regular markup structure are likely to be lists, which search engines may wish to analyze differently than pages with running text.

Markup can be meta-analyzed as well. It is plausible that

pages with many mistakes in the markup are more likely to be of lower quality than pages with no mistakes. Patterns in the markup used may allow a search engine to identify the web authoring tool used to create the page, which in turn might be useful for recovering some amount of structure from the page. Markup might be particularly useful for clustering web pages by author, as authors often use the same template for most of the pages they write.

And, of course, HTML tags can be analyzed for what semantic information can be inferred. In addition to the header tags mentioned above, there are tags that control the font face (bold, italic), size, and color. These can be analyzed to determine which words in the document the author thinks are particularly important.

One advantage of HTML, or any markup language that maps very closely to how the content is displayed, is that there is less opportunity for abuse: it is difficult to use HTML markup in a way that encourages search engines to think the marked text is important, while to users it appears unimportant. For instance, the fixed meaning of the <H1> tag means that any text in an H1 context will appear prominently on the rendered web page, so it is safe for search engines to weigh this text highly. However, the reliability of HTML markup is decreased by Cascading Style Sheets [World Wide Web Consortium,], which separate the names of tags from their representation.

There has been research in extracting information from what structure HTML does possess. For instance, [Chakrabarti *et al.*, 2001; Chakrabarti, 2001] created a DOM tree of an HTML page and used this information to increase the accuracy of topic distillation, a link-based analysis technique.

However, there has been less research addressing the fact HTML markup is primarily descriptive, that is, that it is usually inserted to affect the way a document appears to a viewer. Such research could benefit from studies of human perception: how people view changes in font size and face as affecting the perceived importance of text, how much more likely people are to pay attention to text at the top of a page than the bottom, and so forth. As newspaper publishers have long known, layout conveys semantic information, but it's not trivial to extract it.

Turning HTML into its markup is also a challenge. It is possible to render the page, of course, but this is computationally expensive. Is there any way to figure out, say, if a given piece of HTML text is in the "middle" of a rendered HTML page without actually rendering it?

Of course, HTML text is only one example of vaguely structured data. What other kinds of content exists that is somewhere between unstructured data and semi-structured data in terms of quantity of annotation? How does it differ from HTML text? For that matter, the continuum of structure is not well-mapped. What techniques appropriate for unstructured data work equally well with vaguely structured data? What techniques work for semi-structured data? How can these techniques be improved as data gets more structured, and is there any way to map the improvements down to less structured forms of data (perhaps by imputing something "structural" to the data, even if that doesn't correspond to any

intuitive idea of structure)?

8 Conclusions

In this paper we presented some challenging problems faced by current web search engines. There are other fruitful areas of research related to web search engines we did not touch on. For instance, there are challenging systems issues that arise when hundreds of millions of queries over billions of web pages have to be serviced every day without any downtime and as inexpensively as possible. Furthermore, there are interesting user interface issues: What user interface does not confuse novice users, does not clutter the screen, but still fully empowers the experienced user? Finally, are there other ways to mine the collection of web pages so as to provide a useful service to the public at large?

9 Resources

Here are two resources for the research community:

Stanford's WebBase project (<http://www-diglib.stanford.edu/~testbed/doc2/WebBase/>) distributes its content of web pages.

Web term document frequency is available at Berkeley's Web Term Document Frequency and Rank site (<http://elib.cs.berkeley.edu/docfreq/>).

References

- [Ahonen *et al.*, 1994] H. Ahonen, H. Mannila, and E. Nikunen. "Generating grammars for SGML tagged texts lacking DTD." *PODP'94 – Workshop on Principles of Document Processing*, 1994.
- [Berland *et al.*, 2001] G. K. Berland, M. N. Elliott, L. S. Morales, J. I. Algazy, R. L. Kravitz, M. S. Broder, D. E. Kanouse, J. A. Muñoz, J.-A. Puyol, M. Lara, K. E. Watkins, H. Yang, and E. A. McGlynn. "Health Information on the Internet Accessibility, Quality, and Readability in English and Spanish." *Journal of the American Medical Association*, 285(2001):2612–2621.
- [Bharat *et al.*, 2000] K. Bharat, A. Z. Broder, J. Dean, and M. Henzinger. "A comparison of Techniques to Find Mirrored Hosts on the World Wide Web." *Journal of the American Society for Information Science*, 31(2000):1114–1122.
- [Brin *et al.*, 1995] S. Brin, J. Davis, and H. García-Molina. "Copy detection mechanisms for digital documents." *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1995, pages 398–409.
- [Brin and Page, 1998] S. Brin and L. Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." In *Proceedings of the 7th International World Wide Web Conference (WWW7)*, 1998, pages 107–117. Also appeared in *Computer Networks* 30(1998): 107–117.
- [Brin *et al.*, 1998] S. Brin, L. Page, R. Motwani, and T. Winograd. "What can you do with a Web in your Pocket?" *Bulletin of the Technical Committee on Data Engineering*, 21(1998):37–47.
- [Broder, 1997] A. Z. Broder. "On the resemblance and containment of documents." In *Proceedings of Compression and Complexity of Sequences*, IEEE Computer Society, 1997, pages 21–29.
- [Chakrabarti *et al.*, 2001] S. Chakrabarti, M. Joshi, and V. Tawde. Enhanced topic distillation using text, markup tags, and hyperlinks. In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001, pages 208–216.
- [Chakrabarti, 2001] S. Chakrabarti. Integrating the Document Object Model with hyperlinks for enhanced topic distillation and information extraction. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*, 2001, pages 211–220.
- [Cho *et al.*, 2000] J. Cho, N. Shivakumar, and H. Garcia-Molina. "Finding replicated web collections." In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2000, pages 355–366.
- [Craswell *et al.*, 2001] N. Craswell, D. Hawking, and S. Robertson. "Effective Site Finding using Link Anchor Information." In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001, pages 250–257.
- [Faraday, 2001] P. Faraday. "Attending to Web Pages." *CHI 2001 Extended Abstracts (Poster)*, 2001, pages 159–160.
- [Henzinger *et al.*, 2002] M. R. Henzinger, R. Motwani, and C. Silverstein. "Challenges in Web Search Engines." In *ACM SIGIR Forum*, 36(2):11–23, 2002.
- [Kleinberg, 1998] J. Kleinberg. "Authoritative sources in a hyperlinked environment." In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998, pages 668–677.
- [Kumar *et al.*, 1999] S. R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins. "Trawling emerging cyber-communities automatically." In *Proceedings of the 8th International World Wide Web Conference (WWW8)*, 1999, pages 1481–1493.
- [Joachims, 2002] T. Joachims. "Evaluating Search Engines using Clickthrough Data". Under submission, 2002.
- [Nestorov *et al.*, 1998] S. Nestorov, S. Abiteboul, and R. Motwani. "Extracting Schema from Semistructured Data." In *Proceedings of the ACM SIGMOD Conference on Management of Data*, 1998, pages 295–306.
- [Silverstein *et al.*, 1999] C. Silverstein, M. R. Henzinger, J. Marais, and M. Moricz. "Analysis of a very large AltaVista query log." *ACM SIGIR Forum*, 33(1999):6–12.
- [World Wide Web Consortium,] World Wide Web Consortium. "Web Style Sheets." <http://www.w3.org/Style/>.

Deploying Information Agents on the Web

Craig A. Knoblock

University of Southern California
Information Sciences Institute and Computer Science Department
4676 Admiralty Way
Marina del Rey, CA 90292, USA
knoblock@isi.edu

Abstract

The information resources on the Web are vast, but much of the Web is based on a browsing paradigm that requires someone to actively seek information. Instead, one would like to have information agents that continuously attend to one's personal information needs. Such agents need to be able to extract the relevant information from web sources, integrate data across sites, and execute efficiently in a networked environment. In this paper I describe the technologies we have developed to rapidly construct and deploy information agents on the Web. This includes wrapper learning to convert online sources into agent-friendly resources, query planning and record linkage to integrate data across different sites, and streaming dataflow execution to efficiently execute agent plans. I also describe how we applied this work within the Electric Elves project to deploy a set of agents for continuous monitoring of travel itineraries.

1 Introduction

There is a tremendous amount of information available on the Web, but the access to this information is largely dependent on the information providers. The individual web sites decide what information to provide to the visitors to their site and how a visitor will access that information. There is little in the way of capabilities to combine information across sites and only the most primitive capabilities to monitor information on individual sites. Of course, sites may provide integrated access to specific data sources or sophisticated monitoring capabilities on their data, but users of the Web are dependent on a site to make these capabilities available.

In contrast, imagine a world where access to information is driven by the consumers of the data. Where it is not only possible, but simple to task your own personal information agents to gather specific information, monitor for changes in the gathered data, and to notify you of important information or changes. The challenges are that the required information is often embedded in HTML pages, the data is organized in different ways on different sites, there are a huge number of ways the information could be combined, and it can be slow and cumbersome to combine the information.

We have developed a set of core technologies to simplify the task of building intelligent agents for gathering and monitoring information on the Web [Knoblock *et al.*, 2003; 2001; Barish and Knoblock, 2002a]. The technologies include the ability to gather information from the Web, to link that information with related information, to build plans to integrate the various sources of data, and to efficiently execute these plans in the Web environment. We have applied these technologies to build agents for a variety of applications [Barish *et al.*, 2000; Ambite *et al.*, 2002], including an application for monitoring travel plans from online sources.

Researchers have developed a variety of agents that have been deployed on the Web over the years. Some notable systems include the Internet Softbot [Etzioni and Weld, 1994], an agent that interacts with a range of Internet resources, BargainFinder [Krulwich, 1996], a comparison shopping agent for CDs, ShopBot [Perkowitz *et al.*, 1997], a system for automatically locating and incorporating new stores into a comparison shopping agent, and Warren [Decker *et al.*, 1997], a system for gathering data on a financial portfolio. There has also been a significant amount of research on the underlying technologies required for developing agents on the Internet [Levy and Weld, 2000]. As noted by Etzioni [1996], the Web provides a tremendous opportunity for building intelligent software agents. Yet, surprisingly few have seized this opportunity. This is almost certainly because there are many technical issues that must be addressed to build such agents. Our work is designed to address these issues and simplify the task of building agents for the Web.

In this paper I first describe an example set of agents for monitoring travel plans. Then I will briefly describe the technologies that we have developed to gather data from web sources, link data across sources, generate plans to integrate the data, and efficiently execute these plans. Finally, I present directions for future research and conclusions.

2 Information Agents for Monitoring Travel

As part of the Electric Elves project [Chalupsky *et al.*, 2001; Ambite *et al.*, 2002] we have applied our agent technologies to build a set of agents for various tasks including tracking visitor schedules, monitoring meeting schedules, and monitoring a user's travel plans. In the case of monitoring travel plans, this task is particularly well-suited for applying agent technology for several reasons: a) this is a fairly compli-

cated task with many possible forms of failure ranging from flight cancellations and schedule changes to hotel rooms being given away when a traveler arrives late at night, b) there are a large number of online resources that can be exploited to anticipate problems and keep a traveler informed, and c) these tasks would be tedious and impractical for a human to perform with the same level of attention that could be provided by a set of software agents.

To deploy a set of agents for monitoring a planned trip, the user first enters the travel itinerary and then specifies which aspects of the trip she would like to have the agents monitor. A set of information agents are then spawned to perform the requested monitoring activities. For the travel planning application, we developed the following set of agents to monitor a trip:

- A *airfare-monitoring agent* that tracks the current price of a flight itinerary. The agent sends a notification on price increases and/or decreases. A traveler might consider reticketing if the price drops significantly below what they paid.
- A *schedule-change agent* that keeps track of the published schedule for a given flight itinerary and notifies a traveler of any change to this itinerary. Small schedule changes occur quite frequently, especially if one purchases tickets well in advance of a flight. Travel agents are supposed to notify their customers of changes, but one usually arrives at the airport before discovering that the scheduled departure time has changed.
- A *flight-status agent* that continually monitors the status of a flight. When a change of status or cancellation is detected, the traveler is immediately notified. This agent also sends a fax to the hotel if the flight arrival is delayed past 5pm in order to ensure that the hotel room is held for the traveler. This agent differs from what is available from commercial sites, such as ual.com, which simply check the status a fixed period of time prior to the flight. In contrast, our agents maintain state and can notify the user of multiple status changes without sending messages that provide no new information.
- An *earlier-flight agent* that checks for flights that leave before the scheduled flight. It also checks the status of these flights to avoid suggesting delayed or cancelled flights. This agent is particularly handy when one finishes a meeting early or one wants to skip out of a particularly boring meeting.
- A *flight-connection agent* that monitors a users scheduled flights and if there is a connecting flight it wakes up a few minutes before the projected landing time, checks the status and gate information of the connecting flight and also searches for any earlier flights to the same destination that the user might be able to take instead. This agent is particularly useful when there are only a few minutes to make an earlier connection that is about to depart.
- A *restaurant-finding agent* that locates the nearest restaurant based on the user's GPS location. On request, it suggests the five closest restaurants providing cuisine type,

price, address, phone number, latitude, longitude, and distance from the user's location.

These agents are scheduled to run at regular intervals, where the agents are woken up to perform their task. The agents can cancel their own task once it is complete and can change the interval in which they are run based on the information from other agents. The agents often invoke other agents to help them perform their tasks. For example, the flight-status agent calls another agent that extracts the flight status information directly from a web site and it invokes the hotel notification agent, which in turn sends a message to the fax agent.

Figure 1 shows the messages that various agents generated during actual use of the system. The original set of agents were in use for about a year and then based on feedback and requests from the users we recently developed a new set of agents that provide improved capabilities.

(a) *Airfare-Monitoring Agent*: Airfare dropped message
The airfare for your American Airlines itinerary (IAD - LAX) dropped to \$281.

(b) *Schedule-Change Agent*:
The schedule of your United Airlines flight 1287 has changed from 7:00 PM to 7:31 PM.

(c) *Flight-Status Agent*: Flight delayed message
Your United Airlines flight 190 has been delayed. It was originally scheduled to depart at 11:45 AM and is now scheduled to depart at 12:30 PM. The new arrival time is 7:59 PM.

(d) *Flight-Status Agent*: Flight cancelled message
Your Delta Air Lines flight 200 has been cancelled.

(e) *Flight-Status Agent*: Fax to a hotel message
Attention : Registration Desk
I am sending this message on behalf of David Pynadath, who has a reservation at your hotel. David Pynadath is on United Airlines 190, which is now scheduled to arrive at IAD at 7:59 PM. Since the flight will be arriving late, I would like to request that you indicate this in the reservation so that the room is not given away.

(f) *Earlier-Flight Agent*:
The status of your currently scheduled flight is:
190 LAX (11:45 AM) - IAD (7:29 PM) 45 minutes Late
The following United Airlines flight arrives earlier than your flight:
946 LAX (8:31 AM) - IAD (3:35 PM) 11 minutes Late

(g) *Flight-Connection Agent*:
Your connecting United Airlines flight 925 will depart at 9:45 PM (25 minutes late) at gate C6.

(h) *Restaurant-Finding Agent*:
These are the five closest restaurants from your location.
Wingmaster's on I St, American, 1825 I St NW, 202-429-0058, \$5-10, Lat: 38.90111, Lon: -77.04158, 0.23 miles
...

Figure 1: Actual messages sent by monitoring agents

3 Gathering Data from Web Sources

A key capability for information agents is the ability to reliably access information. As the Web moves towards XML and Web Services, accessing data could become greatly simplified. However, movement in this direction has been quite

slow and for various reasons many sources will remain available only in HTML, so there is still a critical need to turn HTML sources into agent-enabled sources.

The challenge in building wrappers for online sources is how to achieve broad coverage and high accuracy with minimal user input. The two general approaches to this problem are supervised machine learning techniques [Kushmerick, 1997; Muslea *et al.*, 2001; Hsu and Dung, 1998] and unsupervised grammar induction techniques [Lerman *et al.*, 2001; Crescenzi *et al.*, 2001; Hong and Clark, 2001]. The unsupervised grammar induction techniques have the advantage of no user input, but they are not able to handle the full range of semistructured sources. In contrast, the supervised learning techniques apply to a wider set of sites, but can require a significant amount of labeled data to achieve high accuracy.

We developed an machine learning algorithm called Stalker [Muslea *et al.*, 2001] that requires labeled data, but attempts to minimize the amount of information that must be provided by a user. Given labeled data, the system employs a greedy set covering algorithm to learn extraction rules that define a wrapper for a source. We minimize the amount of labeled data required by decomposing the learning problem into a number of simpler subproblems, which require fewer examples to learn. The decomposition is based on the hierarchical structure of the information in a web source. This approach allows Stalker to learn how to extract data from complicated sites that involve lists of information and even arbitrary nesting of embedded lists.

An issue for any learning system, even Stalker, is that to achieve high accuracy the system must see the right set of examples. Since the expectation in a wrapper is to extract the data with 100% accuracy, finding a representative set of examples is a critical part of the problem. Rather than relying on the user to identify these examples, we developed an active learning technique called Co-Testing [Muslea *et al.*, 2000; Muslea, 2002; Muslea *et al.*, 2003] that selects the most information examples to label. Co-Testing works by learning multiple classifiers using different views of the same problem. In the case of wrapper learning, the system exploits the fact that it can learn equally well a classifier by finding landmarks from the beginning of the page or by finding landmarks from the end of the page. The system can then exploit the fact that both classifiers should agree if they have learned the same concept and any disagreement provides a source of training examples. Both classifiers are applied to the unlabeled examples and the user is asked to label the examples where there is disagreement. This allows the system to quickly identify the unusual cases in the data set to rapidly converge on an accurate set of extraction rules.

Another important challenge building wrappers is to ensure that they continue to work properly over time. This problem has not received much attention. The exception is the work by Kushmerick [2000] who developed an approach that uses the global properties of a page, such as the density of HTML tokens on a page, to determine when the page or even the specific information being extracted has changed. The limitation of this approach is that it is too coarse to detect some sites that have changed [Lerman *et al.*, 2003].

We developed a wrapper maintenance system that can re-

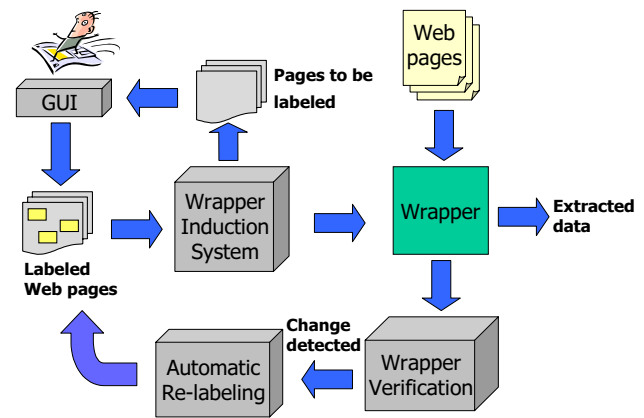


Figure 2: Wrapper induction, verification, and reinduction process

pair wrappers by learning a description of the content extracted by a wrapper [Lerman *et al.*, 2003]. This approach learns a pattern by using a hierarchy of pattern types, such as number or capitalized word, and then learning a description of the beginning and ending of the information that is being extracted. The resulting description or learned patterns are then stored and compared to the information being extracted. The patterns are compared statistically to avoid false positives due to examples that have not been seen before. In a large test set, this approach was able to identify 95% of the Web sites that had changed.

Once the system has identified a source that has changed, the learned patterns can then be used to automatically relabel the site and run the labeled examples through Stalker, the wrapper learning system. The wrapper learning, validation, and reinduction process are illustrated in Figure 2.

4 Linking Information

Once data has been extracted from a web site, then one frequently needs to combine it with other information in order to perform some task. For example, if one wants to build an agent for recommending nearby restaurants, then one might want to combine the data on a restaurant review site with the department of health site to ensure that the restaurant has an adequate health rating. The problem is that the information on these two distinct sites will often refer to the restaurants in different ways – the name, address, and phone number may all have slight variations that preclude simply joining the two sites across a single attribute.

To address this problem, we developed a machine learning approach to record linkage [Tejada *et al.*, 2002; 2001]. In this approach the system, called ActiveAtlas, compares the attributes that are shared across the two data sets and learns two things. First, it uses a committee of decision tree learners to decide whether two records are matched based on the strength of the match of the various attributes (Figure 3). Second, it improves the accuracy of these matches by learning an appropriate set of weights on a library of transformation rules (Figure 4). ActiveAtlas takes an active learning approach to

	Name	Street	Phone
Zagat's	Art's Deli	12224 Ventura Boulevard.	818-756-4124
Dept of Health	Art's Delicatessen	12224 Ventura Blvd.	818/755-4100

Figure 3: ActiveAtlas learns which attributes are important to decide whether two records should be linked

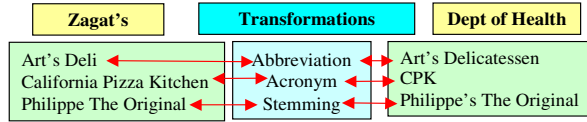


Figure 4: ActiveAtlas also learns the weighting of the transformation rules for an application

select examples for the user to label in order maximize the accuracy of the matches and minimize the amount of user input. Compared to other approaches to this problem [Cohen, 2000; Sarawagi and Bhamidipaty, 2002], the combination of the transformation weight learning and the active learning of the rules allows the system to achieve very high accuracy in matching entities.

5 Planning to Integrate Sources

Once an agent has access to the sources and can link the data across sources, the problem remains how to compose a set of information sources to perform some task. We developed an approach to automatically planning the integration of these sources to answer queries. In this approach, which is implemented in the Ariadne information mediator [Knoblock *et al.*, 1998; 2001], the contents of the sources available to the system are described using a common model [Ambite *et al.*, 2001]. The system uses this model to create a plan that specifies both the data sources and the specific operations to be performed to satisfy a request.

One of the interesting problems is that due to the large search space and the need to optimize the plans, traditional planning techniques do not scale. To overcome this problem we developed a general-purpose planning approach, called Planning by Rewriting [Ambite and Knoblock, 2001], and we use it as the planner for Ariadne [Ambite and Knoblock, 2000]. In Planning by Rewriting, the system starts with an initial, but suboptimal plan, and then the planner performs a local search through the space of plan transformations to maximize the given evaluation criterion. In the query planning application, the planner searches through the space of sources and the operations on these sources to find an efficient way to process a query. Figure 5 shows a simple example of how Planning by Rewriting searches through the space of plan transformations. Researchers have explored a variety of approaches to this general problem of planning for information gathering (see [Lambrecht and Kambhampati, 1997] for a summary of this work).

We are currently exploring planning techniques for composing Web Services [Thakkar *et al.*, 2003]. The Web Ser-

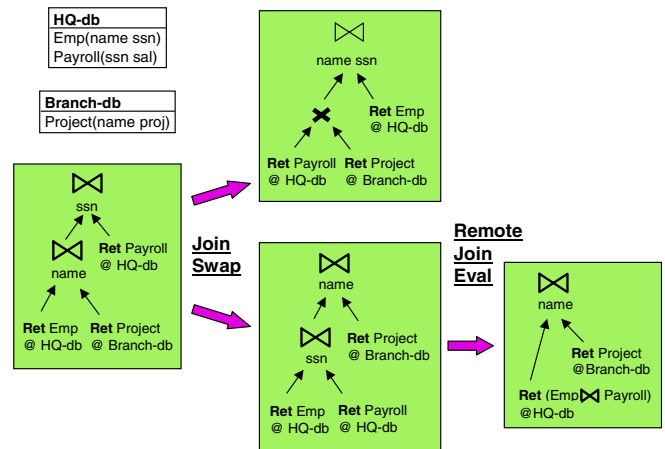


Figure 5: Planning by Rewriting searches through the space of possible transformations on a plan

vice infrastructure provides access to online sources in a form with which agents can interact without having to construct a wrapper for a site. In addition, the Semantic Web provides semantic-level descriptions of the services that are available [Ankolenkar *et al.*, 2002]. Our approach to integration planning builds on previous work on data integration [Levy, 2000] and applies the inverse rules approach of Duschka [1997].

One challenge is that the sources available online often have restrictions on how they can be accessed in that certain inputs may be required to access a source. In the inverse rules framework this means that computing a complete set of answers to a query may require recursion. Since it can be expensive to execute integration plans in a language such as Datalog, we have developed an approach to automatically convert the recursive plans produced by the inverse rules algorithm into a streaming dataflow execution framework that can efficiently execute these plans [Thakkar and Knoblock, 2003]. This execution framework is described next.

6 Executing Plans

Given a plan for performing some task on the Web, an agent needs to be able to efficiently execute this plan. In the Web environment, sources can be quite slow and the latencies of the sources are also unpredictable since they can be caused by heavy loads on both servers and networks. Since the primary bottleneck of most agent plans on the web is retrieving data from online sources, we would like to execute information requests as early as possible. To address these issues, we have developed a streaming dataflow language and executor, called Theseus [Barish and Knoblock, 2002a], which is optimized for the Web environment in the following three ways. First, since the executor is based on a dataflow paradigm, actions are executed as soon as the data becomes available. Second, Theseus performs the actions in a plan in separate threads, so they can be run asynchronously and in parallel. Third, the system streams the output from one action to the next so that sequential operations can be executed in parallel.

Theseus is similar to network query engines, such as Telegraph [Hellerstein *et al.*, 2000] or Tukwila [Ives *et al.*, 2002],

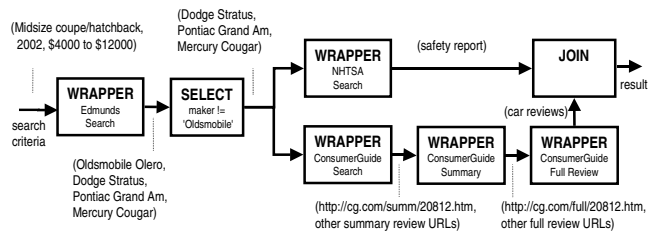


Figure 6: Example plan for integrating data from three car-related sites

in that they are also streaming dataflow execution systems. However, the network query engines focus on the efficient execution of XML queries, while Thesues provides an expressive language for expressing information gathering and monitoring plans. The Theseus language supports capabilities that go beyond network query engines in that it supports recursion, notification operations, and writing and reading from databases to support monitoring tasks.

Recently we developed an approach to increase the potential parallelism in a streaming dataflow execution system. This optimization technique, called speculative execution [Barish and Knoblock, 2002b; 2003], attempts to predict the results of an operation based on data and patterns that it has seen in the past. The predicted results can then be used to speculate about the operations that will need to be performed later in the plan. The system decides where to speculate by analyzing a plan and determining the critical paths. On these paths it then inserts a “speculate” operation, which uses input to earlier operations to predict the input to later operations. The system also inserts a “confirm” operation, which ensures that the final result is correct regardless of whether the prediction is correct. This approach to optimizing streaming dataflow plans can achieve arbitrary speedups by speculating on the speculations. If the system is able to make accurate predictions, the executor could speculate on all of the input, execute the entire plan in parallel, and then confirm all of the results.

Figure 6 shows an example agent plan for integrating car-related data from three online sources. This plan first uses the Edmunds.com site to find the midsize cars priced between \$4000 and \$12000. Next it selects out those cars made by Oldsmobile. Then for each of those cars, in parallel it calls both the NHTSA site to get safety reports and the Consumer Guide site to retrieve car reviews. Finally, all of this information is combined into a single report. Figure 7 shows an abstract version of the same plan with the speculation operations inserted into the plan. The use of speculative execution in this plan makes it possible to invoke all three web sites in parallel.

The effectiveness of the speculation technique depends on making accurate predictions. We have developed a learning system that uses decision tree learning to make predictions on similar inputs and transducer learning to discover patterns in Web navigation paths. The learning system is described in more detail in [Barish and Knoblock, 2003].

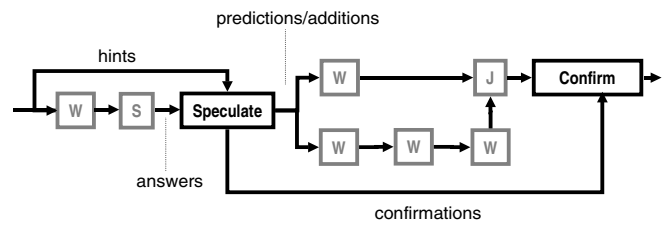


Figure 7: Augmented plan for speculative execution

7 Future Directions

Given the growing interest in both Web Services and the Semantic Web [Hendler, 2001], we believe it will become easier to rapidly create and deploy agents on the Web. As more semantic information becomes available about online sources, we plan to exploit this information to automatically discover and integrate new sources of information. Similarly, as web services become available that can support transactions, we plan to move beyond gathering and monitoring of online sources to build agents that perform more complex tasks, such as not just finding the lowest price ticket, but also purchasing it.

There are many possible uses of agents for retrieving and monitoring data from online sources. Ideally, users could define their own tasks that they want an agent to perform. For example, I might want an agent that monitors airfares and notifies me the moment I can buy a ticket to Hawaii for less than \$300. Someone else might want an agent to monitor for travel delays in their connecting airport and notify them when the average delay exceeds 30 minutes. The possibilities are endless. We are currently working on what we call an Agent Wizard, which allows the user to define new agents for monitoring tasks simply by answering a set of questions about the task. The resulting system will work similar to the Excel Chart Wizard, which converts numerical data into charts by asking the user a set of questions. The Agent Wizard will automatically build the corresponding Theseus plan and schedule the monitoring task for the user.

Another exciting direction is to deploy agents to collect and learn about online data and then use the results to make predictions about the world. For example, we recently developed a system called Hamlet that advises a user about whether they should immediately buy a ticket on a particular flight or wait for a possible drop in the price [Etzioni *et al.*, 2003]. Hamlet makes these recommendations by collecting data on the current pricing of flights on a particular route and then learning a model of the pricing in order to make predictions about future price behavior. In a simulation using real data, Hamlet was able to save 607 simulated passengers \$283,904, which was 88.6% of the savings possible with complete knowledge of the future price changes. Hamlet provides a compelling example of the potential of information agents.

8 Conclusion

The World Wide Web provides a tremendous opportunity for AI researchers to build, deploy and test software agents. The Web provides a real world environment that sidesteps many

of the difficult issues of sensing and control and makes it possible to explore higher level capabilities. We have developed the tools and infrastructure for rapidly constructing agents on the Web for performing various types of information gathering and integration tasks.

While the agents we have developed are extremely useful, there are many interesting and challenging problems that remain to be solved in order to widely deploy agents on the Web. Agents need to be able to robustly accomplish their tasks, responding appropriately to failures of various types. They must be able to communicate flexibly with people and other software agents, ideally in natural language. They need the ability to explain their behavior, especially as the tasks they perform become more complex. And, of course, we want agents that can learn from their past experience to both broaden their capabilities and improve their performance.

Acknowledgments

I want to thank my collaborators for their many contributions to the projects described in this paper. Steve Minton has worked closely with me on defining, building, and executing many of the research projects described here. Jose Luis Ambite, Greg Barish, Maria Muslea, Jean Oh, Snehal Thakkar, and Rattapoon Tuchinda all helped build the travel application of the Electric Elves. Ion Muslea developed the wrapper learning systems, Kristina Lerman developed the wrapper maintenance and repair techniques, Sheila Tejada developed the record linkage approach, Snehal Thakkar and Jose Luis Ambite developed the various query planning algorithms, and Greg Barish built the Theseus executor and speculative execution techniques. I also want to thank both Doug Dyer and Robert Herklotz for their support of my research over the years.

This research was supported in part by the Air Force Office of Scientific Research under grant numbers F49620-01-1-0053 and F49620-02-1-0270, in part by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory under contract/agreement numbers F30602-01-C-0197 and F30602-00-1-0504, in part by the United States Air Force under contract number F49620-02-C-0103, and in part by a gift from the Microsoft Corporation.

The U.S. Government is authorized to reproduce and distribute reports for Governmental purposes notwithstanding any copy right annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

References

- [Ambite and Knoblock, 2000] Jose Luis Ambite and Craig A. Knoblock. Flexible and scalable cost-based query planning in mediators: A transformational approach. *Artificial Intelligence Journal*, 118(1-2):115–161, April 2000.
- [Ambite and Knoblock, 2001] Jose Luis Ambite and Craig A. Knoblock. Planning by rewriting. *Journal of Artificial Intelligence Research*, 15:207–261, 2001.
- [Ambite *et al.*, 2001] Jose Luis Ambite, Craig A. Knoblock, Ion Muslea, and Andrew Philpot. Compiling source descriptions for efficient and flexible information integration. *Journal of Intelligent Information Systems*, 16(2):149–187, March 2001.
- [Ambite *et al.*, 2002] Jose Luis Ambite, Greg Barish, Craig A. Knoblock, Maria Muslea, Jean Oh, and Steven Minton. Getting from here to there: Interactive planning and agent execution for optimizing travel. In *Proceedings of the Fourteenth Conference on Innovative Applications of Artificial Intelligence (IAAI-2002)*, pages 862–869, AAAI Press, Menlo Park, CA, 2002.
- [Ankolenkar *et al.*, 2002] Anupriya Ankolenkar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Drew McDermott, Sheila A. McIlraith, Srinu Narayanan, Massimo Paolucci, Terry R. Payne, and Katia Sycara. Daml-s: Web service description for the semantic web. In *Proceedings of the First International Semantic Web Conference (ISWC)*, Sardinia, Italy, 2002.
- [Barish and Knoblock, 2002a] Greg Barish and Craig A. Knoblock. An efficient and expressive language for information gathering on the web. In *Proceedings of the AIPS-2002 Workshop on Is there life after operator sequencing? – Exploring real world planning*, pages 5–12, Toulouse, France, 2002.
- [Barish and Knoblock, 2002b] Greg Barish and Craig A. Knoblock. Speculative execution for information gathering plans. In *Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling (AIPS 2002)*, pages 184–193, AAAI Press, Menlo Park, CA, 2002.
- [Barish and Knoblock, 2003] Greg Barish and Craig A. Knoblock. Learning value predictors for the speculative execution of information gathering plans. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, Acapulco, Mexico, 2003.
- [Barish *et al.*, 2000] Greg Barish, Craig A. Knoblock, Yi-Shin Chen, Steven Minton, Andrew Philpot, and Cyrus Shahabi. The TheaterLoc virtual application. In *Proceedings of Twelfth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI-2000)*, Austin, Texas, 2000.
- [Chalupsky *et al.*, 2001] Hans Chalupsky, Yolanda Gil, Craig A. Knoblock, Kristina Lerman, Jean Oh, David V. Pynadath, Thomas A. Russ, and Milind Tambe. Electric elves: Applying agent technology to support human organizations. In *Proceedings of the Conference on Innovative Applications of Artificial Intelligence*, 2001.
- [Cohen, 2000] William Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18:288–321, 2000.
- [Crescenzi *et al.*, 2001] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *Proceedings of 27th International Conference on Very Large Data Bases*, pages 109–118, 2001.
- [Decker *et al.*, 1997] Keith Decker, Anandee Pannu, Katia Sycara, and Mike Williamson. Designing behaviors for information agents. In *Proceedings of the First International Conference on Autonomous Agents (Agents’97)*, pages 404–412, ACM Press, New York, 1997.
- [Duschka, 1997] Oliver M. Duschka. *Query Planning and Optimization in Information Integration*. PhD thesis, Stanford University, Department of Computer Science, 1997.
- [Etzioni and Weld, 1994] Oren Etzioni and Daniel S. Weld. A softbot-based interface to the Internet. *Communications of the ACM*, 37(7), 1994.

- [Etzioni *et al.*, 2003] Oren Etzioni, Craig A. Knoblock, Rattapoom Tuchinda, and Alexander Yates. To buy or not to buy: Mining airline fare data to minimize ticket purchase price. Submitted for Publication, 2003.
- [Etzioni, 1996] Oren Etzioni. Moving up the information food chain: Deploying softbots on the world wide web. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1322–1326, AAAI Press / MIT Press, Menlo Park, 1996.
- [Hellerstein *et al.*, 2000] Joseph M. Hellerstein, Michael J. Franklin, Sirish Chandrasekaran, Amol Deshpande, Kris Hildrum, Sam Madden, Vijayshankar Raman, and Mehul A. Shah. Adaptive query processing: technology in evolution. *IEEE Data Engineering Bulletin*, 23(2):7–18, 2000.
- [Hendler, 2001] James Hendler. Agents on the web. *IEEE Intelligent Systems, Special Issue on the Semantic Web*, 16(2):30–37, March/April 2001.
- [Hong and Clark, 2001] Theodore W. Hong and Keith L. Clark. Using grammatical inference to automate information extraction from the Web. In *Principles of Data Mining and Knowledge Discovery, Lecture Notes in Computer Science*, volume 2168, pages 216–227. Springer-Verlag, 2001.
- [Hsu and Dung, 1998] Chun-Nan Hsu and Ming-Tzung Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.
- [Ives *et al.*, 2002] Zachary G. Ives, Alon Y. Halevy, and Daniel S. Weld. An XML query engine for network-bound data. *VLDB Journal*, 11(4):380–402, 2002.
- [Knoblock *et al.*, 1998] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Pragnesh Jay Modi, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. Modeling web sources for information integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, Madison, WI, 1998.
- [Knoblock *et al.*, 2001] Craig A. Knoblock, Steven Minton, Jose Luis Ambite, Naveen Ashish, Ion Muslea, Andrew G. Philpot, and Sheila Tejada. The ARIADNE approach to web-based information integration. *International Journal of Cooperative Information Systems (IJCIS), Special Issue on Intelligent Information Agents: Theory and Applications*, 10(1/2):145–169, 2001.
- [Knoblock *et al.*, 2003] Craig A. Knoblock, Kristina Lerman, Steven Minton, and Ion Muslea. Accurately and reliably extracting data from the web: A machine learning approach. In Piotr S. Szczepaniak, Javier Segovia, Janusz Kacprzyk, and Lotfi A. Zadeh, editors, *Intelligent Exploration of the Web*, pages 275–287. Springer-Verlag, Berkeley, CA, 2003.
- [Krulwich, 1996] Bruce Krulwich. The bargainfinder agent: Comparison price shopping on the internet. In *Agents, Bots, and other Internet Beasties*, pages 257–263. Macmillan Publishing, May 1996.
- [Kushmerick, 1997] Nicholas Kushmerick. *Wrapper Induction for Information Extraction*. PhD thesis, Department of Computer Science and Engineering, University of Washington, 1997.
- [Kushmerick, 2000] Nicholas Kushmerick. Wrapper verification. *World Wide Web*, 3(2):79–94, 2000.
- [Lambrecht and Kambhampati, 1997] Eric Lambrecht and Subbarao Kambhampati. Planning for information gathering: A tutorial survey. ASU CSE technical report 96-017, Department of Computer Science and Engineering, Arizona State University, May 1997.
- [Lerman *et al.*, 2001] Kristina Lerman, Craig A. Knoblock, and Steven Minton. Automatic data extraction from lists and tables in web sources. In *Proceedings of the IJCAI 2001 Workshop on Adaptive Text Extraction and Mining*, Seattle, WA, 2001.
- [Lerman *et al.*, 2003] Kristina Lerman, Steven N. Minton, and Craig A. Knoblock. Wrapper maintenance: A machine learning approach. *Journal of Artificial Intelligence Research*, 18:149–181, 2003.
- [Levy and Weld, 2000] Alon Y. Levy and Daniel S. Weld. Intelligent internet systems. *Artificial Intelligence*, 118(1-2):1–14, April 2000.
- [Levy, 2000] Alon Y. Levy. Logic-based techniques in data integration. In Jack Minker, editor, *Logic-Based Artificial Intelligence*. Kluwer Academic Publisher, 2000.
- [Muslea *et al.*, 2000] Ion Muslea, Steven Minton, and Craig A. Knoblock. Selective sampling with redundant views. In *Proceedings of the 17th National Conference on Artificial Intelligence*, 2000.
- [Muslea *et al.*, 2001] Ion Muslea, Steven Minton, and Craig A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2), March 2001.
- [Muslea *et al.*, 2003] Ion Muslea, Steven Minton, and Craig A. Knoblock. Active learning with strong and weak views: A case study on wrapper induction. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-2003)*, Acapulco, Mexico, 2003.
- [Muslea, 2002] Ion Muslea. *Active Learning with Multiple Views*. PhD thesis, Department of Computer Science, University of Southern California, 2002.
- [Perkowitz *et al.*, 1997] Mike Perkowitz, Robert B. Doorenbos, Oren Etzioni, and Daniel S. Weld. Learning to understand information on the internet: An example-based approach. *Journal of Intelligent Information Systems*, 8(2):133–153, 1997.
- [Sarawagi and Bhamidipaty, 2002] Sunita Sarawagi and Anuradha Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta, Canada, 2002.
- [Tejada *et al.*, 2001] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning object identification rules for information integration. *Information Systems*, 26(8), 2001.
- [Tejada *et al.*, 2002] Sheila Tejada, Craig A. Knoblock, and Steven Minton. Learning domain-independent string transformation weights for high accuracy object identification. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2002)*, Edmonton, Alberta, Canada, 2002.
- [Thakkar and Knoblock, 2003] Snehal Thakkar and Craig A. Knoblock. Efficient execution of recursive integration plans. In *Proceedings of 2003 IJCAI Workshop on Information Integration on the Web*, Acapulco, Mexico, 2003.
- [Thakkar *et al.*, 2003] Snehal Thakkar, Jose-Luis Ambite, and Craig A. Knoblock. A view integration approach to dynamic composition of web services. In *Proceedings of 2003 ICAPS Workshop on Planning for Web Services*, Trento, Italy, 2003.

Constraint Satisfaction, Databases, and Logic

Phokion G. Kolaitis*

Computer Science Department
University of California, Santa Cruz
Santa Cruz, CA 95064, U.S.A
kolaitis@cs.ucsc.edu

1 Introduction

Constraint satisfaction problems constitute a broad class of algorithmic problems that are ubiquitous in several different areas of artificial intelligence and computer science. In their full generality, constraint satisfaction problems are NP-complete and, thus, presumed to be algorithmically intractable. To cope with the intractability of these problems, researchers have devoted considerable research efforts to both the design of heuristic algorithms for constraint satisfaction and the pursuit of “islands of tractability”, that is, special cases of constraint satisfaction problems for which polynomial-time algorithms exist.

During the past decade, the pursuit of “islands of tractability” of constraint satisfaction has been intensified and has led to a number of discoveries that have also unveiled tight connections between constraint satisfaction, database theory, logic, and universal algebra. Our goal in this paper is to present an overview of the current state of affairs in the study of the computational complexity of constraint satisfaction with emphasis on the connections of this area of research with database theory and logic. The paper is organized as follows: Section 2 contains the precise definition of the CONSTRAINT SATISFACTION PROBLEM and its reformulation as the HOMOMORPHISM PROBLEM; Section 3 contains some of the connections between constraint satisfaction problems and database theory; the remaining Sections 4, 5, and 6 contain a high-level account of some of the main results about the computational complexity of constraint satisfaction and the pursuit of tractable cases of this problem.

2 The Constraint Satisfaction Problem and the Homomorphism Problem

Constraint satisfaction problems were originally introduced by Montanari [Mon74] to model problems in computer vision. Since that time, however, it has been realized that numerous important problems in artificial intelligence and computer science can be modeled as constraint satisfaction problems (see [Dec92a; PJ97]). An instance of the CONSTRAINT SATISFACTION PROBLEM (CSP) consists of a set of variables, a set of possible values for the variables, and a set of constraints on tuples of variables that restrict the combinations of values that the variables may take; the question

is to determine whether there is an assignment of values to the variables so that the constraints are satisfied. More precisely, a CSP-instance is a triple (V, D, C) , where V is a finite set of *variables*, D is a finite *domain* of values for the variables, and C is a set *constraints* (t, R) , where t is a tuple $t = (x_1, \dots, x_m)$ of variables for some m and R is a relation on D of arity m . A *solution* is a mapping $h : V \rightarrow D$ such that, for every constraint $(t, R) \in C$, we have that $h(t) = (h(x_1), \dots, h(x_m)) \in R$.

Consider the Boolean satisfiability problem 3-SAT: given a 3CNF-formula φ with variables $x_1, \dots, x_n$ and clauses $c_1, \dots, c_k$, is φ satisfiable? Such an instance of 3-SAT can be thought of as the CSP-instance in which the set of variables is $V = \{x_1, \dots, x_n\}$, the domain is $D = \{0, 1\}$, and the constraints are determined by the clauses of φ . For example, a clause of the form $(x \vee \neg y \vee z)$ gives rise to the constraint $((x, y, z), \{0, 1\}^3 - \{(0, 1, 0)\})$. In an analogous manner, 3-COLORABILITY can be modeled as a constraint satisfaction problem. Indeed, an instance $\mathbf{H} = (V, E)$ of 3-COLORABILITY can be thought of as the CSP-instance in which the set of variables is the set V of the nodes of the graph $\mathbf{H}$, the domain is the set $D = \{r, b, g\}$ of three colors, and the constraints are the pairs $((u, v), Q)$, where $(u, v) \in E$ and $Q = \{(r, b)(b, r), (r, g)(g, r), (b, g)(g, b)\}$ is the inequality relation on D .

A *vocabulary* is a finite set of relational symbols $R_1, \dots, R_m$ each of which has a fixed arity. A *relational structure* over some vocabulary is a tuple $\mathbf{A} = (A, R_1^{\mathbf{A}}, \dots, R_m^{\mathbf{A}})$ such that A is a non-empty set, called the *universe* of $\mathbf{A}$, and each $R_i^{\mathbf{A}}$ is a relation on A having the same arity as the symbol R_i . Let $\mathbf{A}$ and $\mathbf{B}$ be two relational structures over the same vocabulary. A *homomorphism* h from $\mathbf{A}$ to $\mathbf{B}$ is a mapping $h : A \rightarrow B$ from the universe A of $\mathbf{A}$ to the universe B of $\mathbf{B}$ such that, for every relation $R^{\mathbf{A}}$ of $\mathbf{A}$ and every tuple $(a_1, \dots, a_n) \in R^{\mathbf{A}}$, we have that $(h(a_1), \dots, h(a_n)) \in R^{\mathbf{B}}$. Feder and Vardi [FV98] were the first to point out that the CONSTRAINT SATISFACTION PROBLEM can be identified with the HOMOMORPHISM PROBLEM: given two relational structures $\mathbf{A}$ and $\mathbf{B}$, is there a homomorphism h from $\mathbf{A}$ to $\mathbf{B}$? The intuition behind this identification is that the structure $\mathbf{A}$ represents the variables and the tuples of variables that occur in the constraints, while the structure $\mathbf{B}$ represents the domain of values and the tuples of values that these constrained tuples of variables are

*Partially supported by NSF Grant No. IIS-9907419

allowed to take. Moreover, the homomorphisms from $\mathbf{A}$ to $\mathbf{B}$ are precisely the assignments of values to variables that satisfy the constraints. For instance, 3-COLORABILITY is equivalent to the problem of deciding whether there is a homomorphism h from a given graph $\mathbf{H}$ to the complete graph $\mathbf{K}_3 = (\{r, b, g\}, \{(r, b)(b, r), (r, g)(g, r), (b, g)(g, b)\})$ with 3 nodes. More generally, k -COLORABILITY, $k \geq 2$, amounts to the existence of a homomorphism from a given graph H to the complete graph $\mathbf{K}_k$ with k nodes (also known as the k -clique). Numerous other important NP-complete problems can be viewed as special cases of the HOMOMORPHISM PROBLEM (and, hence, also of the CONSTRAINT SATISFACTION PROBLEM). For example, consider the CLIQUE problem: given a graph $\mathbf{H}$ and an integer k , does $\mathbf{H}$ contain a clique of size k ? A moment's reflection shows that this is equivalent to the question: given a complete graph $\mathbf{K}_k$ and a graph $\mathbf{H}$, is there a homomorphism from $\mathbf{K}_k$ to $\mathbf{H}$?

The conceptual insights gained from the identification of the CONSTRAINT SATISFACTION PROBLEM with the HOMOMORPHISM PROBLEM have facilitated the use of techniques from universal algebra in the study of constraint satisfaction [Jea98; FV98]. Moreover, they have clarified the tight connections between constraint satisfaction and database theory. We discuss some of these connections in the next section.

3 Constraint Satisfaction and Relational Databases

The most frequently asked queries in relational database systems involve the computation of the *join* of two or more relations in a database. Instead of spelling out the formal definition of the join operation, let us consider for concreteness a database relation $R(A, B, C)$ with A, B, C as attributes and a database relation $S(B, C, D, E)$ with B, C, D, E as attributes. Then the join $R \bowtie S$ consists of all quintuples (a, b, c, d, e) such that $(a, b, c) \in R$ and $(b, c, d, e) \in S$. Several different researchers, including [Bib88], have pointed out that computing the set of all solutions of a CSP-instance can be viewed as a join evaluation problem. Indeed, if $(V, D, \mathcal{C})$ is a CSP-instance, then, for every constraint $(V_{i_1}, \dots, V_{i_m}), R$ in $\mathcal{C}$, let $R_i(V_{i_1}, \dots, V_{i_m})$ be the relation R itself viewed as a database relation with attributes $V_{i_1}, \dots, V_{i_m}$. If $R_1, \dots, R_t$ is the collection of all database relations obtained this way, then the join $R_1 \bowtie \dots \bowtie R_t$ consists of all solutions to the CSP-instance $(V, D, \mathcal{C})$.

Join evaluation is a special case of conjunctive query evaluation. Specifically, an n -ary *conjunctive query* Q over a relational vocabulary is a query definable by a positive existential first-order formula of the form $(\exists z_1) \dots (\exists z_s) \psi(x_1, \dots, x_n, z_1, \dots, z_s)$, where $\psi(x_1, \dots, x_n, z_1, \dots, z_s)$ is a conjunction of atomic formulas. For example, the binary conjunctive query “there is a path of length 3 from x_1 to x_2 ” is definable by the formula $(\exists z_1)(\exists z_2)(E(x_1, z_1) \wedge E(z_1, z_2) \wedge E(z_2, x_2))$. Note that joins are precisely those conjunctive queries in which none of the variables in ψ is quantified. At other extreme, a *Boolean conjunctive query* is a conjunctive query in which all variables of ψ have been quantified out.

Every finite relational structure $\mathbf{A}$ gives rise to a *canonical* Boolean conjunctive query $Q^{\mathbf{A}}$; the positive existential first-order sentence defining $Q^{\mathbf{A}}$ asserts that there exist as many elements as the cardinality of the universe of $\mathbf{A}$ and states all atomic facts satisfied by tuples from the universe of $\mathbf{A}$. For example, if $\mathbf{A} = (A, E)$ is the 3-cycle with $A = \{1, 2, 3\}$ and $E = \{(1, 2), (2, 3), (3, 1)\}$, then the canonical conjunctive query $Q^{\mathbf{A}}$ is definable by the sentence

$$(\exists x_1)(\exists x_2)(\exists x_3)(E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_3, x_1)).$$

The following basic result, due to Chandra and Merlin [CM77], establishes a strong connection between homomorphisms and conjunctive queries.

Theorem 3.1: [CM77] *The following are equivalent for finite relational structures $\mathbf{A}$ and $\mathbf{B}$.*

1. *There is a homomorphism h from $\mathbf{A}$ to $\mathbf{B}$.*
2. $Q^{\mathbf{B}} \subseteq Q^{\mathbf{A}}$, *that is, $Q^{\mathbf{B}}$ logically implies $Q^{\mathbf{A}}$.*
3. $\mathbf{B} \models Q^{\mathbf{A}}$, *that is, the structure $\mathbf{B}$ satisfies the canonical query $Q^{\mathbf{A}}$ of $\mathbf{A}$.*

As an example, we saw earlier that a graph $\mathbf{G} = (V, E)$ is 3-colorable if and only if there is a homomorphism from $\mathbf{G}$ to $\mathbf{K}_3$. Consequently, Theorem 3.1 implies that $\mathbf{G}$ is 3-colorable if and only if $\mathbf{K}_3$ satisfies the canonical query $Q^{\mathbf{G}}$ of $\mathbf{G}$.

In view of the identification of the CONSTRAINT SATISFACTION PROBLEM with the HOMOMORPHISM PROBLEM, the preceding Theorem 3.1 implies that the CONSTRAINT SATISFACTION PROBLEM can also be identified with two fundamental problems in database theory: *conjunctive query evaluation* and *conjunctive query implication (or containment)*. This fundamental connection between constraint satisfaction and database theory was brought to front stage and further investigated in [KV00a].

4 Computational Complexity of Constraint Satisfaction

The CONSTRAINT SATISFACTION PROBLEM is NP-complete, because it is clearly in NP and also contains NP-hard problems as special cases, including 3-SAT, 3-COLORABILITY, and CLIQUE. As explained in Garey and Johnson's classic monograph [GJ79], one of the main ways to cope with NP-completeness is to identify polynomial-time solvable cases of the problem at hand that are obtained by imposing restrictions on the possible inputs. For instance, HORN 3-SAT, the restriction of 3-SAT to Horn 3CNF-formulas, is solvable in polynomial-time using a unit-propagation algorithm. Similarly, it is known that 3-COLORABILITY restricted to graphs of bounded treewidth is solvable in polynomial time (see [DF99]). In the case of constraint satisfaction, the pursuit of tractable cases has evolved over the years from the discovery of isolated cases to the discovery of large “islands of tractability” of constraint satisfaction. In what follows, we will give an account of some of the progress made in this area. We begin by introducing some terminology and notation that will enable us to formalize the concept of an “island of tractability” of constraint satisfaction using the fact that the CONSTRAINT SATISFACTION PROBLEM can be identified with the HOMOMORPHISM PROBLEM.

In general, an instance of the HOMOMORPHISM PROBLEM consists of two arbitrary relational structures $\mathbf{A}$ and $\mathbf{B}$. Thus, all restricted cases of this problem can be obtained by imposing restrictions on the input structures $\mathbf{A}$ and $\mathbf{B}$.

Definition 4.1: Let $\mathcal{A}, \mathcal{B}$ be two classes of relational structures. We write $\text{CSP}(\mathcal{A}, \mathcal{B})$ to denote the restriction of the HOMOMORPHISM PROBLEM to input structures from $\mathcal{A}$ and $\mathcal{B}$. In other words,

$$\text{CSP}(\mathcal{A}, \mathcal{B}) = \{(\mathbf{A}, \mathbf{B}) : \mathbf{A} \in \mathcal{A}, \mathbf{B} \in \mathcal{B} \text{ and a homomorphism } h : \mathbf{A} \rightarrow \mathbf{B} \text{ exists}\}$$

An *island of tractability* of constraint satisfaction is a pair $(\mathcal{A}, \mathcal{B})$ of classes of relational structures such that $\text{CSP}(\mathcal{A}, \mathcal{B})$ is in the complexity class P of all decision problems solvable in polynomial time.

The ultimate goal in the pursuit of islands of tractability of constraint satisfaction is to identify or somehow characterize all classes $\mathcal{A}$ and $\mathcal{B}$ of relational structures such that $\text{CSP}(\mathcal{A}, \mathcal{B})$ is in P. The basic starting point in this investigation is to consider the cases in which one of the two classes $\mathcal{A}, \mathcal{B}$ is as small as possible, while the other is as large as possible. This amounts to considering the cases in which one of $\mathcal{A}, \mathcal{B}$ is the class *All* of all relational structures over some arbitrary, but fixed, relational vocabulary, while the other is a singleton $\{\mathbf{B}\}$ consisting of some fixed structure $\mathbf{B}$ over that vocabulary. Thus, the starting points of the investigation is to determine, for every relational structure $\mathbf{B}$, the computational complexity of the decision problems $\text{CSP}(\{\mathbf{B}\}, \text{All})$ and $\text{CSP}(\text{All}, \{\mathbf{B}\})$.

Clearly, for each fixed $\mathbf{B}$, the decision problem $\text{CSP}(\{\mathbf{B}\}, \text{All})$ can be solved in polynomial time, because, given a structure $\mathbf{A}$, the existence of a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ can be checked by testing all functions h from the universe B of $\mathbf{B}$ to the universe A of $\mathbf{A}$ (the total number of such functions is $|A|^{|B|}$, which is a polynomial number in the size of the structure $\mathbf{A}$). At the other extreme, however, the situation is quite different, since the computational complexity of $\text{CSP}(\text{All}, \{\mathbf{B}\})$ may very well depend on the particular structure $\mathbf{B}$. Indeed, $\text{CSP}(\text{All}, \{\mathbf{K}_3\})$ is NP-complete, because it is the 3-COLORABILITY problem; in contrast, $\text{CSP}(\text{All}, \{\mathbf{K}_2\})$ is in P, because it is the 2-COLORABILITY problem.

For simplicity, in what follows, for every fixed structure $\mathbf{B}$, we put

$$\text{CSP}(\mathbf{B}) = \text{CSP}(\text{All}, \{\mathbf{B}\})$$

and call this the *non-uniform* constraint satisfaction problem associated with $\mathbf{B}$. Thus, the first major goal in the study of the computational complexity of constraint satisfaction is to identify those structures $\mathbf{B}$ for which $\text{CSP}(\mathbf{B})$ is in P. Although this goal has yet to be realized, much progress has been made towards it. The next section contains a bird's-eye view of some of the main results obtained to date.

5 The Computational Complexity of Non-Uniform Constraint Satisfaction

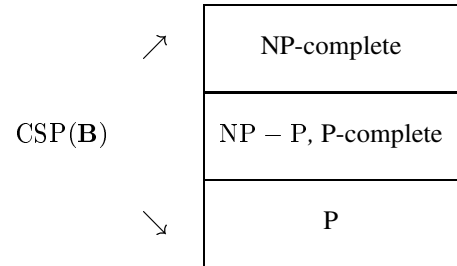
The first major result in the study of non-uniform constraint satisfaction problems was obtained by Schaefer [Sch78],

who, in effect, classified the computational complexity of all Boolean non-uniform constraint satisfaction problems. A *Boolean* structure is simply a relational structure with a 2-element universe, that is, a structure of the form $\mathbf{B} = (\{0, 1\}, R_1^{\mathbf{B}}, \dots, R_m^{\mathbf{B}})$. A *Boolean non-uniform constraint satisfaction problem* is a problem of the form $\text{CSP}(\mathbf{B})$ with $\mathbf{B}$ a Boolean structure. These problems are also known as GENERALIZED SATISFIABILITY PROBLEMS, because they can be viewed as variants of Boolean satisfiability problems in which the formulas are conjunctions of generalized connectives [GJ79]. In particular, they contain the well known problems k -SAT, $k \geq 2$, 1-IN-3-SAT, POSITIVE 1-IN-3-SAT, NOT-ALL-EQUAL 3-SAT, and MONOTONE 3-SAT as special cases. For example, 3-SAT is $\text{CSP}(\mathbf{B})$, where $\mathbf{B} = (\{0, 1\}, R_0, R_1, R_2, R_3)$ and R_i is the set of truth assignments that satisfy a 3-clause in which the first i -literals are negated, $i = 0, 1, 2, 3$ (thus, $R_0 = \{0, 1\}^3 - \{(0, 0, 0)\}$). Similarly, MONOTONE 3-SAT is $\text{CSP}(\mathbf{B})$, where $\mathbf{B} = (\{0, 1\}, R_0, R_3)$.

Ladner [Lad75] showed that if $P \neq NP$, then there are decision problems in NP that are neither NP-complete, nor they belong to P. Consequently, it is conceivable that a given family of NP-problems contains problems of such intermediate complexity. Schaefer [Sch78], however, showed that the family of all Boolean non-uniform constraint satisfaction problems contains no problems of intermediate complexity.

Theorem 5.1: (Schaefer's Dichotomy Theorem [Sch78])

- If $\mathbf{B} = (\{0, 1\}, R_1^{\mathbf{B}}, \dots, R_m^{\mathbf{B}})$ is Boolean structure, then either $\text{CSP}(\mathbf{B})$ is in P or $\text{CSP}(\mathbf{B})$ is NP-complete. In a picture,



- Moreover, there is a polynomial-time algorithm to decide, given a Boolean structure $\mathbf{B}$, whether $\text{CSP}(\mathbf{B})$ is in P or it is NP-complete.

Schaefer [Sch78] actually showed that there are exactly six types of Boolean structures such that $\text{CSP}(\mathbf{B})$ is in P, and provided explicit descriptions of them. Specifically, he showed that $\text{CSP}(\mathbf{B})$ is in P precisely when at least one of the following six conditions is satisfied:

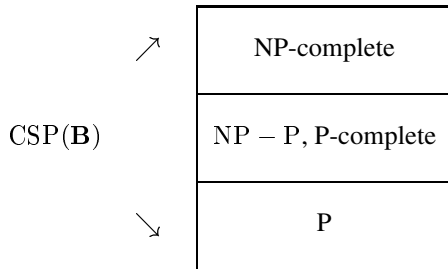
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of $\mathbf{B}$ is 0-valid, that is, $R_i^{\mathbf{B}}$ contains the all-zeroes tuple $(0, \dots, 0)$.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of $\mathbf{B}$ is 1-valid, that is, $R_i^{\mathbf{B}}$ contains the all-ones tuple $(1, \dots, 1)$.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of $\mathbf{B}$ is *bijunctive*, that is, $R_i^{\mathbf{B}}$ is the set of truth assignments satisfying some 2CNF formula.

- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of $\mathbf{B}$ is *Horn*, that is, $R_i^{\mathbf{B}}$ is the set of truth assignments satisfying some Horn formula.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of $\mathbf{B}$ is *dual Horn*, that is, $R_i^{\mathbf{B}}$ is the set of truth assignments satisfying some dual Horn formula.
- Every relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, of $\mathbf{B}$ is *affine*, that is, $R_i^{\mathbf{B}}$ is the set of solutions to a system of linear equations over the two-element field.

Schaefer's Dichotomy Theorem yields a complete classification of the complexity of $\text{CSP}(\mathbf{B})$ for Boolean structures $\mathbf{B}$. At the same time, it raises the challenge of classifying the computational complexity of $\text{CSP}(\mathbf{B})$ for arbitrary relational structures $\mathbf{B}$. To this effect, Feder and Vardi [FV98] formulated the following important conjecture.

Conjecture 5.2: (Feder-Vardi Dichotomy Conjecture)

If $\mathbf{B} = (\mathbf{B}, R_1^{\mathbf{B}}, \dots, R_m^{\mathbf{B}})$ is an arbitrary relational structure, then either $\text{CSP}(\mathbf{B})$ is in P or $\text{CSP}(\mathbf{B})$ is NP-complete. In a picture,



The Feder-Vardi Dichotomy Conjecture inspired intensive research efforts that resulted into significant advances towards resolving it. In particular, quite recently Bulatov confirmed two important cases of this conjecture that are described next.

Theorem 5.3: Let $\mathbf{B} = (B, R_1^{\mathbf{B}}, \dots, R_m^{\mathbf{B}})$ be a relational structure.

- (Bulatov [Bul02a]) If $B = \{0, 1, 2\}$, then either $\text{CSP}(\mathbf{B})$ is in P or $\text{CSP}(\mathbf{B})$ is NP-complete.
- (Bulatov [Bul03]) If every non-empty subset of B is one of the relations $R_i^{\mathbf{B}}$ of $\mathbf{B}$, then either $\text{CSP}(\mathbf{B})$ is in P or $\text{CSP}(\mathbf{B})$ is NP-complete.

Moreover, in both cases there is a polynomial-time algorithm to decide whether $\text{CSP}(\mathbf{B})$ is in P or it is NP-complete.

The second of the two cases above is known as *conservative constraint satisfaction*. In effect, it is the case in which all possible unary constraints on the domain are present.

In spite of the progress made, the Feder-Vardi Dichotomy Conjecture remains unresolved for $\text{CSP}(\mathbf{B})$ with $|B| \geq 4$. The research efforts towards this conjecture, however, have also resulted into the discovery of broad sufficient conditions for tractability of non-uniform constraint satisfaction that have provided unifying explanations for numerous seemingly disparate tractability results and have also led to the discovery of new islands of tractability of $\text{CSP}(\mathbf{B})$. These broad sufficient conditions for tractability of $\text{CSP}(\mathbf{B})$ are based on

concepts and techniques from two different areas: universal algebra and logic. The approach via universal algebra yields sufficient conditions for tractability of $\text{CSP}(\mathbf{B})$ in terms of *closure* properties of the relations in $\mathbf{B}$ under certain functions on B . The approach via logic yields sufficient conditions for tractability in terms of *expressibility* of $\text{CSP}(\mathbf{B})$ in Datalog. In the remainder of this section, our primary focus is on the latter approach.

Datalog is the main database query language used in deductive database systems (see [Ull89]). In a nutshell, a Datalog program is a negation-free and function-free logic program. More precisely, a *Datalog program* is a finite set of rules of the form

$$T(\bar{x}) : - S_1(\bar{y}_1), \dots, S_r(\bar{y}_r),$$

where $T, S_1, \dots, S_r$ are relation symbols. In effect, the right-hand side of each rule (called the *body* of the rule) is a conjunctive query in which all variables not occurring in the left-hand side (called the *head* of the rule) are existentially quantified. Datalog embodies recursion because relation symbols may occur both in the heads and the bodies of rules. Those that do are the *recursive* or *intensional database predicates* (IDBs) of the program, while the remaining relation symbols are the *extensional database predicates* (EDBs). One of the IDB predicates is singled out as the *goal* of the program.

As a standard example, the *transitive closure* of the edge relation E of a graph $\mathbf{H} = (V, E)$ is defined by the following Datalog program.

$$\begin{array}{lcl} T(x, y) & :- & E(x, y) \\ T(x, y) & :- & E(x, z) \wedge T(z, y) \end{array}$$

Every Datalog program can be evaluated “bottom-up” in a polynomial number of iterations on a given database. For example, the k -th iteration of the above Datalog program yields the set of all pairs of nodes of $\mathbf{H}$ that are connected via a path of length at most k ; moreover, at most n iterations suffice to compute the transitive closure of E , where n is the number of nodes in V . It follows that each fixed Datalog program can be evaluated in time polynomial in the size of a given relational structure. Consequently, if a query Q is definable by a Datalog program, then Q is in P. Thus, expressibility in Datalog is a sufficient condition for tractability.

Another important feature of Datalog is that queries definable by Datalog programs are *preserved under homomorphisms*. This means that if a structure $\mathbf{A}$ satisfies the goal of a Datalog program and there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$, then also $\mathbf{B}$ satisfies the goal of the program.

Let $\mathbf{B} = (B, R_1^{\mathbf{B}}, \dots, R_m^{\mathbf{B}})$ be a relational structure. It is easy to see that, except for trivial situations, $\text{CSP}(\mathbf{B})$ is not preserved under homomorphisms, which implies that $\text{CSP}(\mathbf{B})$ is *not* expressible in Datalog. Thus, at first sight, it appears that there is no link between expressibility in Datalog and tractability of non-uniform constraint satisfaction. A moment's reflection, however, reveals that the *complement* $\overline{\text{CSP}(\mathbf{B})}$ of $\text{CSP}(\mathbf{B})$ is preserved under homomorphisms, where

$$\overline{\text{CSP}(\mathbf{B})} = \{\mathbf{A} : \text{no homomorphism } h : \mathbf{A} \rightarrow \mathbf{B} \text{ exists}\}.$$

Consequently, it is conceivable that, for some structures $\mathbf{B}$, the non-uniform constraint satisfaction problem $\text{CSP}(\mathbf{B})$ is in P because its complement $\overline{\text{CSP}(\mathbf{B})}$ is expressible in Datalog. Feder and Vardi [FV98] pursued this link in depth and demonstrated that expressibility of $\overline{\text{CSP}(\mathbf{B})}$ in Datalog is a unifying explanation for numerous tractability results about $\text{CSP}(\mathbf{B})$.

As an important concrete example, consider 2-COLORABILITY, which is the same problem as $\text{CSP}(\mathbf{K}_2)$. The following Datalog program with Q as its goal expresses NON 2-COLORABILITY, since a graph is 2-colorable if and only if it contains no cycles of odd length.

$$\begin{array}{l|l} O(X, Y) & :- E(X, Y) \\ O(X, Y) & :- O(X, Z), E(Z, W), E(W, Y) \\ Q & :- O(X, X) \end{array}$$

As two additional important examples, recall that if $\mathbf{B} = (\{0, 1\}, R_1^{\mathbf{B}}, \dots, R_m^{\mathbf{B}})$ is a Boolean structure such that every relation $R_i^{\mathbf{B}}$ is Horn or every relation $R_i^{\mathbf{B}}$ is bijunctive, then $\text{CSP}(\mathbf{B})$ is in P. It can be shown that in both these cases $\overline{\text{CSP}(\mathbf{B})}$ is expressible in Datalog.

To gauge the broad spectrum of tractable cases covered by Datalog, it is perhaps worth comparing tractability via Datalog to tractability via closure properties, which, as mentioned earlier, is an approach to tractability based on universal algebra. For this, we need to first introduce some basic concepts from universal algebra. If R is an n -ary relation on a set B and $f : B^k \rightarrow B$ is a function, then we say that R is *closed under f* if for every k -tuples $(a_{11}, \dots, a_{1n}), \dots, (a_{k1}, \dots, a_{kn})$ in R , we have that the n -tuple $(f(a_{11}, \dots, a_{k1}), \dots, f(a_{1n}, \dots, a_{kn}))$ is also in R . If $\mathbf{B} = (B, R_1^{\mathbf{B}}, \dots, R_m^{\mathbf{B}})$ is a relational structure, then a *polymorphism* of $\mathbf{B}$ is a function $f : B^k \rightarrow B$ for some $k \geq 1$ such that each relation $R_i^{\mathbf{B}}$, $1 \leq i \leq m$, is closed under f . We write $\text{Pol}(\mathbf{B})$ for the set of all polymorphisms of $\mathbf{B}$. As it turns out, the complexity of $\text{CSP}(\mathbf{B})$ is intimately connected to the kinds of functions that $\text{Pol}(\mathbf{B})$ contains. In particular, it has been shown that if $\text{Pol}(\mathbf{B})$ contains functions that satisfy certain algebraic identities, then $\text{CSP}(\mathbf{B})$ is in P. This connection has been investigated in depth and with much success by Jeavons and his collaborators in a sequence of papers, including [JCG97; JCC98], and by Bulatov [Bul02b] (but also Feder and Vardi [FV98] had results along these lines). The following are the presently known most general sufficient conditions for tractability of $\text{CSP}(\mathbf{B})$ based on closure properties.

Theorem 5.4: Let $\mathbf{B} = (B, R_1^{\mathbf{B}}, \dots, R_m^{\mathbf{B}})$ be a relational structure.

- If $\text{Pol}(\mathbf{B})$ contains a near-unanimity function, then $\text{CSP}(\mathbf{B})$ is in P.
- If $\text{Pol}(\mathbf{B})$ contains a set function, then $\text{CSP}(\mathbf{B})$ is in P.
- If $\text{Pol}(\mathbf{B})$ contains a Maltsev function, then $\text{CSP}(\mathbf{B})$ is in P.

A k -ary function f with $k \geq 3$ is a *near unanimity function* if $f(x_1, \dots, x_k) = y$, for every k -tuple $(x_1, \dots, x_k)$ such that at least $k - 1$ of the x_i 's are equal to y . Note that

the ternary majority function on $\{0, 1\}$ is a near unanimity function. A k -ary function f with $k \geq 2$ is a *set function* if for every k -tuple $(x_1, \dots, x_k)$, we have that $f(x_1, \dots, x_k)$ depends only on the set $\{x_1, \dots, x_k\}$. Note that the Boolean binary functions $\wedge$ and $\vee$ are set functions. Finally, a ternary function $f(x, y, z)$ is a *Maltsev function* if, for every x and y , it satisfies the identities $f(y, y, x) = f(x, y, y) = x$. Note that the Boolean function $x \oplus y \oplus z$ is a Maltsev function. It should be pointed out that the preceding Theorem 5.4 contains as special cases the non-trivial tractable cases in Schaefer's Dichotomy Theorem 5.1. Indeed, it is known that a Boolean relation is bijunctive if and only if it is closed under the ternary majority operation; moreover, a Boolean relation is Horn (dual Horn) if and only if it is closed under $\wedge$ (respectively, $\vee$); finally, a Boolean relation is affine if and only if it is closed under $x \oplus y \oplus z$.

Concerning the comparison between Datalog and closure properties, it can be shown that if $\text{Pol}(\mathbf{B})$ contains a near-unanimity function or a set function, then $\overline{\text{CSP}(\mathbf{B})}$ is expressible in Datalog. Thus, expressibility in Datalog subsumes two of the three sufficient conditions for tractability based on closure properties. It also known, however, that there are structures $\mathbf{B}$ (in fact, even Boolean structures) such that $\text{Pol}(\mathbf{B})$ contains a Maltsev function, but $\overline{\text{CSP}(\mathbf{B})}$ is not expressible in Datalog. At the same time, there are structures $\mathbf{B}$ such that $\text{Pol}(\mathbf{B})$ does not contain any near unanimity functions, set functions, or Maltsev functions, yet $\text{CSP}(\mathbf{B})$ is tractable because $\overline{\text{CSP}(\mathbf{B})}$ is expressible in Datalog.

In what follows in this section, we will take a closer look at the connections between Datalog and non-uniform constraint satisfaction. In particular, we will address the following question: when is $\overline{\text{CSP}(\mathbf{B})}$ expressible in Datalog? As we will see, expressibility of $\overline{\text{CSP}(\mathbf{B})}$ in Datalog can be characterized in terms of pebble games and also in terms of consistency properties.

Combinatorial games are a versatile tool in analyzing the expressive power of logics. The most well known among these games are the Ehrenfeucht-Fraïssé-games for first-order logic (see [EFT94]). A different family of games, known as k -pebble games, has been used to study fixed-point logics and infinitary logics with finitely many variables (see [KV90]). We now describe a variant of k -pebble games that are suitable for analyzing the expressive power of Datalog [KV95].

Definition 5.5: Let $k \geq 2$ be a positive integer. The *existential k -pebble game* (or, in short, the $(\exists, k)$ -pebble game) is played between two players, the Spoiler and the Duplicator, on two relational structures $\mathbf{A}$ and $\mathbf{B}$ according to the following rules: each player has k pebbles labeled $1, \dots, k$; on the i -th move of a round of the game, $1 \leq i \leq k$, the Spoiler places a pebble on an element a_i of $\mathbf{A}$, and the Duplicator responds by placing the pebble with the same label on an element b_i of $\mathbf{B}$. The Spoiler wins the game at the end of that round, if the correspondence $a_i \mapsto b_i$, $1 \leq i \leq k$, is not a homomorphism between the substructures of $\mathbf{A}$ and $\mathbf{B}$ with universes $\{a_1, \dots, a_k\}$ and $\{b_1, \dots, b_k\}$, respectively. Otherwise, the Spoiler removes one or more pebbles, and a new round of the game begins. The Duplicator wins the $(\exists, k)$ -pebble game if he has a *winning strategy*, that is to say, a

systematic way that allows him to sustain playing “forever”, so that the Spoiler can never win a round of the game.

To illustrate this game, let $\mathbf{K}_m$ be the complete undirected graph with m nodes. For every $k \geq 2$, the Duplicator wins the $(\exists, k)$ -pebble game on $\mathbf{K}_k$ and $\mathbf{K}_{k+1}$, but the Spoiler wins the $(\exists, k+1)$ -pebble game on $\mathbf{K}_{k+1}$ and $\mathbf{K}_k$. As another example, let $\mathbf{L}_s$ be the s -element linear order, $s \geq 2$. If $m < n$, then the Duplicator wins the $(\exists, 2)$ -pebble game on $\mathbf{L}_m$ and $\mathbf{L}_n$, but the Spoiler wins the $(\exists, 2)$ -pebble game on $\mathbf{L}_n$ and $\mathbf{L}_m$.

Note that the above description of a winning strategy for the Duplicator in the $(\exists, k)$ -pebble game is rather informal. The concept of a winning strategy can be made precise, however, in terms of families of partial homomorphisms with appropriate closure and extension properties, where a *partial homomorphism from $\mathbf{A}$ to $\mathbf{B}$* is a homomorphism from a substructure of $\mathbf{A}$ to a substructure of $\mathbf{B}$.

Definition 5.6: Let k be a positive integer. A winning strategy for the Duplicator in the existential k -pebble game on $\mathbf{A}$ and $\mathbf{B}$ is a nonempty family $\mathcal{F}$ of partial homomorphisms from $\mathbf{A}$ to $\mathbf{B}$ such that:

1. For every $f \in \mathcal{F}$, the domain $\text{dom}(f)$ of f has at most k elements.
2. $\mathcal{F}$ is closed under subfunctions, which means that if $g \in \mathcal{F}$ and $f \subseteq g$, then $f \in \mathcal{F}$.
3. $\mathcal{F}$ has the k -forth property, which means that for every $f \in \mathcal{F}$ with $|\text{dom}(f)| < k$ and every $a \in A$ on which f is undefined, there is a $g \in \mathcal{F}$ that extends f and is defined on a .

Intuitively, the second condition provides the Duplicator with a “good” move when the Spoiler removes a pebble from an element $\mathbf{A}$, while the third condition provides the Duplicator with a “good” move when the Spoiler places a pebble on an element of $\mathbf{A}$.

For every positive integer k , let k -Datalog be the collection of all Datalog programs such that each rule has at most k distinct variables. The next result describes the connection between $(\exists, k)$ -pebble games and k -Datalog, and also gives some of the algorithmic properties of $(\exists, k)$ -pebble games.

Theorem 5.7: [KV95; KV00a] Let k be a positive integer.

- Assume that Q is a query definable by a k -Datalog program. If $\mathbf{A}$ and $\mathbf{B}$ are two relational structures such that $\mathbf{A}$ satisfies Q and the Duplicator wins the $(\exists, k)$ -pebble game on $\mathbf{A}$ and $\mathbf{B}$, then also $\mathbf{B}$ satisfies Q .
- There is a polynomial-time algorithm to decide whether, given two finite structures $\mathbf{A}$ and $\mathbf{B}$, the Spoiler or the Duplicator wins the $(\exists, k)$ -pebble game on $\mathbf{A}$ and $\mathbf{B}$.
- For every finite relational structure $\mathbf{B}$, there is a k -Datalog program $\pi_{\mathbf{B}}$ that expresses the query: given a finite σ -structure $\mathbf{A}$, does the Spoiler win the $(\exists, k)$ -pebble game on $\mathbf{A}$ and $\mathbf{B}$?

The preceding Theorem 5.7 can be used to characterize when $\overline{\text{CSP}(\mathbf{B})}$ is expressible in Datalog.

Theorem 5.8: [KV00a] Let k be a positive integer and $\mathbf{B}$ a finite relational structure. The following statements are equivalent.

- $\overline{\text{CSP}(\mathbf{B})}$ is expressible in k -Datalog
- $\text{CSP}(\mathbf{B}) = \{\mathbf{A} : \text{Duplicator wins the } (\exists, k)\text{-pebble game on } \mathbf{A} \text{ and } \mathbf{B}\}$.

Note that if a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ exists, then the Duplicator wins the $(\exists, k)$ -pebble game on $\mathbf{A}$ and $\mathbf{B}$ using the values of the homomorphism as his strategy. Thus, Theorem 5.8 asserts that $\overline{\text{CSP}(\mathbf{B})}$ is expressible in k -Datalog if and only if the following property holds: whenever the Duplicator wins the $(\exists, k)$ -pebble on $\mathbf{A}$ and $\mathbf{B}$, a homomorphism from $\mathbf{A}$ to $\mathbf{B}$ exists.

A striking consequence of Theorems 5.7 and 5.8 is that all non-uniform constraint satisfaction problems $\text{CSP}(\mathbf{B})$ for which $\overline{\text{CSP}(\mathbf{B})}$ is expressible in k -Datalog can be solved in polynomial time using the same algorithm, namely, the polynomial-time algorithm for determining the winner in the $(\exists, k)$ -pebble game. Moreover, since expressibility in Datalog subsumes the tractable cases in which $\text{Pol}(\mathbf{B})$ contains a near unanimity function or a set function, it follows that the algorithm for determining the winner in the $(\exists, k)$ -pebble game can also be used to solve $\text{CSP}(\mathbf{B})$ in polynomial time in these two major tractable cases.

Many heuristic algorithms for constraint satisfaction involve “constraint propagation”, which can be intuitively described as the derivation of new constraints from the original ones. This process has been formalized using various *consistency* concepts that make explicit additional constraints implied by the original constraints. The *strong k -consistency* property is the most important one among them; in informal terms, this property holds when every partial solution on fewer than k variables can be extended to a solution on k variables [Dec92b]. Closely related to this is the process of *establishing strong k -consistency*, which is the question of whether additional constraints can be added to a given instance of the CONSTRAINT SATISFACTION PROBLEM in such a way that the resulting instance is strongly k -consistent and has the same space of solutions as the original one (see [Dec92b; KV00b] for the formal definitions). It turns out that the following tight connection exists between strong k -consistency properties and $(\exists, k)$ -pebble games.

Theorem 5.9: [KV00b] The following are equivalent for a CSP-instance (V, D, C) .

- It is possible to establish strong k -consistency for (V, D, C) .
- The Duplicator wins the $(\exists, k)$ -pebble game on $\mathbf{A}$ and $\mathbf{B}$, where $\mathbf{A}, \mathbf{B}$ form the instance of the HOMOMORPHISM PROBLEM associated with (V, D, C) .

Furthermore, when this happens, the set $\mathcal{W}^k(\mathbf{A}, \mathbf{B})$ of all winning configurations for the Duplicator in the $(\exists, k)$ -pebble game gives rise to the largest (and, hence, least constrained) CSP-instance that establishes strong k -consistency for (V, D, C) .

The preceding theorem provides a different perspective on consistency properties and reinforces the usefulness of the

$(\exists, k)$ -pebble games in the study of constraint satisfaction. In particular, Theorem 5.9 implies that computing winning strategies for the Duplicator in the $(\exists, k)$ -pebble game is the most general form of constraint propagation. Moreover, as an immediate consequence of Theorems 5.8 and 5.9, we obtain another characterization of when $\overline{\text{CSP}(\mathbf{B})}$ is expressible in k -Datalog.

Corollary 5.10: *Let k be a positive integer and $\mathbf{B}$ be a fixed finite relational structure. Then the following are equivalent.*

- $\overline{\text{CSP}(\mathbf{B})}$ is expressible in k -Datalog
- For every finite relational structure $\mathbf{A}$, establishing strong k -consistency for $\mathbf{A}$ and $\mathbf{B}$ implies that there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$.

The results presented in this section make a strong case for the importance of the property “ $\overline{\text{CSP}(\mathbf{B})}$ in expressible in Datalog” as a broad sufficient condition for tractability of $\text{CSP}(\mathbf{B})$, since this property subsumes numerous tractable cases of non-uniform constraint satisfaction. In view of this, one would like to have an efficient algorithm to test whether, given a relational structure $\mathbf{B}$, we have that $\overline{\text{CSP}(\mathbf{B})}$ is expressible in Datalog. Such an algorithm could be used as a building block in heuristic algorithms for constraint satisfaction, where one first tests for expressibility in Datalog before resorting to some other exhaustive search procedure. At present, however, no such efficient algorithm is known. In fact, it is not even known whether expressibility of $\overline{\text{CSP}(\mathbf{B})}$ in Datalog is a decidable property. More precisely, the following problem, originally posed by Feder and Vardi [FV98], remains open.

Open Problem: Let $k \geq 2$ be a fixed positive integer. Is there an algorithm to decide whether, given $\mathbf{B}$, we have that $\overline{\text{CSP}(\mathbf{B})}$ is expressible in k -Datalog?

6 Uniform Constraint Satisfaction and Bounded Treewidth

In Section 5, we focused on the pursuit of islands of tractability of non-uniform constraint satisfaction, that is, islands of the form

$$\text{CSP}(\mathbf{B}) = \text{CSP}(\text{All}, \{\mathbf{B}\}),$$

where $\mathbf{B}$ is a fixed relational structure. In particular, we saw that substantial progress has been made towards obtaining a complete classification of all relational structures $\mathbf{B}$ for which $\text{CSP}(\mathbf{B})$ is solvable in polynomial time. The state of affairs, however, is different for constraint satisfaction problems of the form $\text{CSP}(\mathcal{A}, \mathcal{B})$, where neither $\mathcal{A}$ nor $\mathcal{B}$ is a singleton class. We call such problems *uniform constraint satisfaction problems*, because an instance of $\text{CSP}(\mathcal{A}, \mathcal{B})$ is a pair of structures $\mathbf{A} \in \mathcal{A}$ and $\mathbf{B} \in \mathcal{B}$ (unlike $\text{CSP}(\mathbf{B})$, where an instance is just a structure $\mathbf{A}$).

At present, we are far away from even coming close to a characterization of all classes $\mathcal{A}$ and $\mathcal{B}$ such that $\text{CSP}(\mathcal{A}, \mathcal{B})$ is solvable in polynomial time. Nonetheless, significant progress has been made in the case in which $\mathcal{B}$ is the class *All* of all relational structures over an arbitrary, but fixed, vocabulary. In what follows, we present the main results concerning

islands of tractability of uniform constraint satisfaction of the form $\text{CSP}(\mathcal{A}, \text{All})$.

As is well known, many algorithmic problems that are “hard” on arbitrary graphs become “easy” on trees. This motivated researchers to investigate whether the concept of a tree can be appropriately relaxed while maintaining good computational behavior. As part of their seminar work on graph minors, Robertson and Seymour introduced the concept of *treewidth* and showed that graphs of *bounded treewidth* are “tree-like” structures exhibiting such good behavior. The monograph by Downey and Fellows [DF99] contains complete definitions and characterizations of these concepts. Here, instead of giving the standard definition of treewidth in terms of *tree compositions*, we present an equivalent one in terms of *partial k -trees*.

Definition 6.1: Let k be a positive integer.

- A graph $\mathbf{H}$ is a *k -tree* if either $\mathbf{H}$ is a $\mathbf{K}_{k+1}$ -clique or there are a k -tree $\mathbf{G}$, nodes $a_1, \dots, a_k$ in $\mathbf{G}$ forming a $\mathbf{K}_k$ -clique, and a node b in $\mathbf{H} - \mathbf{G}$ such that $\mathbf{H}$ is obtained from $\mathbf{G}$ by connecting b to each of the nodes $a_1, \dots, a_k$ (thus, forming a $\mathbf{K}_{k+1}$ -clique).
- A graph is a *partial k -tree* if it is a subgraph (not necessarily induced) of a k -tree.
- The *treewidth* of a graph $\mathbf{H}$, denoted by $\text{tw}(\mathbf{H})$, is the smallest k such that $\mathbf{H}$ is a partial k -tree.
- We write $\mathcal{T}(k)$ to denote the class of all graphs $\mathbf{H}$ such that $\text{tw}(\mathbf{H}) < k$.

Clearly, if $\mathbf{T}$ is a tree, then $\text{tw}(\mathbf{T}) = 1$. Similarly, if $n \geq 3$ and $\mathbf{C}_n$ is the n -element cycle, then $\text{tw}(\mathbf{C}_n) = 2$. At the other end of the scale, $\text{tw}(\mathbf{K}_k) = k - 1$, for every $k \geq 2$. Computing the treewidth of a graph is an intractable problem. Specifically, the following problem is NP-complete: given a graph $\mathbf{H}$ and an integer $k \geq 1$, is $\text{tw}(\mathbf{H}) \leq k$? Nonetheless, Bodlaender [Bod93] showed that for every fixed integer $k \geq 1$, there is a linear-time algorithm such that, given a graph $\mathbf{H}$, it determines whether or not $\text{tw}(\mathbf{H}) < k$.

The notion of treewidth can be defined for arbitrary finite relational structures $\mathbf{A} = (A, R_1^{\mathbf{A}}, \dots, R_m^{\mathbf{A}})$ using the *Gaifman graph* $\mathbf{G}^{\mathbf{A}}$ of the structure $\mathbf{A}$. Specifically, $\text{tw}(\mathbf{A}) = \text{tw}(\mathbf{G}^{\mathbf{A}})$, where $\mathbf{G}^{\mathbf{A}} = (A, E)$ and

$$E(a, a') \iff a, a' \text{ occur in a tuple of } R_i^{\mathbf{A}}, \text{ for some } i.$$

In what follows, $\mathcal{T}(k)$ will denote the class of all relational structures of treewidth less than k over a fixed relational vocabulary.

Dechter and Pearl [DP89] and Freuder [Fre90] showed that the classes of structures of bounded treewidth give rise to large islands of tractability of uniform constraint satisfaction.

Theorem 6.2: [DP89; Fre90] *If $k \geq 2$ is a positive integer, then $\text{CSP}(\mathcal{T}(k), \text{All})$ is in P.*

The polynomial-time algorithm for $\text{CSP}(\mathcal{T}(k), \text{All})$ in the above theorem is often described as a *bucket elimination algorithm* [Dec99]. It should be noted that it is not a constraint propagation algorithm. Instead, this algorithm uses the bound on the treewidth to test if a solution to the constraint satisfaction problem exists by solving a join evaluation problem in which all intermediate relations are of bounded arity.

Kolaitis and Vardi [KV00b], and Dalmau, Kolaitis and Vardi [DKV02] investigated certain logical aspects of the treewidth of a relational structure and showed that this combinatorial concept is closely connected to the canonical conjunctive query of the structure being definable in a fragment of first-order logic with a fixed number of variables. This made it possible to show that the tractability of $\text{CSP}(\mathcal{T}(k), \text{All})$ can be explained in terms of expressibility in k -Datalog. Moreover, it led to the discovery of larger islands of tractability of uniform constraint satisfaction.

Definition 6.3: Let $k \geq 2$ be a positive integer.

- FO^k is the collection of all first-order formulas with at most k distinct variables.
- L^k is the collection of all FO^k -formulas built using atomic formulas, conjunction, and existential first-order quantification only.

As an example, it is easy to see that if $\mathbf{C}_n$ is the n -element cycle, $n \geq 3$, then the canonical conjunctive query $Q^{\mathbf{C}_n}$ is expressible in L^3 . For instance, $Q^{\mathbf{C}_4}$ is logically equivalent to

$$(\exists x \exists y \exists z)(E(x, y) \wedge E(y, z) \wedge (\exists y)(E(z, y) \wedge E(y, x))).$$

As mentioned earlier, for every $n \geq 3$, we have that $\text{tw}(\mathbf{C}_n) = 2$. The next result shows that this relationship between treewidth and number of variables needed to express the canonical conjunctive query is not an accident.

Theorem 6.4: [KV00b; DKV02] Let $k \geq 2$ be a positive integer.

- If $\mathbf{A} \in \mathcal{T}(k)$, then the canonical conjunctive query $Q^{\mathbf{A}}$ is expressible in L^k .
- If $\mathbf{B}$ is an arbitrary, but fixed, structure, then $\text{CSP}(\mathcal{T}(k), \{\mathbf{B}\})$ is expressible in k -Datalog.
- $\text{CSP}(\mathcal{T}(k), \text{All})$ can be solved in polynomial time by determining whether, given a structure $\mathbf{A} \in \mathcal{T}(k)$ and an arbitrary structure $\mathbf{B}$, the Spoiler or the Duplicator wins the $(\exists, k)$ -pebble on $\mathbf{A}$ and $\mathbf{B}$.

As a consequence of the above Theorem 6.4, we see that $\text{CSP}(\mathcal{T}(k), \text{All})$ can be solved in polynomial time using a constraint propagation algorithm that is quite different from the bucket elimination algorithm in Theorem 6.2.

Definition 6.5: Let $\mathbf{A}$ and $\mathbf{B}$ be two relational structures.

- We say that $\mathbf{A}$ and $\mathbf{B}$ are *homomorphically equivalent*, denoted $\mathbf{A} \sim_h \mathbf{B}$, if there are homomorphisms $h : \mathbf{A} \rightarrow \mathbf{B}$ and $h' : \mathbf{B} \rightarrow \mathbf{A}$.
- We say that $\mathbf{B}$ is the *core* of $\mathbf{A}$, and write $\text{core}(\mathbf{A}) = \mathbf{B}$, if
 1. $\mathbf{B}$ is a substructure of $\mathbf{A}$.
 2. There is no homomorphism $h : \mathbf{B} \rightarrow \mathbf{B}'$ from $\mathbf{B}$ to a proper substructure $\mathbf{B}'$ of $\mathbf{B}$.

Clearly, $\text{core}(\mathbf{K}_k) = \mathbf{K}_k$ and $\text{core}(\mathbf{C}_n) = \mathbf{C}_n$. Moreover, if $\mathbf{H}$ is a 2-colorable graph, then $\text{core}(\mathbf{H}) = \mathbf{K}_2$. It should be noted that cores play an important role in database query processing and optimization (see [CM77]). The next result shows that they can also be used to characterize when the canonical conjunctive query is definable in L^k .

Theorem 6.6: Let $k \geq 2$ be a positive integer and $\mathbf{A}$ a relational structure. Then the following statements are equivalent.

- $Q^{\mathbf{A}}$ is definable in L^k .
- There is a structure $\mathbf{B} \in \mathcal{T}(k)$ such that $\mathbf{A} \sim_h \mathbf{B}$.
- $\text{core}(\mathbf{A}) \in \mathcal{T}(k)$.

Definition 6.7: If $k \geq 2$ is a positive integer, then $\mathcal{H}(\mathcal{T}(k))$ is the class of all relational structures over some fixed vocabulary that are homomorphically equivalent to a structure in $\mathcal{T}(k)$.

Equivalently, $\mathcal{H}(\mathcal{T}(k))$ is the class of all relational structures $\mathbf{A}$ such that $\text{core}(\mathbf{A})$ has treewidth less than k .

It should be noted that $\mathcal{T}(k)$ is properly contained in $\mathcal{H}(\mathcal{T}(k))$. Indeed, it is known that there are 2-colorable graphs of arbitrarily large treewidth. In particular, *grids* are known to have these properties (see [DF99]). Yet, these graphs are members of $\mathcal{H}(\mathcal{T}(2))$, since their core is $\mathbf{K}_2$.

Theorem 6.8: [DKV02] Let $k \geq 2$ be a positive integer.

- If $\mathbf{B}$ is an arbitrary, but fixed, structure, then $\text{CSP}(\mathcal{H}(\mathcal{T}(k)), \{\mathbf{B}\})$ is expressible in k -Datalog.
- $\text{CSP}(\mathcal{H}(\mathcal{T}(k)), \text{All})$ is in P . Moreover, $\text{CSP}(\mathcal{H}(\mathcal{T}(k)), \text{All})$ can be solved in polynomial time by determining whether, given a structure $\mathbf{A} \in \mathcal{H}(\mathcal{T}(k))$ and an arbitrary structure $\mathbf{B}$, the Spoiler or the Duplicator wins the $(\exists, k)$ -pebble on $\mathbf{A}$ and $\mathbf{B}$.

The preceding Theorem 6.8 yields new islands of tractability for uniform constraint satisfaction that properly subsume the islands of tractability constituted by the classes of structures of bounded treewidth. However, this expansion of the tractability landscape comes at a certain price. Specifically, as seen earlier, for every fixed $k \geq 2$, there is a polynomial-time algorithm for determining membership in $\mathcal{T}(k)$ [Bod93]. In contrast, it has been shown that, for every fixed $k \geq 2$, determining membership in $\mathcal{H}(\mathcal{T}(k))$ is an NP-complete problem [DKV02]. Thus, these new islands of tractability are not as easily accessible as the earlier ones.

Since $\mathcal{H}(\mathcal{T}(k))$ contains structures of arbitrarily large treewidth, the bucket elimination algorithm cannot be used to solve $\text{CSP}(\mathcal{H}(\mathcal{T}(k)), \text{All})$ in polynomial time. Thus, Theorem 6.8 also shows that determining the winner of the $(\exists, k)$ -pebble is a polynomial time algorithm that applies to islands of tractability not covered by the bucket elimination algorithm.

It is now natural to ask whether there are classes $\mathcal{A}$ of relational structures over some fixed vocabulary such that they are larger than the classes $\mathcal{H}(\mathcal{T}(k))$ and $\text{CSP}(\mathcal{A}, \text{All})$ is solvable in polynomial time. A rather unexpected and remarkable new result by Grohe [Gro03] essentially shows that *no* such classes exist, provided a certain complexity-theoretic hypothesis is true.

Theorem 6.9: [Gro03] Assume that $\text{FPT} \neq \text{W}[1]$. If $\mathcal{A}$ is a recursively enumerable class of relational structures over some fixed vocabulary such that $\text{CSP}(\mathcal{A}, \text{All})$ is in P , then there is a positive integer k such that $\mathcal{A} \subseteq \mathcal{H}(\mathcal{T}(k))$.

The hypothesis $FPT \neq W[1]$ is a statement in *parametrized complexity* that is analogous to the hypothesis $P \neq NP$, and it is widely accepted as being true (see [DF99]). In effect, Grohe's Theorem 6.9 is a converse to Theorem 6.8. Together, these two theorems yield a complete characterization of all islands of tractability of the form $CSP(\mathcal{A}, All)$. Moreover, they reveal that all tractable cases of the form $CSP(\mathcal{A}, All)$ can be solved by the same polynomial-time algorithm, namely, the algorithm for determining the winner in the $(\exists, k)$ -pebble game. In other words, all tractable cases of constraint satisfaction of the form $CSP(\mathcal{A}, All)$ can be solved in polynomial time using constraint propagation.

References

- [Bib88] W. Bibel. Constraint satisfaction from a deductive viewpoint. *Artificial Intelligence*, 35:401–413, 1988.
- [Bod93] H.L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 226–234, 1993.
- [Bul02a] A. Bulatov. A dichotomy theorem for constraints on a three-element set. In *Proc. 43rd IEEE Symposium on Foundations of Computer Science*, pages 649–658, 2002.
- [Bul02b] A. Bulatov. Maltsev constraints are tractable. Technical Report PRG-RR-02-05, Oxford University, 2002.
- [Bul03] A. Bulatov. Tractable conservative constraint satisfaction problems. In *Proc. 18th IEEE Symposium on Logic in Computer Science*, 2003.
- [CM77] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proc. 9th ACM Symp. on Theory of Computing*, pages 77–90, 1977.
- [Dec92a] R. Dechter. Constraint networks. In S.C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 276–185. Wiley, New York, 1992.
- [Dec92b] R. Dechter. From local to global consistency. *Artificial Intelligence*, 55(1):87–107, May 1992.
- [Dec99] R. Dechter. Bucket elimination: a unifying framework for reasoning. *Artificial Intelligence*, 113(1–2):41–85, 1999.
- [DF99] R.G. Downey and M.R. Fellows. *Parametrized Complexity*. Springer-Verlag, 1999.
- [DKV02] V. Dalmau, Ph.G. Kolaitis, and M.Y. Vardi. Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Proc. of the Eighth Int. Conference on Principles and Practice of Constraint Programming (CP 2002)*, Lecture Notes in Computer Science, pages 311–326. Springer, 2002.
- [DP89] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, pages 353–366, 1989.
- [EFT94] H.-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic*. Springer-Verlag, 2nd edition, 1994.
- [Fre90] E.C. Freuder. Complexity of K-tree structured constraint satisfaction problems. In *Proc. of 8th National Conference on Artificial Intelligence*, pages 4–9, 1990.
- [FV98] T. Feder and M.Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: a study through Datalog and group theory. *SIAM J. on Computing*, 28:57–104, 1998. Preliminary version in *Proc. 25th ACM Symp. on Theory of Computing*, May 1993, pp. 612–622.
- [GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability - A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 1979.
- [Gro03] M. Grohe. The complexity of homomorphism and constraint satisfaction problems seen from the other side. Submitted for publication, 2003.
- [JCC98] P. Jeavons, D. Cohen, and M.C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1–2):251–65, May 1998.
- [JCG97] P. Jeavons, D. Cohen, and M. Gyssens. Closure properties of constraints. *Journal of the ACM*, 44(4):527–48, 1997.
- [Jea98] P. Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1–2):185–204, 1998.
- [KV90] Ph. G. Kolaitis and M. Y. Vardi. 0-1 laws for infinitary logics. In *Proc. 5th IEEE Symp. on Logic in Computer Science*, pages 156–167, 1990.
- [KV95] Ph. G. Kolaitis and M. Y. Vardi. On the expressive power of Datalog: tools and a case study. *Journal of Computer and System Sciences*, 51(1):110–134, August 1995.
- [KV00a] Ph.G. Kolaitis and M.Y. Vardi. Conjunctive-query containment and constraint satisfaction. *Journal of Computer and System Sciences*, pages 302–332, 2000. Earlier version in: *Proc. 17th ACM Symp. on Principles of Database Systems (PODS '98)*.
- [KV00b] Ph.G. Kolaitis and M.Y. Vardi. A game-theoretic approach to constraint satisfaction. In *Proc. of the 17th National Conference on Artificial Intelligence (AAAI 2000)*, pages 175–181, 2000.
- [Lad75] R. E. Ladner. On the structure of polynomial time reducibility. *Journal of the Association for Computing Machinery*, 22(1):155–171, 1975.
- [Mon74] U. Montanari. Networks of constraints: fundamental properties and application to picture processing. *Information Science*, 7:95–132, 1974.
- [PJ97] J. Pearson and P. Jeavons. A survey of tractable constraint satisfaction problems. Technical Report CSD-TR-97-15, Royal Holloway University of London, 1997.
- [Sch78] T.J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th ACM Symp. on Theory of Computing*, pages 216–226, 1978.
- [Ull89] J. D. Ullman. *Database and Knowledge-Base Systems, Volumes I and II*. Computer Science Press, 1989.

Web Intelligence (WI): What Makes Wisdom Web?

(Invited Talk at IJCAI'03)

Jiming Liu

Web Intelligence Consortium (WIC) &
Computer Science Department, Hong Kong Baptist University
Kowloon Tong, Hong Kong jiming@comp.hkbu.edu.hk

Abstract

Web Intelligence (WI) sheds new light on directions for scientific research and development which explores the fundamental roles as well as practical impacts of Artificial Intelligence (AI) and advanced Information Technology (IT) on the next generation of Web-empowered products, systems, services, and activities. This paper gives new perspectives on the future WI research and highlights some of the research challenges and initiatives.

The notion of Web Intelligence (WI) was first introduced in [10; 17; 18; 19; 21; 22]. Since then, WI has become a fast growing **research field** which concerns scientific research and development to explore the fundamental roles as well as practical impacts of Artificial Intelligence (AI) and advanced Information Technology (IT) on the next generation of Web-empowered systems, services, and environments. WI is *aimed* at producing theories and technologies that will enable us to *optimally* utilize the global connectivity, as offered by the Web infrastructure, in life, work, and play.

1 Been There, Done That

The Web has irrevocably revolutionized the world we live in. This impact is inevitable due to the facts that the Web connectivity rapidly increases and that the online information astronomically explodes. In order not only to live with such a change but also to benefit from the information infrastructure that the Web has empowered, we have witnessed the fast development as well as applications of many WI techniques and technologies, which cover:

1. **Internet-level communication, infrastructure, and security protocols.** The Web is regarded as a *computer-network system*. WI techniques for this level include, for instance, Web data prefetching systems built upon Web surfing patterns to resolve the issue of Web latency. The intelligence of the Web prefetching comes from adaptive learning based on observations of user surfing behavior.
2. **Interface-level multimedia presentation standards.** The Web is regarded as an *interface* for human-Internet interaction. WI techniques for this level are used to develop intelligent Web interfaces in which the capabilities of adaptive cross-language processing, person-

alized multimedia representation, and multimodal data processing are required.

3. **Knowledge-level information processing and management tools.** The Web is regarded as a *distributed data/knowledge base*. We need to develop semantic markup languages to represent the semantic contents of the Web available in machine-understandable formats for agent based computing, such as searching, aggregation, classification, filtering, managing, mining, and discovery on the Web [3].
4. **Application-level ubiquitous computing and social intelligence environments.** The Web is regarded as a *basis for establishing social networks* that contain communities of people (or organizations or other social entities) connected by social relationships, such as friendship, coworking or information exchange with common interests. They are Web-supported social networks or virtual communities. The study of WI concerns the important issues central to social network intelligence (social intelligence for short) [7; 15]. Furthermore, the multimedia contents on the Web are not only accessible from stationary platforms, but increasingly from mobile platforms [16]. Ubiquitous Web access and computing from various wireless devices need adaptive personalization for which WI techniques are used to construct models of user interests by inferring from user behavior and actions [1; 4].

In spite of on-going WI technological advances, it is still rather unclear to many *what will be the next paradigm shift in WI*. In what follows, I will present new **perspectives** on this question and at the same time point out some WI research challenges.

2 What's Next?

In the movie *Star Wars: Episode II*, there is an interesting scene: When Obi Wan Kenobi failed to locate any relevant information about a mysterious planet (where later he discovered the clone manufacturing ground), he turned to his friend for advice. His friend, who apparently knew more than the Jedi's academy knowledge banks combined, had the following in reply: Other people seek *knowledge*, but you my friend know *wisdom*.

The reply in the above scene also provides an answer to our earlier question. The next paradigm shift in WI lies in the notion of *wisdom*. The **goal** of the new generation WI is to enable users to gain new **wisdom** of living, working, playing, and learning, in addition to *information* search and *knowledge* queries. Here, the word of *wisdom*, according to the Webster Dictionary (Page: 1658) [14], implies the following meanings:

1. The quality of being wise; knowledge, and *the capacity to make due use of it; knowledge of the best ends and the best means; discernment and judgment; discretion; sagacity; skill; dexterity.*
2. The results of wise judgments; scientific or practical truth; acquired knowledge; erudition.

In the Web context, the manifestation of wisdom can best be illustrated with a minimalist **Wisdom Web** example.

3 When the Web Offers Practical Wisdom ...

Imagine that this is your first trip to the city of Montreal. You would like to find a really nice place to spend your evening. So, you walk into a Cyber Cafe on Sherbrook Street (the only street that you can recognize), and decide to get some *practical wisdom* from a public **Wisdom Web** outlet. You log in with a user name, "Spiderman", and ask:

"What is the best night life in Montreal during this season of the year?"

The Wisdom Web *thinks* for about a second or two and then responds:

"Spiderman, the hockey games are on during this season of the year. Would you like to go?"

You reply:

"Yes."

Then the Wisdom Web suggests:

"As far as I know, there are still some tickets left and you may purchase some at the Montreal Forum. It is easy to get there by taking Metro to the Atwater station."

Now you decide that this could be an interesting evening for you...

One hour later, you arrive at the ticket office by Metro, but surprisingly find that the tickets left are all for the day after tomorrow when you will be traveling in Quebec City.

As you are a bit disappointed, you notice that there is a free **Wisdom Web Kiosk** right beside the ticket office. Well, that's convenient. So, without too much hesitation, you log on to the Wisdom Web, again as "Spiderman". The Wisdom Web still remembers your conversations an hour ago. As soon as it recognizes that you are "Spiderman", it says to you:

"Hello Spiderman, you were in such a hurry last time that I couldn't have a chance to tell you that all tickets available here are only for the day after tomorrow. They are quite expensive too..."

4 10 Capabilities of the Wisdom Web

In order to make the above Wisdom Web scenario a reality, the following 10 fundamental capabilities have to be incorporated and standardized:

1. **Self-organizing servers:** The Wisdom Web will automatically regulate the functions and cooperations of related Websites and application services available. A Wisdom Web server automatically self-nominates to other services its functional *roles* as well as corresponding spatial or temporal *constraints* and operational *settings*.
2. **Specialization:** A Wisdom Web server is an *agent* by itself, which is specialized in performing some roles in a certain service. The *association* of its roles with any service will be measured and updated dynamically, for instance, the association may be forgotten if it is not used for some time.
3. **Growth:** The population of **Wisdom Agents** will dynamically change, as new agents are self-reproduced by their parent agents in order to become more specialized, or aged agents are deactivated.
4. **Autocatalysis:** As various roles of wisdom agents are created through specialization and activated by the **Wisdom Search** requests, their associations with some services and among themselves must be autocatalytically aggregated. In this respect, the autocatalysis of associations is similar to the pheromone laying for positive feedback in an ant colony.
5. **Problem Solver Markup Language (PSML):** PSML is necessary for wisdom agents to specify their roles and settings as well as relationships with any other services.
6. **Semantics:** The Wisdom Web needs to understand what are meant by "Montreal", "season", "year", and "night life", and what is the right judgment of "best", by understanding the granularities of their corresponding subjects and the whereabouts of their ontology definitions.
7. **Metaknowledge:** Besides semantic knowledge extracted and manipulated in the *Wisdom Search*, it is also essential for wisdom agents to incorporate a dynamically created source of metaknowledge that deals with the relationships between concepts and the spatial or temporal constraint knowledge in planning and executing services. It allows agents to self-resolve their conflict of interests.
8. **Planning:** In the above example, the goal is to find a function or an event that may sound attractive to a visitor. The constraint is that they must be happening during this season. There are involved two associated subgoals: In order to have an access to the recommended function or event, one needs a ticket. Further, in order to go to get the ticket, one can travel by Metro. In the Wisdom Web, ontology alone will *not* be sufficient.
9. **Personalization:** The Wisdom Web remembers the recent encounters and relates different episodes together, according to (1) "Spiderman", (2) time, and (3) attainability of (sub)goals. It may further identify other goals

as well as courses of actions for this user as their conversation goes on.

10. **A sense of humor:** Although the Wisdom Web does not explicitly tell a funny story, it adds some punch lines to the situation or anxiety that “Spiderman” is presently in when he/she logs on for the second time, which will make “Spiderman” feel absurd.

5 Research Challenges

In the preceding sections, I have stated the goal and vision of the new generation WI, and what constitutes a Wisdom Web. In this section, I will point out some WI research challenges.

5.1 Problem Solver Markup Language (PSML)

One practical Wisdom Web issue that has to be addressed is how to integrate distributed and centralized information sources/structures. Information/knowledge on the Web can be either globally distributed throughout the Web or locally hosted, centralized on an intelligent portal providing Web services (*i.e.*, an intelligent service provider) that is integrated to its own cluster of specialized intelligent applications. However, each approach has a serious flaw. As pointed out by Alesso and Smith [1], the intelligent portal approach limits uniformity and access, while the global semantic Web approach faces combinatory complexity limitations.

A way to solve the above issue is to develop and use the Problem Solver Markup Language (PSML) for collecting globally distributed contents and knowledge from Web-supported, semantic social networks and incorporating them with locally operational knowledge/databases in an enterprise or community for locally centralized, adaptable Web intelligent services.

The core of PSML is based on **distributed inference engines** that can perform automatic reasoning on the Web by incorporating contents and metaknowledge, *autonomically* collected and transformed from the semantic Web, with locally operational knowledge/databases. A feasible way to work with PSML is to use existing (Prolog-like) logic languages, together with dynamic contents and metaknowledge collection and transformation agents.

By using this information transformation approach, the dynamic, global information sources on the Web can be combined with the local information sources in an enterprise portal for decision making and business intelligence.

Ohsuga *et al.* previously used KAUS for representing local information sources and for performing inferences and reasoning. KAUS is a knowledge based system that involves Multi-Layer Logic and databases based on the relational data model [13]. KAUS enables the representations of knowledge and data in the first order logic form with multi-layer data structures. It can readily be used for inferences and reasoning as well as for transforming and managing both knowledge and data.

5.2 Automatic Ontology Construction

Ontology is the basis for designing and implementing the Semantic Web for Web intelligence. It plays a crucial role in enabling Web based knowledge processing and management,

such as Web community communication, semantics based agent communication, knowledge based Web retrieval, Web content understanding, and Web community discovery.

Although ontology engineering has been studied over the last decade, there is still a demand for more **automatic methods** in comprehensive ontology construction and learning [20]. The tedious, cumbersome task of existing manual ontology construction can easily become a bottleneck in the development of semantics based Web intelligence.

5.3 Wisdom Web Based Computing Paradigm

The paradigm of **Wisdom Web based computing** is aimed at providing not only a medium for seamless information exchange and knowledge sharing but also a type of **man-made resources for sustainable knowledge creation, and scientific and social evolution**. The Wisdom Web will rely on *grid-like service agencies* that self-organize, learn, and evolve their courses of actions in order to perform service tasks as well as their identities and interrelationships in communities. They will also cooperate and compete among themselves in order to optimize their as well as others resources and utilities.

Self-organizing, learning agents are computational entities that are capable of self-improving their performance in dynamically changing and unpredictable task environments. In [8], Liu has provided a comprehensive overview of several studies in the field of **autonomy oriented computing**, with in-depth discussions on self-organizing and adaptive techniques for developing various embodiments of agent based systems, such as autonomous robots, collective vision and motion, autonomous animation, and search and segmentation agents. The core of those techniques is the notion of synthetic or emergent autonomy based on behavioral self-organization.

5.4 A New Turing Test

In order to effectively develop the new generation WI systems, we need to define **benchmark** applications, *i.e.*, a new *Turing Test*, that will capture and demonstrate the Wisdom Web capabilities.

Take the Wisdom Web based computing benchmark as an example. We can use a service task of compiling and generating a market report on an existing product or a *potential market report on a new product*. In order to get such service jobs done, an information agent on the Wisdom Web will mine and integrate available Web information, which will in turn be passed onto a market analysis agent. Market analysis will involve the quantitative simulations of customer behavior in a marketplace, instantaneously handled by other serviced agencies, involving a large number of grid agents [2]. Since the number of variables concerned may be in the order of hundreds or thousands, it can easily cost a single system years to generate one predication.

6 Recent Initiatives

In this section, I will describe two recent WI research initiatives: (1) information foraging agents and (2) ubiquitous agent communities.

6.1 Information Foraging Agents

One of the important roles of wisdom agents is concerned with searching and filtering information from distributed Web sources. In this respect, understanding and developing the right information foraging behaviors for wisdom agents will become a real challenge. In order to tackle this challenge, the first step is to have a clear picture of how humans search information, what kind of human behavior dominates, and what is the origin of Web usage regularities. There have been few efforts on describing various aspects of human Web surfing regularities [5; 6]. However, the origin and interrelated variables of the regularities still remain unknown.

Liu and Zhang have developed a model for information foraging agents on the Web [11]. The foraging agent model takes into account Web topology, information distribution, and agent interest profile. They have validated this model using empirical Web log datasets, and used it to explain the origin of regularities in Web surfing. In particular, they have discovered that it is the unique distribution of agent interest that leads to the regularities in agent surfing behavior, *i.e.*, a power law distribution of agent surfing depth. The Web topology can only influence the shape parameters of the distribution without changing the nature of the distribution. Also discovered is that the power law of link click frequency is largely due to agent purposeful surfing behavior. This work shows that Web regularities are interrelated.

Also in their studies, they have studied three categories of foraging agents, according to their interests and familiarities with the Web: **Random agents** who have no obvious intention in Web surfing, **rational agents** who have certain goals to achieve but are not familiar with the Web structure, and **recurrent agents** who have certain specific intents and are very familiar with the Web structure. The ability to predict the contents at the next level nodes becomes stronger when moving from random to recurrent agents. Experimental results with respect to the three agent categories have unveiled that the regularities of agent surfing depth on pages and domains still remain the same, while a power law of link click frequency distribution will disappear as we move from recurrent agents to random agents. This result shows that the order existing in link click frequency comes from agent's content prediction ability, that is whether or not an agent can determine the next step according to its own interest and current information.

6.2 Ubiquitous Agent Communities

One of the key features in wisdom agents is *ubiquity*. Ubiquitous agents are autonomous entities that automatically explore and exploit WI-based Web services. There have been some earlier work performed in the domains of e-business and e-commerce. Related issues and applications in which various tasks and objectives were achieved can be found in [9].

In this area of research, a reasonable challenge to begin with would be to develop and demonstrate a **Ubiquitous Agent Community (UAC)**, *i.e.*, an intelligent infrastructure that enables agents to look ahead, plan, and deliver what a user wants [12]. It works like a personal agency. For instance, it can help a user to effectively manage tedious daily

routine activities, such as processing emails, placing orders, making meeting arrangements, downloading news, etc.

A UAC can interact with its users in various ways. For instance, as a person uses a smart handheld device to enter a subway station on his/her way to work, a UAC can seamlessly upload a list of things to be done to the person's handheld device. Along with each of the things listed, the information provided by the UAC can also include its corresponding tasks necessary to be carried out. Thus, inside the moving subway train, he/she will be able to go through each of the recommended task items and to further verify and delegate certain jobs to a community agent (c-agent) in the UAC, such as forwarding a report or making a meeting appointment. In addition to planning and executing the items on a calendar, the user can also receive other personalized services based on his/her profile, which may include sports news, urgent emails, and specific documents prepared for the day's meetings.

Figure 1 gives an illustrative example of personal services that can be provided by a community of competing and cooperating agents, *i.e.*, a Ubiquitous Agent Community (UAC).

A UAC contains two types of agents: **community agents (c-agents)** and **service agents (s-agents)**. The former makes task plans as well as contracting decisions on what and where some tasks will be carried out, whereas the latter performs the delegated tasks. In the illustrative example of Figure 1, suppose a user writes a few words on his/her PDA regarding what he/she plans to do. A c-agent in the distributed UAC will recognize the intention of the user and find similar service cases from its case base. Thereafter, it will perform task planning based on the similarity between the task at hand and the previous tasks. Once the c-agent completes its task planning, a cluster of s-agents will be called upon to distribute and carry out the planned subtasks.

In the UAC, s-agents will not act alone, but work coordinately, in order to accomplish their subtasks. Coordination is one of the most important characteristics of UAC agents. The coordination of UAC agents is essentially the coordination of their ubiquitous service actions.

Take a buyer-seller market community as an example. The goal of each agent is to maximize its own interest; agents are antagonistic in a certain sense. At the same time, the market exists on the basis of buyer-seller competitions. In this case, the coordination of the market agents is reflected in their competitions for the best prices.

Generally speaking, the coordination of UAC agents involves both competition and cooperation. UAC agents handle their services in two main stages: task allocation and task execution. As mentioned above, a service task often consists of several subtasks. Each of the subtasks will be allocated to a specific s-agent. Thus, as far as the task allocation is concerned, various s-agents will compete among themselves in order to get a subtask assignment. In this case, coordination manifests itself as competition. On the other hand, during the course of task execution, an s-agent will require assistance from other s-agents in order to effectively fulfill the task at hand. Thus, coordination also manifests itself as cooperation, as s-agents exchange information and services among each other. In many ways, the technical issues of implementing UAC can be reduced to those of implementing coordination

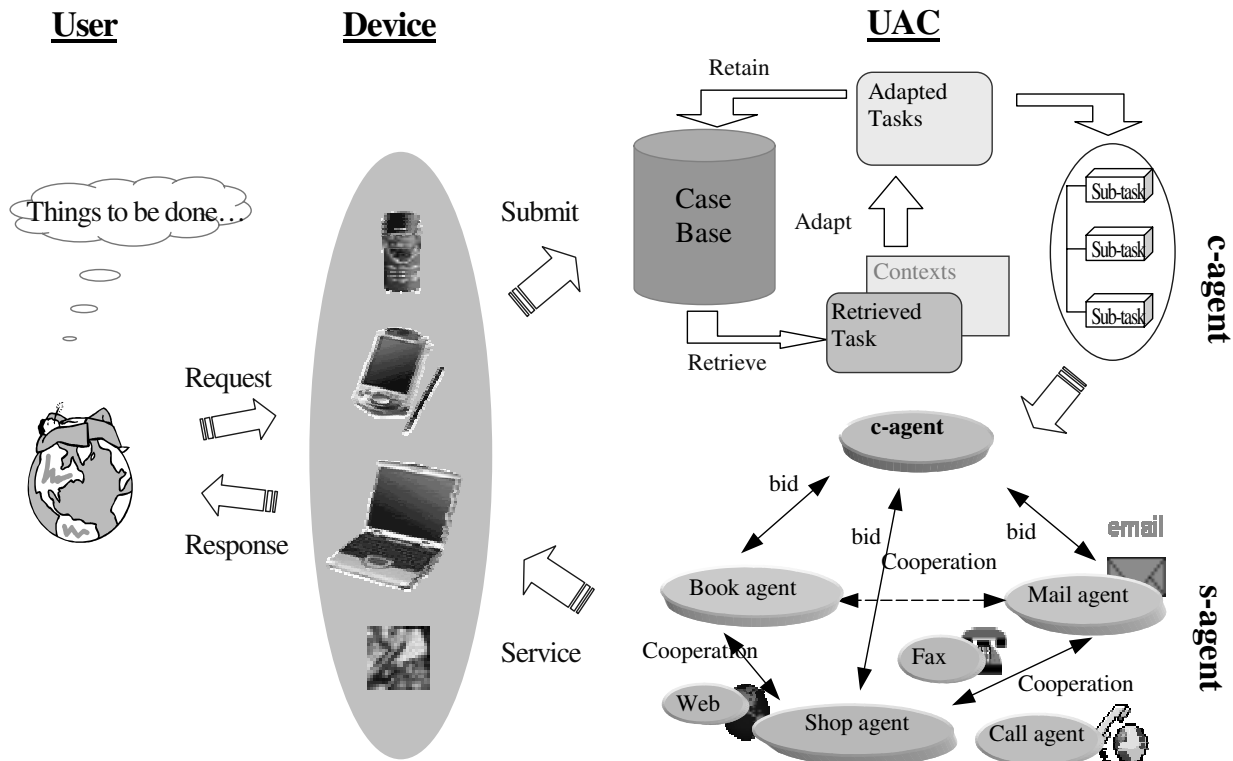


Figure 1: A scenario of services being provided by a community of competing and cooperating ubiquitous agents. The community is called a Ubiquitous Agent Community (UAC).

among ubiquitous agents.

7 Conclusions

I have started this paper by giving perspectives on the new generation WI. The Web, as an information infrastructure, will remain to create a strong impact in the future by offering world-wide presence and connectivity. The major challenge of the Web research and development in the next decade will be WI-centric, focusing on how we can intelligently make the best use of the widely available Web connectivity. The new WI technologies to be developed will be precisely determined by *human needs in a post-industrial era*; namely:

1. **information** empowerment;
2. **knowledge** sharing;
3. virtual **social** clustering;
4. **service** enrichment;
5. practical **wisdom** development.

The next paradigm shift in WI will be towards the notion of **wisdom**. Developing the **Wisdom Web** will become a tangible goal for WI research. The new generation WI will enable

humans to gain wisdom of living, working, playing, in addition to information search and knowledge queries.

I believe that our current efforts will pave the way toward the next biggest technological invention of this century, the Wisdom Web, whose significance and impact will be equivalent to, or even far beyond, those of Alexander Graham Bell's invention of the telephone in 1876.

The Web Intelligence Consortium (WIC)

To meet the strong demands for participation and the growing interests in WI, the Web Intelligence Consortium (WIC) was formed in spring 2002. The WIC (<http://wi-consortium.org/>) is an international non-profit organization dedicated to promoting world-wide scientific research and industrial development in the era of Web and agent intelligence. Its activities include collaborations with WI research centers throughout the world and organization/individual members, technology showcases at IEEE/WIC conferences and workshops, and WIC publications.

In addition to various special issues on WI published by several international journals, including IEEE Com-

puter, a WI-focused scientific journal, *Web Intelligence and Agent Systems: An International Journal* (<http://wi-consortium.org/journal.html>), has been successfully launched as the official journal of the WIC.

Acknowledgments

I would like to thank Ning Zhong and Yiyu Yao for their continuing collaborations in the development of Web Intelligence (WI). The contents of this paper include their contributions. I am very grateful to people who have joined or supported the WI community, in particular, the WIC Advisory Board members: Edward A. Feigenbaum, Setsuo Ohsuga, Benjamin Wah, Philip Yu, and Lotfi A. Zadeh, all the WIC Technical Committee Members, as well as the IEEE TCCI Chair, Xindong Wu. I thank them for their strong support.

References

- [1] H. P. Alesso and C. F. Smith. *The Intelligent Wireless Web*, Addison-Wesley, 2002.
- [2] F. Berman. From TeraGrid to knowledge grid. *Communications of the ACM*, 44:27-28, 2001.
- [3] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, 284:34-43, 2001.
- [4] D. Billsus, C. A. Brunk, C. Evans, B. Gladish, and M. Pazzani. Adaptive interfaces for ubiquitous Web access, *Communications of the ACM*, 45:34-38, 2002.
- [5] B. A. Huberman, P. L. T. Pirolli, J. E. Pitkow, and R. M. Lukose. Strong regularities in World Wide Web surfing. *Science*, 280:96-97, 1997.
- [6] B. A. Huberman and L.A. Adamic. Growth dynamics of the World-Wide Web. *Nature*, 410:131, 1999.
- [7] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins: The Web and social networks. *IEEE Computer Special Issue on Web Intelligence*, 35(11):32-36, November 2002.
- [8] J. Liu. *Autonomous Agents and Multiagent Systems*, World Scientific Publishing, 2001.
- [9] J. Liu and Y. Ye, (eds.). *Advances in E-commerce Agents: Brokerage, Negotiation, Security, and Mobility*, Springer, 2001.
- [10] J. Liu, N. Zhong, Y. Y. Yao, and Z. W. Ras. The Wisdom Web: New challenges for Web Intelligence (WI). *Journal of Intelligent Information Systems*, Kluwer Academic Publishers, 20(1):5-9, 2003.
- [11] J. Liu and S. W. Zhang. Characterizing Web usage regularities with information foraging agents. *IEEE Transactions on Knowledge and Data Engineering* (in press).
- [12] J. Liu and C. Y. Yao. Rational competition and cooperation in ubiquitous agent communities. *Knowledge Based Systems*, Elsevier Science (in press).
- [13] S. Ohsuga. Framework of knowledge based systems - Multiple meta-level architecture for representing problems and problem solving processes. *Knowledge Based Systems*, 3(4):204-214, Elsevier Science, 1990.
- [14] N. Porter (ed.). *Webster's Revised Unabridged Dictionary*, G&C. Merriam Co., 1913.
- [15] P. Raghavan. Social networks: From the Web to the enterprise. *IEEE Internet Computing*, 6(1):91-94, 2002.
- [16] M. Weiser. The future of ubiquitous computing on campus. *Communications of the ACM*, 41(1):41-42, 1998.
- [17] Y. Y. Yao, N. Zhong, J. Liu, and S. Ohsuga. Web Intelligence (WI): Research challenges and trends in the new information age. N. Zhong, Y. Y. Yao, J. Liu, and S. Ohsuga (eds.), *Web Intelligence: Research and Development*, LNAI 2198, pages 1-17, Springer, 2001.
- [18] N. Zhong, J. Liu, Y. Y. Yao, and S. Ohsuga. Web Intelligence (WI). In *Proceedings of the 24th IEEE Computer Society International Computer Software and Applications Conference (COMPSAC 2000)*, pages 469-470, IEEE Computer Society Press, Taipei, Taiwan, October 25-28, 2000.
- [19] N. Zhong, Y. Y. Yao, J. Liu, and S. Ohsuga (eds.). *Web Intelligence: Research and Development*, LNAI 2198, Springer, 2001.
- [20] N. Zhong. Representation and construction of ontologies for Web Intelligence. *International Journal of Foundations of Computer Science (IJFCS)*, 13(4):555-570, World Scientific Publishing, 2002.
- [21] N. Zhong, J. Liu, and Y. Y. Yao. In search of the Wisdom Web. *IEEE Computer*, 35(11):27-31, November 2002.
- [22] N. Zhong, J. Liu, and Y. Y. Yao (eds.). *Web Intelligence*, Springer, 2003.

Self-reconfiguring Robots: Successes and Challenges

Daniela Rus

Department of Computer Science
Dartmouth College, Hanover NH 03755, USA
rus@cs.dartmouth.edu

Abstract

We wish to create versatile robots by using self-reconfiguration: hundreds of small modules autonomously organize and reorganize as geometric structures to best fit the terrain on which the robot has to move, the shape of the object the robot has to manipulate, or the sensing needs for the given task. Self-reconfiguration allows large collections of small robots to actively organize as the most optimal geometric structure to perform useful coordinated work.

A self-reconfiguring robot consists of a set of identical modules that can dynamically and autonomously reconfigure in a variety of shapes, to best fit the terrain, environment, and task. Self-reconfiguration leads to versatile robots that can support multiple modalities of locomotion and manipulation. Self-reconfiguring robots constitute large scale distributed systems. Because the modules change their location continuously they also constitute ad-hoc networks.

This talk will discuss the challenges and successes of creating self-reconfiguring robots, ranging from designing hardware capable of self-reconfiguration to developing distributed controllers and planners for such systems that are scalable, adaptive, and support real-time behavior.

Acknowledgments

Support for this work was provided through NSF awards IRI-9714332, EIA-9901589, IIS-9818299, IIS-9912193, EIA-0202789 and 0225446, ONR award N00014-01-1-0675 and DARPA Task Grant F-30602-00-2-0585.

Automated Verification: Graphs, Logic, and Automata

Moshe Y. Vardi*

Rice University

Houston, TX, USA

vardi@cs.rice.edu

<http://www.cs.rice.edu/~vardi>

Abstract

Automated verification is one of the most successful applications of automated reasoning in computer science. In automated verification one uses algorithmic techniques to establish the correctness of the design with respect to a given property. Automated verification is based on a small number of key algorithmic ideas, tying together graph theory, automata theory, and logic. In this self-contained talk I will describe how this “holy trinity” gave rise to automated-verification tools, and mention some applications to planning.

1 Introduction

The recent growth in computer power and connectivity has changed the face of science and engineering, and is changing the way business is being conducted. This revolution is driven by the unrelenting advances in semiconductor manufacturing technology. Nevertheless, the U.S. semiconductor community faces a serious challenge: chip designers are finding it increasingly difficult to keep up with the advances in semiconductor manufacturing. As a result, they are unable to exploit the enormous capacity that this technology provides. The International Technology Roadmap for Semiconductors suggests that the semiconductor industry will require productivity gains greater than the historical 20% per-year to keep up with the increasing complexity of semiconductor designs. This is referred to as the “design productivity crisis”. As designs grow more complex, it becomes easier to introduce flaws into the design. Thus, designers use various validation techniques to verify the correctness of the design. Unfortunately, these techniques themselves grow more expensive and difficult with design complexity. As the validation process has begun to consume more than half the project design resources, the semiconductor industry has begun to refer to this problem as the “validation crisis”.

Formal verification is a process in which mathematical techniques are used to guarantee the correctness of a design with respect to some specified behavior. Automated

formal-verification tools, based on *model-checking technology* [Clarke *et al.*, 1986; Lichtenstein and Pnueli, 1985; Queille and Sifakis, 1981; Vardi and Wolper, 1986] have enjoyed a substantial and growing use over the last few years, showing an ability to discover subtle flaws that result from extremely improbable events [Clarke *et al.*, 1999]. While until recently these tools were viewed as of academic interest only, they are now routinely used in industrial applications, resulting in decreased time to market and increased product integrity [Kurshan, 1997]. It is fair to say that automated verification is one of the most successful applications of automated reasoning in computer science [Clarke and Wing, 1996].

2 Basic Theory

The first step in formal verification is to come up with a *formal specification* of the design, consisting of a description of the desired behavior. One of the more widely used specification languages for designs is *temporal logic* [Pnueli, 1977]. In *linear* temporal logics, time is treated as if each moment in time has a unique possible future. Thus, linear temporal formulas are interpreted over linear sequences, and we regard them as describing the behavior of a single computation of a system. (An alternative approach is to use *branching* time. For a discussion of linear vs. branching time, see [Vardi, 2001].)

In the linear temporal logic LTL, formulas are constructed from a set *Prop* of atomic propositions using the usual Boolean connectives as well as the unary temporal connective *X* (“next”), *F* (“eventually”), *G* (“always”), and the binary temporal connective *U* (“until”). For example, the LTL formula $G(\text{request} \rightarrow F \text{ grant})$, which refers to the atomic propositions *request* and *grant*, is true in a computation precisely when every state in the computation in which *request* holds is followed by some state in the future in which *grant* holds. The LTL formula $G(\text{request} \rightarrow (\text{request } U \text{ grant}))$ is true in a computation precisely if, whenever *request* holds in a state of the computation, it holds until a state in which *grant* holds is reached.

LTL is interpreted over *computations*, which can be viewed as infinite sequences of truth assignments to the atomic propositions; i.e., a computation is a function $\pi : \mathbb{N} \rightarrow 2^{\text{Prop}}$ that assigns truth values to the elements of *Prop* at each time instant (natural number). For a computation π and a point $i \in \mathbb{N}$, the notation $\pi, i \models \varphi$ indicates that a formula

*Supported in part by NSF grants CCR-9988322, CCR-0124077, IIS-9908435, IIS-9978135, and EIA-0086264, by BSF grant 9800096, and by a grant from the Intel Corporation.

φ holds at the point i of the computation π . For example, $\pi, i \models X\varphi$ iff $\pi, i+1 \models \varphi$, and $\pi, i \models \varphi U \psi$ iff for some $j \geq i$, we have $\pi, j \models \psi$ and for all $k, i \leq k < j$, we have $\pi, k \models \varphi$. We say that π *satisfies* a formula φ , denoted $\pi \models \varphi$, iff $\pi, 0 \models \varphi$. The connectives F and G can be defined in terms of the connective U : $F\varphi$ is defined as $\text{true } U \varphi$, and $G\varphi$ is defined as $\neg F\neg\varphi$.

Designs can be described in a variety of formal description formalisms. Regardless of the formalism used, a *finite-state design* can be abstractly viewed as a *labeled transition system*, i.e., as a structure of the form $M = (W, W_0, R, V)$, where W is the finite set of states that the system can be in, $W_0 \subseteq W$ is the set of initial states of the system, $R \subseteq W^2$ is a transition relation that indicates the allowable state transitions of the system, and $V : W \rightarrow 2^{Prop}$ assigns truth values to the atomic propositions in each state of the system. (A labeled transition system is essentially a Kripke structure.) A *path* in M that *starts at* u is a possible infinite behavior of the system starting at u , i.e., it is an infinite sequence $u_0, u_1 \dots$ of states in W such that $u_0 = u$, and $u_i R u_{i+1}$ for all $i \geq 0$. The sequence $V(u_0), V(u_1) \dots$ is a *computation* of M that *starts at* u . It is the sequence of truth assignments visited by the path. The *language* of M , denoted $L(M)$ consists of all computations of M that start at a state in W_0 . Note that $L(M)$ can be viewed as a language of infinite words over the alphabet 2^{Prop} . $L(M)$ can be viewed as an abstract description of a system, describing all possible “traces”. We say that M *satisfies* an LTL formula φ if all computations in $L(M)$ satisfy φ , that is, if $L(M) \subseteq \text{models}(\varphi)$. When M satisfies φ we also say that M is a model of φ , which explain why the technique is known as *model checking* [Clarke et al., 1999].

One of the major approaches to automated verification is the *automata-theoretic approach*, which underlies many model checkers, e.g., SPIN [Holzmann, 1997]. The key idea underlying the automata-theoretic approach is that, given an LTL formula φ , it is possible to construct a finite-state automaton A_φ on infinite words that accepts precisely all computations that satisfy φ . The type of finite automata on infinite words we consider is the one defined by Büchi 1962. A *Büchi automaton* is a tuple $A = (\Sigma, S, S_0, \rho, F)$, where Σ is a finite alphabet, S is a finite set of states, $S_0 \subseteq S$ is a set of initial states, $\rho : S \times \Sigma \rightarrow 2^S$ is a nondeterministic transition function, and $F \subseteq S$ is a set of accepting states. A *run* of A over an infinite word $w = a_1 a_2 \dots$, is a sequence $s_0 s_1 \dots$, where $s_0 \in S_0$ and $s_i \in \rho(s_{i-1}, a_i)$ for all $i \geq 1$. A run $s_0, s_1, \dots$ is *accepting* if there is some accepting state that repeats infinitely often, i.e., for some $s \in F$ there are infinitely many i 's such that $s_i = s$. The infinite word w is *accepted* by A if there is an accepting run of A over w . The *language* of infinite words accepted by A is denoted $L(A)$. The following fact establishes the correspondence between LTL and Büchi automata [Vardi and Wolper, 1994] (for a tutorial introduction for this correspondence, see [Vardi, 1996]):

Theorem 1 *Given an LTL formula φ , one can build a Büchi automaton $A_\varphi = (\Sigma, S, S_0, \rho, F)$, where $\Sigma = 2^{Prop}$ and $|S| \leq 2^{O(|\varphi|)}$, such that $L(A_\varphi)$ is exactly the set of computations satisfying the formula φ .*

This correspondence reduces the verification problem to

an automata-theoretic problem as follows [Vardi and Wolper, 1986]. Suppose that we are given a system M and an LTL formula φ . We check whether $L(M) \subseteq \text{models}(\varphi)$ as follows: (1) construct the automaton $A_{\neg\varphi}$ that corresponds to the *negation* of the formula φ , (2) take the *cross product* of the system M and the automaton $A_{\neg\varphi}$ to obtain an automaton $A_{M,\varphi}$, such that $L(A_{M,\varphi}) = L(M) \cap L(A_{\neg\varphi})$, and (3) check whether the language $L(A_{M,\varphi})$ is nonempty, i.e., whether $A_{M,\varphi}$ accepts *some* input.

Theorem 2 *Let M be a labeled transition system and φ be an LTL formula. Then M satisfies φ iff $L(A_{M,\varphi}) = \emptyset$.*

If $L(A_{M,\varphi})$ is empty, then the design is correct. Otherwise, the design is incorrect and the word accepted by $L(A_{M,\varphi})$ is an incorrect computation. Model-checking tools use algorithms for checking emptiness of Büchi automata [Emerson and Lei, 1985; Courcoubetis et al., 1992] to check emptiness of $L(A_{M,\varphi})$. In case of nonemptiness, the incorrect computation is presented to the user as a finite trace, possibly followed by a cycle. Thus, once the automaton $A_{\neg\varphi}$ is constructed, the verification task is reduced to automata-theoretic problems, namely, intersecting automata and testing emptiness of automata, which have highly efficient solutions [Vardi, 1996]. Furthermore, using data structures that enable compact representation of very large state space makes it possible to verify designs of significant complexity [Burch et al., 1992].

The linear-time framework is not limited to using LTL as a specification language. ForSpec is a recent extension of LTL, designed to address the need of the semiconductor industry [Armoni et al., 2002]. There are those who prefer to use automata on infinite words as a specification formalism [Vardi and Wolper, 1994]; in fact, this is the approach of COSPAN [Kurshan, 1994]. In this approach, we are given a design represented as a finite transition system M and a property represented by a Büchi (or a related variant) automaton P . The design is correct if all computations in $L(M)$ are accepted by P , i.e., $L(M) \subseteq L(P)$. This approach is called the *language-containment* approach. To verify M with respect to P , we: (1) construct the automaton P^c that *complements* P , (2) take the product of the system M and the automaton P^c to obtain an automaton $A_{M,P}$, and (3) check that the automaton $A_{M,P}$ is nonempty. As before, the design is correct iff $A_{M,P}$ is empty. Thus, the verification task is again reduced to automata-theoretic problems, namely intersecting and complementing automata and testing emptiness of automata.

3 Concluding Remarks

Over the last few years, automated formal verification tools, such as model checkers, have shown their ability to provide a thorough analysis of reasonably complex designs [Goering, 1997]. Companies such as AT&T, Cadence, Fujitsu, HP, IBM, Intel, Motorola, NEC, SGI, Siemens, and Sun are using model checkers increasingly on their own designs to reduce time to market and ensure product quality, cf. [Beer et al., 1994].

There has recently also been an fruitful interaction between model checking and planning in AI. On one hand, the usage of satisfiability technology to solve planning problems [Kautz and Selman, 1992] has been adopted into model checking

[Biere *et al.*, 1999; Copty *et al.*, 2001]. At the same time, symbolic techniques from model checking have been adopted into AI planning [Cimatti and Roveri, 2000]. Finally, temporal logic and the automata-theoretic perspective has shown to be useful for planning with temporally extended goals [Bacchus and Kabanza, 1998; 2000; Giacomo and Vardi, 1999; Calvanese *et al.*, 2002].

References

- [Armoni *et al.*, 2002] R. Armoni, L. Fix, R. Gerth, B. Ginsburg, T. Kanza, A. Landver, S. Mador-Haim, A. Tiemeyer, E. Singerman, M.Y. Vardi, and Y. Zbar. The ForSpec temporal language: A new temporal property-specification language. In *Proc. 8th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02)*, Lecture Notes in Computer Science 2280, pages 296–311. Springer-Verlag, 2002.
- [Bacchus and Kabanza, 1998] F. Bacchus and F. Kabanza. Planning for temporally extended goals. *Ann. of Mathematics and Artificial Intelligence*, 22:5–27, 1998.
- [Bacchus and Kabanza, 2000] F. Bacchus and F. Kabanza. Using temporal logics to express search control knowledge for planning. *AI J.*, 116(1–2):123–191, 2000.
- [Beer *et al.*, 1994] I. Beer, S. Ben-David, D. Geist, R. Gewirtzman, and M. Yoeli. Methodology and system for practical formal verification of reactive hardware. In *Proc. 6th Conference on Computer Aided Verification*, volume 818 of *Lecture Notes in Computer Science*, pages 182–193, Stanford, June 1994.
- [Biere *et al.*, 1999] A. Biere, A. Cimatti, E.M. Clarke, M. Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs. In *Proc. 36th Design Automation Conference*, pages 317–320. IEEE Computer Society, 1999.
- [Büchi, 1962] J.R. Büchi. On a decision method in restricted second order arithmetic. In *Proc. Internat. Congr. Logic, Method. and Philos. Sci. 1960*, pages 1–12, Stanford, 1962. Stanford University Press.
- [Burch *et al.*, 1992] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, June 1992.
- [Calvanese *et al.*, 2002] D. Calvanese, G. De Giacomo, and M.Y. Vardi. Reasoning about actions and planning in LTL action theories. In *Proc. 8th Int'l Conf. on the Principles of Knowledge Representation and Reasoning*, pages 593–602. Morgan Kaufmann, 2002.
- [Cimatti and Roveri, 2000] A. Cimatti and M. Roveri. Conformant planning via symbolic model checking. *J. of AI Research*, 13:305–338, 2000.
- [Clarke and Wing, 1996] E.M. Clarke and J.M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28:626–643, 1996.
- [Clarke *et al.*, 1986] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, January 1986.
- [Clarke *et al.*, 1999] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
- [Copty *et al.*, 2001] F. Copty, L. Fix, R. Fraer, E. Giunchiglia, G. Kamhi, A. Tacchella, and M.Y. Vardi. Benefits of bounded model checking at an industrial setting. In *Computer Aided Verification, Proc. 13th International Conference*, volume 2102 of *Lecture Notes in Computer Science*, pages 436–453. Springer-Verlag, 2001.
- [Courcoubetis *et al.*, 1992] C. Courcoubetis, M.Y. Vardi, P. Wolper, and M. Yannakakis. Memory efficient algorithms for the verification of temporal properties. *Formal Methods in System Design*, 1:275–288, 1992.
- [Emerson and Lei, 1985] E.A. Emerson and C.-L. Lei. Temporal model checking under generalized fairness constraints. In *Proc. 18th Hawaii International Conference on System Sciences*, North Hollywood, 1985. Western Periodicals Company.
- [Giacomo and Vardi, 1999] G. De Giacomo and M.Y. Vardi. Automata-theoretic approach to planning for temporally extended goals. In *Proc. European Conf. on Planning*, volume 1809 of *Lecture Notes in AI*, pages 226–238. Springer-Verlag, 1999.
- [Goering, 1997] R. Goering. Model checking expands verification's scope. *Electronic Engineering Today*, February 1997.
- [Holzmann, 1997] G.J. Holzmann. The model checker SPIN. *IEEE Trans. on Software Engineering*, 23(5):279–295, May 1997. Special issue on Formal Methods in Software Practice.
- [Kautz and Selman, 1992] H. Kautz and B. Selman. Planning as satisfiability. In *Proc. European Conference on Artificial Intelligence*, pages 359–379, 1992.
- [Kurshan, 1994] R.P. Kurshan. *Computer Aided Verification of Coordinating Processes*. Princeton Univ. Press, 1994.
- [Kurshan, 1997] R.P. Kurshan. Formal verification in a commercial setting. In *Proc. Conf. on Design Automation (DAC'97)*, volume 34, pages 258–262, 1997.
- [Lichtenstein and Pnueli, 1985] O. Lichtenstein and A. Pnueli. Checking that finite state concurrent programs satisfy their linear specification. In *Proc. 12th ACM Symp. on Principles of Programming Languages*, pages 97–107, New Orleans, January 1985.
- [Pnueli, 1977] A. Pnueli. The temporal logic of programs. In *Proc. 18th IEEE Symp. on Foundation of Computer Science*, pages 46–57, 1977.
- [Queille and Sifakis, 1981] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in Cesar. In *Proc. 5th International Symp. on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag, 1981.

- [Vardi and Wolper, 1986] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st Symp. on Logic in Computer Science*, pages 332–344, Cambridge, June 1986.
- [Vardi and Wolper, 1994] M.Y. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1–37, November 1994.
- [Vardi, 1996] M.Y. Vardi. An automata-theoretic approach to linear temporal logic. In F. Moller and G. Birtwistle, editors, *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 238–266. Springer-Verlag, Berlin, 1996.
- [Vardi, 2001] M.Y. Vardi. Branching vs. linear time: Final showdown. In *Proc. Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 2031 of *Lecture Notes in Computer Science*, pages 1–22. Springer-Verlag, 2001.

Automated Reasoning: Past Story and New Trends*

Andrei Voronkov

The University of Manchester

Abstract

We overview the development of first-order automated reasoning systems starting from their early years. Based on the analysis of current and potential applications of such systems, we also try to predict new trends in first-order automated reasoning. Our presentation will be centered around two main motives: efficiency and usefulness for existing and future potential applications.

This paper expresses the views of the author on past, present, and future of theorem proving in first-order logic gained during ten years of working on the development, implementation, and applications of the theorem prover Vampire, see [Riazanov and Voronkov, 2002a]. It reflects our recent experience with applications of Vampire in verification, proof assistants, theorem proving, and semantic Web, as well as the analysis of future potential applications.

1 Theorem Proving in First-Order Logic

The idea of automatic theorem proving has a long history both in mathematics and computer science. For a long time, it was believed by many that hard theorems in mathematics can be proved in a completely automatic way, using the ability of computers to perform fast combinatorial calculations. The very first experiments in automated theorem proving have shown that the purely combinatorial methods of proving first-order theorems are too weak even for proving theorems regarded as relatively easy by mathematicians.

Provability in first-order logic is a very hard combinatorial problem. First-order logic is undecidable, which means that there is no terminating procedure checking provability of formulas. There are decidable classes of first-order formulas but formulas of these classes do not often arise in applications. Due to undecidability, very short formulas may turn out to be extremely complex, while very long ones rather easy. Sometimes first-order provers find proofs consisting of several thousand steps in a few seconds, but sometimes it takes hours to find a ten-step proof. The theory of first-order reasoning is centered around the completeness theorems while in practice completeness is often not an issue due to the intrinsic

complexity of first-order reasoning and is often compromised in favor of a faster inference mechanism.

However, despite the complexity of theorem proving in first-order logic, an impressive progress in the area has been achieved in recent years. This progress is due to several factors: development of theory, implementation techniques, and experience with applications. Modern theorem provers are very powerful programs able to find in a few seconds very complex combinatorial proofs which would require several man-month to be solved by mathematicians. Many ideas developed in theoretical papers in this area are on their way to implementation. There is extensive research on aspects of theorem proving vital for applications, such as proof-checking and interfacing with proof assistants. The next generation of theorem provers will incorporate some of these ideas and become even more powerful tools increasingly used for applications which are currently beyond their reach.

The aim of this paper is to give a brief overview of first-order provers and point out new trends in theorem proving which are likely to be implemented in theorem provers of the next generation.

Currently, theorem provers are used in the following way. The user specifies a problem by giving a set of *axioms* (a set of first order formulas or clauses) and a *conjecture* (again, a first-order formula or a set of clauses). If the input is given by first-order formulas, the prover should check whether the conjecture logically follows from the axioms. If the input is given by a set of clauses, the prover should check whether the set of clauses is inconsistent. In either case, for many applications it is desirable that the prover output a proof, if the logical consequence or inconsistency has been established. The proofs should either be human-readable (for example, when the provers are used for proving theorems in mathematics), or machine-checkable (for example, when provers are used as subsystems of proof assistants or verification systems).

2 Applications

The main application area of theorem provers has been, and continues to be, *verification of software and hardware*. Full applications of this kind usually cannot be directly represented in the first-order form, so provers are normally used to prove subgoals generated by other systems, for example VHDL-to-first-order transformation systems or proof assistants based on higher-order logic or type theories. There are

*Partially supported by a grant from EPSRC.

too many papers on this subject to be mentioned here. Finite-state model checkers and interactive proof assistants are currently prevailing in verification, but with the growing complexity of hardware first-order logic and its extensions are likely to play an increasingly important role.

Theorem proving in mathematics has been the first application area for theorem provers. Provers are not very good at working in structured mathematical theories but they are very efficient in fields of mathematics where combinatorial reasoning is required, for example, in algebra. Applications of provers in algebra are numerous. The monograph [McCune and Padmanabhan, 1996] gives an example of how algebra can be developed with the help of a theorem prover.

Symbolic computation and computer algebra systems need help from theorem provers, for example, when side conditions for applying a simplification rule have to be checked. However, these systems are already very complex and either do not use provers at all or use very primitive ones. Interfacing theorem provers and symbolic computation systems is becoming an important application of theorem provers.

Knowledge bases play a significant role in the semantic web project. Some of these knowledge bases use first-order logic or its simple extensions, see, for example, [IEEE SUO-KIF project, 2002].

Teaching logic and mathematics is an area where first-order provers can serve as valuable tools. The first experiments in teaching mathematics using first-order provers are reported in [McMath *et al.*, 2001].

Other application of first-order provers include retrieval of software components [Schumann and Fischer, 1997], reasoning in non-classical logics (e.g., [de Nivelle *et al.*, 2000]), and program synthesis [Stickel *et al.*, 1994]. First-order theorem provers are in practice used much more than it is reported in the literature, since many companies and organizations do not disclose proprietary use of provers.

3 Modern Theorem Provers

The main currently supported systems are given in Figure 1. Among these systems, Setheo is based on model elimination, while other provers are based on resolution. In the rest of this paper we will overview resolution theorem provers only.

In the early papers, e.g., [Robinson, 1965; Robinson and Wos, 1969] resolution is defined as a logical system consisting of several inference rules operating on clauses, for example, *resolution* and *paramodulation*:

$$\frac{A \vee C \quad \neg A' \vee D}{(C \vee D)\text{mgu}(A, A')} ; \quad \frac{s = t \vee C \quad L[s'] \vee D}{(C \vee L[t] \vee D)\text{mgu}(s, s')} .$$

where mgu denotes the most general unifier. Since these rules are *local*, i.e., their applicability is identified only by a small number of clauses and the result of a rule application does not change the previously generated clauses, resolution is implemented using *saturation algorithms*.

These algorithms operate on a set of clauses S , initially the set of input clauses. Roughly speaking, they are based on the following loop.

1. Apply inferences to clauses in S , adding to S the conclusions of these inferences.

2. If the empty clause $\square$ is proved, terminate with success.
If no inference rule is applicable, terminate with failure.

The set S is the current *search space*. However, a simple implementation of this loop would hardly solve even some of the problems considered trivial by modern theorem provers due to the fast growth of the search space. Already in the first paper on resolution [Robinson, 1965] it was noted that some clauses can be removed from the search space without losing completeness.

More precisely, it has been observed that clauses *subsumed* by other clauses can be removed from the search space without losing completeness and thus regarded as *redundant*. Later, several other notions of redundancy were discovered. For example, [Brand, 1975] proved that the function reflexivity inferences and paramodulation into a variable are redundant. Modern theorem provers use many *redundancy criteria* to prune the search space. These criteria can be divided in two categories: *redundant inferences* and *redundant clauses*. Many useful notions of redundancy are based on *simplification orders* introduced in [Knuth and Bendix, 1970]. These orders are orders $\succ$ on terms which can be extended to literals and clauses. An example of an inference which is redundant due to the orderings restrictions is a paramodulation inference

$$\frac{s = t \vee C \quad L[s'] \vee D}{(C \vee L[t] \vee D)\theta} ,$$

for which $t\theta \succ s\theta$. In the 1970s–1980s many notions of redundant inferences and clauses were investigated. In the 1990s a general theory of redundancies was described in [Bachmair and Ganzinger, 1994a]. Nearly all state-of-the-art resolution theorem provers are based on this theory. Resolution with redundancy is based on the following saturation algorithm.

1. Apply all *non-redundant* inferences to clauses in S , adding to S those conclusions of these inferences *that are non-redundant*.
2. If the empty clause $\square$ is proved, terminate with success.
If no inference rule is applicable, terminate with failure.
3. Remove all clauses that *become redundant* due to the addition of these conclusions of inferences.

In addition to standard inference rules, these algorithms also operate with *simplifications*. An inference is called a *simplification* if it makes at least one clause in S redundant. Many modern provers implement an eager search for simplifying inferences, while ordinary generating inferences are applied lazily.

In modern provers inference selection is performed via *clause selection*. For this reason saturation algorithms implemented in such provers are called the *given clause algorithms*. There are two main concretisations of the saturation algorithm based on the clause selection: the Otter algorithm [Lusk, 1992; McCune, 1994] and the Discount algorithm [Denzinger *et al.*, 1997]. These algorithms are described and analyzed in more detail in [Riazanov and Voronkov, 2002b]. The simple description of these algorithms given above may create an illusion of their simplicity, but in fact, some of these algorithms are extremely complex, and problems related to

prover	affiliation	references
General-purpose provers		
Otter	Argonne National Laboratory	[McCune, 1994]
Setheo	Munich University of Technology	[Moser <i>et al.</i> , 1997]
Spass	Max-Planck Institute	[Weidenbach <i>et al.</i> , 1996]
Vampire	University of Manchester	[Riazanov and Voronkov, 2002a]
Gandalf	Tallinn University of Technology	[Tammet, 1997]
SCOTT	Australian National University	[Slaney <i>et al.</i> , 1994]
E	Munich University of Technology	[Schulz, 2001]
Snark	SRI	[Stickel <i>et al.</i> , 2001]
Bliksem	Max-Planck Institute	[de Nivelle, 2000]

Figure 1: First-order theorem provers

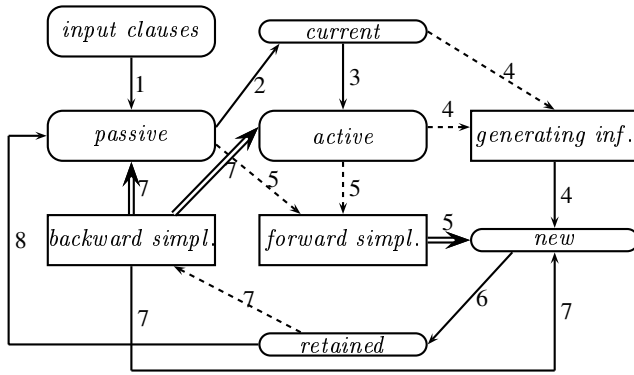


Figure 2: A Given Clause Algorithm of Vampire

the memory management for these algorithms have not been properly solved. For example, one of the given clause algorithms of Vampire is schematically shown in Figure 2 (taken from [Riazanov and Voronkov, 2002a]).

Even proof-search in inference systems with redundancy creates enormously large such spaces. For example, storing 10^6 clauses is not unusual. Of these clauses 10^5 can participate in inferences. Some operations on clauses are very expensive if implemented naively. For example, subsumption on multi-literal clauses is NP-complete and must ideally be performed between each pair of clauses in the search space. It is difficult to imagine an implementation able to perform 10^{12} operations in reasonable time.

In the state-of-the-art theorem provers all expensive operations are implemented using *term indexing* [Graf, 1996; Sekar *et al.*, 2001]. The problem of term indexing can be formulated abstractly as follows. Given a set L of *indexed terms or clauses*, a binary relation R over terms or clauses (called the *retrieval condition*) and a term or clause t (called the *query term or clause*), identify the subset M of L that consists of the terms or clauses l such that $R(l, t)$ holds. A typical retrieval condition used in theorem proving is *subsumption*: retrieve all clauses in L subsumed by t . In order to support rapid retrieval of candidate clauses, we need to process the indexed set into a data structure called the *index*. Modern theorem provers maintain several indexes to support expensive operations. For example, Vampire [Riazanov and Voronkov,

2002a] uses *flatterms* in constant memory for storing temporary clauses, *code trees* [Voronkov, 1995] for forward subsumption, *code trees with precompiled ordering constraints* for forward simplification by unit equalities, *perfectly shared terms* for storing clauses, *shared terms with renaming lists* for backward simplification by unit equalities, *path index with compiled database joins* for backward subsumption and some other indexes.

Serious work with theorem provers requires extensive experiments. Every simple modification to a prover should be tested both for bugs and for efficiency. A typical experiment with Vampire consists of running it for several hours on over 5,000 TPTP problems in a number of modes on a network of around 50 computers. Such experiments require good infrastructure, both software and hardware, to facilitate debugging and evaluation, so Vampire is augmented by a number of programs intended for performing large-scale experiments. The necessity of having such programs and interfacing them with Vampire adds to the complexity of the system.

4 History of Development

Since the early work on automated theorem proving, the area witnessed an impressive progress. This progress is due to several factors described below:

1. Development of *theory*, both of the saturation-based theorem proving with redundancy, see [Bachmair and Ganzinger, 2001; Nieuwenhuis and Rubio, 2001], and of the tableau and model elimination proof-search, see [Hähnle, 2001; Letz and Stenz, 2001; Degtyarev and Voronkov, 2001].
2. Development of *implementation techniques* (term indexing, new algorithms) [Graf, 1996; Sekar *et al.*, 2001; Nieuwenhuis *et al.*, 2001].
3. Growing *experience*, including experiments with applications, the development of TPTP [Sutcliffe and Suttner, 1998], and the annual competitions of theorem provers CASC [Sutcliffe *et al.*, 2002].

Even a naively implemented theorem prover may be powerful enough to solve many problems collected in the TPTP library or created automatically by proof assistants. However, there have been a considerable progress in solving *difficult* problems. Every ten years enhancements in the theory and im-

plementation techniques resulted in a large number of problems which could be solved *several orders of magnitude faster* than by the previous generations of theorem provers. Provers implemented around 1970 were very inefficient, especially for equality reasoning. In 1980 the *paramodulation* rule has been generally adopted and the *general architecture* of saturation-based provers understood. By 1990 provers started to use routinely *simplification orderings* and *term indexing*. The provers used in 2000 employ the *general theory of resolution with redundancy*, and in particular *literal selection functions*. The provers of 2010 are likely to benefit from a better *clause selection*, *built-in theories*, and maybe from parallel or concurrent architectures.

5 Future Generation Theorem Provers

Theorem proving is a very hard problem. The next generation of theorem provers will incorporate new theory, data structures, algorithms, and implementation techniques. Their development will be driven by the quest for *flexibility* and *efficiency*. Flexibility is required to adapt provers to new applications. Efficiency can be reformulated as *controlling redundancy in large search spaces* [Lusk, 1992].

It is unreasonable to expect future theorem provers to be much faster on all possible problems. However, if we can increase performance of provers by *several orders of magnitude* for a *large number of problems coming from applications*, many of these problems will be routinely solved, thus saving time for application developers. The development of next generation provers will require:

1. development of *new theory*;
2. addition of *new features*;
3. development of new *algorithms and data structures*;
4. understanding how the theory developed so far can be efficiently implemented on *top of the existing architectures* of theorem provers;

This development is impossible without considerable implementation efforts and extensive experiments.

There is a large number of research areas to be pursued in automated reasoning in the near future, let me mention some of them.

Built-in theories. Already in the case of equality the naive addition of equality axioms is too inefficient for solving even very simple problems. Development of superposition-based equality reasoning resulted in an impressive improvement of theorem provers. There are important theories other than equality which arise in many applications. Development of specialized reasoning methods for these theories will be a central problem in theorem proving. This problem is different from the problem of designing decision procedures for theories such as Presburger arithmetic because relations and functions used in these theories could be used together with other relations and functions and arbitrary quantifiers. Procedures for the combination of decision and unification algorithms can be of some help [Nelson and Oppen, 1979; Shostak, 1984; Baader and Schulz, 1996; Rueß and Shankar, 2001].

The following built-in theories arise in many applications and are likely to attract attention in automated reasoning.

- AC (the theory of associative and commutative functions). These axioms occur in axiomatizations very often. There are many results related to building-in AC in theorem provers but implementation techniques, including term indexing modulo AC, are still in their infancy. Associativity and commutativity were built in the EQP theorem prover [McCune, 1997] with a considerable success but essentially without term indexing. Term indexing modulo AC was considered in [Bachmair *et al.*, 1993] but only for a very special case.
- Theories of transitive relations and orders [Bachmair and Ganzinger, 1994b].
- Various first-order theories of arithmetic.
- Term algebras and other constructor-based structures.

As a first step toward building-in important theories one can consider creation of libraries of axioms/theorems about commonly used data types. An example of such a project is the Standard Upper Ontology [IEEE SUO-KIF project, 2002]. One can also enhance theorem provers by constructs for specifying built-in theories, for example, by constraints or by additional inference rules.

Inductively defined types. In many applications of interactive proof assistants one has to deal with inductively defined types and functions on these types. The proof assistants such as Isabelle [Paulson, 2002], HOL [Gordon and Melham, 1993], COQ [The Coq Development Team, 2001], Twelf [Pfenning and Schürmann, 1998] have facilities to define data types and functions inductively. First-order theorem provers have no such facilities. The work on building-in inductively defined types can be developed in the following directions:

- First-order reasoning on data types given by inductive definitions.
- First-order reasoning on functions defined over such data types.
- Limited forms of inductive reasoning.

Working with large axiomatizations. Very often theorem provers must work with large axiomatizations containing many irrelevant axioms. This is typical for applications such as reasoning with ontologies, assisting proof assistants, and verification using hierarchically defined theories. Recognition of irrelevant axioms is one of the most important problems in theorem proving. Many of these axioms are definitions of relations and functions in various forms. Recognition of the most typical definitions and efficient work with them play a major role in the future provers. The first steps in this direction are reported in [Ganzinger *et al.*, 2001; Afshordel *et al.*, 2001; Degtyarev *et al.*, 2002].

Ontology-based knowledge representation. First-order formulations of many applied problems, if not created manually, often refer to large axiomatizations. This situation is common for knowledge-base reasoning, proof assistants, and

theorem proving in mathematics. Using these large axiomatizations and trying to extract only a small number of relevant axioms can be a hopeless task. The problem arises of structuring these large axiomatizations in a such a way that a relevant subset of them could be easily extracted automatically and that particular properties of the axiomatization can be exploited by provers. For example, provers could use a fact that a subset of axioms axiomatize properties of a relation in terms of other relations. Development of knowledge representation formalisms suitable for representing large applied theories and adaptation of provers to such formalisms could become an important problem in automated reasoning.

Proof checking. Strangely enough, modern theorem provers are not always good at producing proofs. Some of them print detailed resolution and paramodulation proofs but none gives a proof of the preprocessing steps, such as skolemisation. Without proof-checking one cannot use provers in verification and in assisting proof assistants. We expect that in the near future proof-checking components will be added to all major first-order provers. We conjecture that the main approaches to proof-checking will be similar to those used in proof-carrying code [Necula and Lee, 2000], so that a proof is built using a number of inference rules, and foundational proof-carrying code [Appel, 2001], so that a proof will given in a system whose language is rich enough to prove even the preprocessing steps, such as HOL.

Reasoning with **nonstandard quantifiers**, such as those specifying restrictions on the number of elements satisfying a relation, for example, $\exists^{\leq n}$ (there exists at most n). Such quantifiers are familiar to the description logic community. One can translate formulas with such quantifiers into first-order logic with equality but the translation will create formulas of a prohibitive size. It is interesting to develop special ways of reasoning with such quantifiers, although this is not an obvious task.

Distributed and other non-standard architectures. It has been observed that cooperating heterogeneous theorem provers can perform much better than the mere sum of their components [M.P. Bonacina, 1994; Denzinger and Fuchs, 1999]. However, modern theorem prover architectures are not suited for cooperative or distributed theorem proving.

Non-resolution inference system. The recent rapid progress of the DPLL-based satisfiability checkers [Zhang and Malik, 2002] suggests that non-resolution systems could also play a significant role in first-order theorem proving. The first implementation of first-order DPLL [Baumgartner, 2000] was not very encouraging, but so were the first implementations of resolution. To be more generally applicable, non-resolution system must find an efficient solution to the problem of built-in theories [Degtyarev and Voronkov, 2001].

Satisfiability-checking and model building. For many applications it is desirable to be able to establish satisfiability of sets of clauses and build models for satisfiable sets. Satisfiability of first-order is conceptually a much harder problem than unsatisfiability. The set of satisfiable formulas is not recursively enumerable, which means that there is no semi-decision procedure for satisfiability-checking.

Other features. There are other features required of the next generation first-order theorem provers. For example, for

the naive users who do not know (and usually do not want to know) much about theorem proving, theorem provers must have a strong auto-mode which will try to select automatically a strategy or strategies best suited for solving the given problem. For more advanced users, there should be options to specify term orderings, literal selection, and clause selection.

References

- [Afshordel *et al.*, 2001] B. Afshordel, T. Hillenbrand, and C. Weidenbach. First-order atom definitions extended. In R. Nieuwenhuis and A. Voronkov, editors, *LPAR 2001*, volume 2050 of *LNAI*, pages 309–319. Springer Verlag, 2001.
- [Appel, 2001] A.W. Appel. Foundational proof-carrying code. In *LICS 2001*, pages 247–258, 2001.
- [Baader and Schulz, 1996] F. Baader and K.U. Schulz. Unification in the union of disjoint equational theories: Combining decision procedures. *Journal of Symbolic Computations*, 21:211–243, 1996.
- [Bachmair and Ganzinger, 1994a] L. Bachmair and H. Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *Journal of Logic and Computation*, 4(3):217–247, 1994.
- [Bachmair and Ganzinger, 1994b] L. Bachmair and H. Ganzinger. Rewrite techniques for transitive relations. In *LICS'94*. IEEE Computer Society Press, 1994.
- [Bachmair and Ganzinger, 2001] L. Bachmair and H. Ganzinger. Resolution theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 2, pages 19–99. Elsevier Science, 2001.
- [Bachmair *et al.*, 1993] L. Bachmair, T. Chen, and I.V. Ramakrishnan. Associative-commutative discrimination nets. In M.-C. Gaudel and J.-P. Jouannaud, editors, *TAPSOFT'93*, volume 668 of *LNCS*, pages 61–74. Springer Verlag, 1993.
- [Baumgartner, 2000] P. Baumgartner. FDPLL — a first-order Davis-Putnam-Logemann-Loveland procedure. In D. McAllester, editor, *CADE-17*, volume 1831 of *LNAI*, pages 200–219. Springer Verlag, 2000.
- [Brand, 1975] D. Brand. Proving theorems with the modification method. *SIAM Journal of Computing*, 4:412–430, 1975.
- [de Nivelle *et al.*, 2000] H. de Nivelle, R.A. Schmidt, and U. Hustadt. Resolution-based methods for modal logics. *Logic Journal of the IGPL*, 8(3):265–292, 2000.
- [de Nivelle, 2000] H. de Nivelle. *Bliksem 1.10 User's Manual*. MPI für Informatik, Saarbrücken, 2000.
- [Degtyarev and Voronkov, 2001] A. Degtyarev and A. Voronkov. Equality reasoning in sequent-based calculi. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 10, pages 611–706. Elsevier Science, 2001.
- [Degtyarev *et al.*, 2002] A. Degtyarev, R. Nieuwenhuis, and A. Voronkov. Stratified resolution. *Journal of Symbolic Computations*, pages 1–24, 2002. to appear.
- [Denzinger and Fuchs, 1999] J. Denzinger and D. Fuchs. Cooperation of heterogeneous provers. In T. Dean, editor, *Proc. (IJCAI-99)*, volume 1, pages 10–15, 1999.
- [Denzinger *et al.*, 1997] J. Denzinger, M. Kronenburg, and S. Schulz. DISCOUNT — a distributed and learning equational prover. *Journal of Automated Reasoning*, 18(2):189–198, 1997.
- [Ganzinger *et al.*, 2001] H. Ganzinger, R. Nieuwenhuis, and P. Nivela. *The Saturate system*. Max-Planck Institute für Informatik, 2001.
- [Gordon and Melham, 1993] M. Gordon and T. Melham. *Introduction to HOL*. Cambridge University Press, 1993.

- [Graf, 1996] P. Graf. *Term Indexing*, volume 1053 of *LNCS*. Springer Verlag, 1996.
- [Hähnle, 2001] R. Hähnle. Tableaux and related methods. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 3, pages 100–178. Elsevier Science, 2001.
- [IEEE SUO-KIF project, 2002] IEEE SUO-KIF project. Standard upper ontology knowledge interchange format, 2002.
- [Knuth and Bendix, 1970] D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970.
- [Letz and Stenz, 2001] R. Letz and G. Stenz. Model elimination and connection tableau procedures. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 28, pages 2015–2114. Elsevier Science, 2001.
- [Lusk, 1992] E.L. Lusk. Controlling redundancy in large search spaces: Argonne-style theorem proving through the years. In A. Voronkov, editor, *LPAR '92.*, volume 624 of *LNAI*, pages 96–106, 1992.
- [McCune and Padmanabhan, 1996] W. McCune and R. Padmanabhan. *Automated Deduction in Equational Logic and Cubic Curves*, volume 1095 of *LNAI*. Springer Verlag, 1996.
- [McCune, 1994] W.W. McCune. OTTER 3.0 reference manual and guide. Technical Report ANL-94/6, Argonne National Laboratory, 1994.
- [McCune, 1997] W. McCune. Solution of the robbins problem. *Journal of Automated Reasoning*, 19(3):263–276, 1997.
- [McMath et al., 2001] D. McMath, M. Rozenfeld, and R. Sommer. A computer environment for writing ordinary mathematical proofs. In R. Nieuwenhuis and A. Voronkov, editors, *LPAR 2001*, volume 2050 of *LNAI*, pages 502–511. Springer Verlag, 2001.
- [Moser et al., 1997] M. Moser, O. Ibens, R. Letz, J. Steinbach, C. Goller, J. Schumann, and K. Mayr. SETHEO and E-SETHO—the CADE-13 systems. *Journal of Automated Reasoning*, 18:237–246, 1997.
- [M.P. Bonacina, 1994] W.W. McCune M.P. Bonacina. Distributed theorem proving by peers. In A. Bundy, editor, *CADE-12*, volume 814 of *LNAI*, pages 841–845, 1994.
- [Necula and Lee, 2000] G.C. Necula and P. Lee. Proof generation in the Touchstone theorem prover. In D. McAllester, editor, *CADE-17*, volume 1831 of *LNAI*, pages 25–44. Springer Verlag, 2000.
- [Nelson and Oppen, 1979] G. Nelson and D.C. Oppen. Simplification by cooperating decision procedures. *ACM Transactions on Programming Languages and Systems*, 1(2):245–257, 1979.
- [Nieuwenhuis and Rubio, 2001] R. Nieuwenhuis and A. Rubio. Paramodulation-based theorem proving. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume I, chapter 7, pages 371–443. Elsevier Science, 2001.
- [Nieuwenhuis et al., 2001] R. Nieuwenhuis, T. Hillenbrand, A. Riazanov, and A. Voronkov. On the evaluation of indexing techniques for theorem proving. In R. Goré, A. Leitsch, and T. Nipkow, editors, *IJCAR 2001*, volume 2083 of *LNAI*, pages 257–271. Springer Verlag, 2001.
- [Paulson, 2002] L.C. Paulson. *The Isabelle Reference Manual*. Cambridge University, 2002.
- [Pfenning and Schuermann, 1998] F. Pfenning and C. Schuermann. *Twelf User's Guide*, 1998.
- [Riazanov and Voronkov, 2002a] A. Riazanov and A. Voronkov. The design and implementation of Vampire. *AI Communications*, 15(2-3):91–110, 2002.
- [Riazanov and Voronkov, 2002b] A. Riazanov and A. Voronkov. Limited resource strategy in resolution theorem proving. *Journal of Symbolic Computations*, pages 1–14, 2002. to appear.
- [Robinson and Wos, 1969] G. Robinson and L.T. Wos. Paramodulation and theorem-proving in first order theories with equality. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 135–150. Edinburgh University Press, 1969.
- [Robinson, 1965] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery*, 12(1):23–41, 1965. Reprinted as [Robinson, 1983].
- [Robinson, 1983] J.A. Robinson. A machine oriented logic based on the resolution principle. In J. Siekmann and G. Wrightson, editors, *Automation of Reasoning. Classical Papers on Computational Logic*, volume 1, pages 397–415. Springer Verlag, 1983. Reprinted from [Robinson, 1965].
- [Rueß and Shankar, 2001] H. Rueß and N. Shankar. Deconstructing Shostak. In *LICS 2001*, pages 19–28, 2001.
- [Schulz, 2001] S. Schulz. System abstract: E 0.61. In R. Goré, A. Leitsch, and T. Nipkow, editors, *IJCAR 2001*, volume 2083 of *LNAI*, pages 370–375. Springer Verlag, 2001.
- [Schumann and Fischer, 1997] J. Schumann and B. Fischer. NORA/HAMMR: Making deduction-based software component retrieval practical. In *Proc. Automated Software Engineering (ASE-97)*, pages 246–254. IEEE Computer Society Press, 1997.
- [Sekar et al., 2001] R. Sekar, I.V. Ramakrishnan, and A. Voronkov. Term indexing. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 26, pages 1853–1964. Elsevier Science, 2001.
- [Shostak, 1984] R.E. Shostak. Deciding combinations of theories. *Journal of the Association for Computing Machinery*, 31(1):1–12, 1984.
- [Slaney et al., 1994] J.K. Slaney, E.L. Lusk, and W. McCune. SCOTT: Semantically Constrained Otter (system description). In A. Bundy, editor, *CADE-12*, volume 814 of *LNAI*, pages 764–768, 1994.
- [Stickel et al., 1994] M. Stickel, R. Waldinger, R. Lowry, T. Pressburger, and I. Underwood. Deductive composition of astronomical software from subroutine libraries. In A. Bundy, editor, *CADE-12*, volume 814 of *LNAI*, pages 341–355, 1994.
- [Stickel et al., 2001] M.E. Stickel, R.J. Waldinger, and V.K. Chaudhri. *A Guide to Snark*. SRI, 2001.
- [Sutcliffe and Suttner, 1998] G. Sutcliffe and C. Suttner. The TPTP problem library — CNF release v. 1.2.1. *Journal of Automated Reasoning*, 21(2), 1998.
- [Sutcliffe et al., 2002] G. Sutcliffe, C.B. Suttner, and F. J. Pelletier. The IJCAR ATP system competition. *Journal of Automated Reasoning*, 28(3):307–320, 2002.
- [Tammet, 1997] T. Tammet. Gandalf. *Journal of Automated Reasoning*, 18(2):199–204, 1997.
- [The Coq Development Team, 2001] The Coq Development Team. *The Coq Proof Assistant Reference Manual. Version 7.2*. INRIA, 2001.
- [Voronkov, 1995] A. Voronkov. The anatomy of Vampire: Implementing bottom-up procedures with code trees. *Journal of Automated Reasoning*, 15(2):237–265, 1995.
- [Weidenbach et al., 1996] C. Weidenbach, B. Gaede, and G. Rock. SPASS & FLOTTER. Version 0.42. In M.A. McRobbie and J.K. Slaney, editors, *CADE-13*, volume 1104 of *LNCS*, pages 141–145, 1996.
- [Zhang and Malik, 2002] L. Zhang and S. Malik. The quest for efficient boolean satisfiability solvers. In A. Voronkov, editor, *CADE-18*, volume 2392 of *LNAI*, pages 295–313, 2002.

Automatically Personalizing User Interfaces

Daniel S. Weld Corin Anderson* Pedro Domingos Oren Etzioni

Krzysztof Gajos Tessa Lau† Steve Wolfman

Department of Computer Science & Engineering

University of Washington, Box 352350

Seattle, WA 98195–2350 USA

Abstract

Today's computer interfaces are one-size-fits-all. Users with little programming experience have very limited opportunities to customize an interface to their task and work habits. Furthermore, the overhead induced by generic interfaces will be proportionately greater on small form-factor PDAs, embedded applications and wearable devices. Automatic personalization may greatly enhance user productivity, but it requires advances in *customization* (explicit, user-initiated change) and *adaptation* (interface-initiated change in response to routine user behavior). In order to improve customization, we must make it easier for users to direct these changes. In order to improve adaptation, we must better predict user behavior and navigate the inherent tension between the dynamism of automatic adaptation and the stability required in order for the user to predict the computer's behavior and maintain control. This paper surveys a decade's work on customization and adaptation at the University of Washington, distilling the lessons we have learned.

1 Introduction

Today's software is mass produced with a plethora of features designed to satisfy every user. But since different people are working on different tasks with different styles, there is no way to organize features in a way that makes essential functionality convenient for everyone. Consider word processing, for example; most people will never use automatic line numbering, but legal secretaries couldn't survive without it.

The shift away from the desktop and towards pervasive computing greatly exacerbates the problem of "Steelcase Inspired" software for several reasons. First, the shift from quiet office to ubiquitous use introduces complex environmental factors that increase the differences between the usage patterns of distinct individuals; indeed, a single user will require different features depending on context. Second, the form

factor of mobile devices is enormously more variable than that of desktop machines — for example, the ratio of large to small desktop display size is about four, but it varies by a factor of 625 between a cell-phone and a liveboard.

The wisdom of user-centered design have been well documented [Lewis and Reiman, 1993]. However, as user needs and device characteristics diverge, the traditional design methodology of intensive user studies becomes unscalable. While each user deserves a personalized interface, designed for the device at hand and providing simple access to the commands and features they need, there aren't enough ethnomethodologists and designers to manage, interpret, and respond to so many studies. Indeed, the only resources that scale with the number of users are the users themselves and the computing devices they are using.

We use the term *adaptation* to denote personalization which is automatically performed by the interface without explicit user directives. By *customization* we mean personalization which is directly requested by the user. Our argument, thus, is that adaptivity and customization are the *only* scalable approaches to personalization. Interfaces should automatically adapt to the capabilities of the device at hand, to network connectivity, and to the individual user's activities, location, and context; users should be able to guide and control adaptation through a variety of customization mechanisms.

1.1 Customization and Adaptation

Of course, many existing desktop applications allow limited customization by letting the user (1) select which menus are visible, (2) add buttons to toolbars, (3) define macros, and (4) even add custom functionality via scripting languages such as Visual Basic. While customization is important, most users fail to customize effectively. Few users are comfortable with macros, regular expressions, or scripting languages, and even programmers are often too busy to invest the necessary time *now* in order to speed future sessions. Thus, the challenge is to create improved methods for users to direct their interface, rearrange functionality as well as appearance, and recover from inappropriate adaptations.

Similarly, many systems support limited adaptation, but users don't always appreciate it. Defaults that remember the last option or directory selected are simple and can be helpful, but Microsoft's smart menus disorient, while the Office

*Current address: Google Inc, 2400 Bayshore Parkway, Mountain View, CA, 94043

†Current address: IBM TJ Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598

Assistant's guesses fall far from the mark. The danger of adaptivity is its potential to create confusing inconsistency and rob the user of control. Thus, the challenge is to develop datamining algorithms that can accurately predict a user's behavior and to navigate the tension between too rapid adaptation (disorienting) and too little (inefficient), and to seek interface metaphors that increase user control.

1.2 Deep Deployment

Successful customization and adaptation methods will maximize their benefit when used by a broad range of applications. Ideally, there will be *deep deployment*: a uniform layer at the operating systems level that records clickstream data across applications, and supports ubiquitous personalization. For this to work, the interfaces themselves must be described using declarative representations like those developed by the model-based interface community [Foley *et al.*, 1989; Puerta, 1996]

Deploying personalization methods at the OS layer will bring several advantages. The first is consistency of behavior amongst applications. Second, knowledge of a user's activity with one application may improve adaptation of another's interface. Third, cross-application personalization may offer the highest benefit to users, since many common tasks involve patterns connecting two or more programs.

1.3 Outline

In the rest of this paper, we describe a sequence of our projects which are making progress on this vision. The underlying techniques range along a continuum of user-involvement, and our expository progression follows this vector from customization towards pure adaptation. Along the way we compare related endeavors and summarize lessons learned.

2 Customization by Command

The Internet Softbots project [Etzioni and Weld, 1994] marked the genesis of work on Intelligent User Interfaces at the University of Washington. By acting as a personal assistant, the softbot supported a qualitatively different kind of human-computer interface. Users were able to make high-level requests, and the softbot used search [Weld, 1996], inference [Etzioni *et al.*, 1997], and knowledge to determine how to satisfy the request. Furthermore, the softbot was able to tolerate and recover from ambiguity, omissions, and errors in these requests.

At its core, the softbot could handle goals specified in an expressive subset of first-order logic with modal operators for handling time and for distinguishing information gathering goals from those that requested state changes [Golden and Weld, 1996]. We labeled the softbot a *goal-oriented* interface, because human requests specified *what* the user wanted; the softbot was responsible for deciding *how* and *when* to satisfy the request. Since most users are uncomfortable with logical notation, we provided a forms-oriented interface front end. Unfortunately, we found the forms approach to be unscalable, and goal-specification was a challenge for many users. In response, we investigated two research directions: reliable natural-language interfaces [Popescu *et al.*, 2003;

Yates *et al.*, 2003] and programming by demonstration [Lau *et al.*, 2000].

3 Programming by Demonstration

If it's too hard for users to specify goals, then a natural objective is the design of an interface that can watch a user's normal behavior and help as appropriate. In the case of repetitive tasks, this amounts to programming by demonstration (PBD). Of course, PBD has been studied extensively [Cypher, 1993], but most previous systems were heuristic and domain-specific. We sought a domain-independent approach, suitable for deep deployment, that offered the expressiveness of a scripting language and the ease of macro recording, without a verbatim recorder's accompanying brittleness. While our PBD interface resembles a keystroke-based macro interface, it generalizes from the demonstrated actions to a robust program which is more likely to work in different situations.

It is useful to think of a PBD-interface as having three components: 1) *segmentation* determines when the user is executing an automatable task, 2) *trace induction* predicts what the user is doing from a prefix of her activity trace, and 3) *facilitation* manages user interaction to aid the user in completing her task. The focus of our work was on the trace induction phase.

We formalized PBD trace induction as a learning problem, as follows. A repetitive task may be solved by a program with a loop, where each iteration solves one instance of the task. The PBD system must infer the correct program from a demonstration of the first few iterations. Each action (*e.g.*, move, select, copy, paste, ...) the user performs during this demonstration causes a change in the state of the application (*e.g.*, defines a mapping between editor states). Therefore, we modeled this problem as one of inferring the function that maps one state to the next, based on observations of the state prior to and following each user action.

PBD presents a particularly challenging machine learning problem, because users are extremely reluctant to provide more than a few training instances. Thus the learner must be able to generalize from a very small number of iterations. Yet in order to be useful, a wide range of programs must be learnable. Thus the problem combines a weak bias with the demand for low sample complexity. Our solution, called *version-space algebra*, lets the application designer combine multiple strong biases to achieve a weaker one that is tailored to the application, thus reducing the statistical bias for the least increase in variance. In addition, the learning system must be able to interact gracefully with the user: presenting comprehensible hypotheses, and taking user feedback into account. Version-space algebra addresses this issue as well.

3.1 Version-Space Algebra

Originally developed for concept learning, a *version space* is the subset of a hypothesis space which is consistent with a set of training instances [Mitchell, 1982]. If there is a partial order over candidate hypotheses, one may represent the version space implicitly (*i.e.*, with boundary sets) and manage updates efficiently. Version space algebra defines transformation operators (*e.g.*, union, join, *etc.*) for combining simple

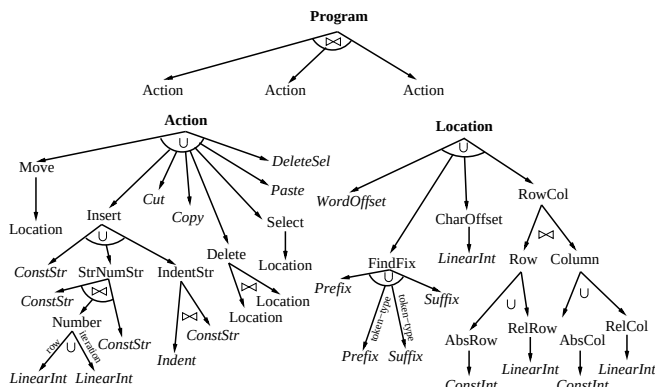


Figure 1: Algebraic specification of the version space for a SMARTEDIT program.

version spaces into more complex ones. We also developed a probabilistic framework for reasoning about the likelihood of each hypothesis in a composite version space.

After constructing a library of reusable, domain-independent component version spaces, we combined a set of primitive spaces to form a bias for learning text-editing programs (Figure 1), which was used in the SMARTEDIT PBD implementation.

3.2 The SmartEdit Implementation

We verified the utility of version-space algebra for PBD with SMARTEDIT, an Emacs-like editor. When a user notices that she is about to perform a repetitive task, she clicks a button to start the PBD recorder. After completing one instance of the task, she clicks another button to mark the first demonstration. SMARTEDIT initializes the version space using the recorded state sequence as the first training example. As the user continues to type, SMARTEDIT updates the version space with every action. In parallel, SMARTEDIT displays what action it thinks the user will next perform, and its confidence in the prediction. These probabilities are calculated by voting the version space. When the user is confident that SMARTEDIT has learned her procedure, she may let it execute in single-step or fully autonomous mode. We explored a number of ways of structuring the user’s dialog with the learner, including decision-theoretic control [Wolfman *et al.*, 2001], but many aspects of facilitation require further study.

Version-space algebra provides SMARTEDIT with several benefits: efficient methods for incremental update and voting, without explicit enumeration of the version space [Lau *et al.*, 2000], and flexibility. By varying the algebraic formulation for the hypothesis space, we experimented with several other version spaces for learning single-loop programs from traces with varying amounts of segmentation information, as well as learning programs with multiple nested loops.

3.3 Experiments and Observations

Empirical studies validated our approach on a range of text-editing scenarios (*e.g.*, bibliographic reformatting, HTML to $\text{\LaTeX}$ conversion, information extraction, *etc.*). We show that a program that generalizes correctly for each of

these scenarios can be learned quickly in as few as one or two training examples. An informal user study confirmed SMARTEDIT’s usability and usefulness, and showed that even novice SMARTEDIT users perform tasks more quickly and concisely with PBD than without.

However, we learned a negative lesson when we started widespread distribution — despite incorporation within Emacs, relatively few people used SMARTEDIT frequently. From this we conclude that the overhead of a macro recorder interface is relatively high. Since most repetitive tasks are short, it is often easier to do them manually and avoid the cognitive load of complex customization. Others have found similar results — users customize relatively little [Mackay, 1991] — perhaps because customization facilities are complex themselves [McGrenere *et al.*, 2002]. We conjecture that widespread PBD adoption requires automatic segmentation, and the adaptation mechanisms, described in the next section, may provide this capability.

4 Adapting to User Behavior

The AI community has a long-standing interest in adaptive interfaces. The Calendar’s Apprentice [Dent *et al.*, 1992] used machine learning to predict meeting location and durations. This research and similar work on email classification [Maes and Kozierok, 1993] led to an important principle about the incorporation of imperfect behavioral predictions in an interface: defaults are an effective way to minimize the cost to the user of (inevitable) poor predictions. Horvitz’s decision-theoretic framework [Horvitz, 1999] resulted in additional principles: graceful degradation of service prediction, expected utility of disambiguation dialogs as a function of user time and attention), and the use of timeouts to minimize cost of prediction errors.¹

Our work on adaptive interfaces focussed initially on website design [Perkowitz and Etzioni, 1997] and led to algorithms for datamining web logs to discover aggregate patterns, which powered the automatic creation of index pages [Perkowitz and Etzioni, 2000]. Later we concentrated on mining individual behavior patterns, generating personalized sites for display on small, wireless devices [Anderson *et al.*, 2001b]. Our emphasis was on “information-goal seeking” behavior, common to wireless internet use, rather than general browsing or surfing. The PROTEUS system modeled adaptation as search through the space of possible websites. Site modification operators included highlighting text, adding shortcut links, and eliding parts of pages. We adopted the decision-theoretic approach, guiding search with expected utility calculations based on a model of the cost of manipulating the small PDA screen and fetching pages over the slow wireless connection.

A small user study revealed two important lessons. For the most part, the adaptations were good ones; PROTEUS suggested useful shortcuts and the elided content was almost always useless to the user. However, the cost of deleting important parts of a page was very high. Reweighting our utility function would have led to more conservative behavior, but

¹It is ironic that the Office Assistant violates most of Horvitz’s principles, which were developed at Microsoft Research.

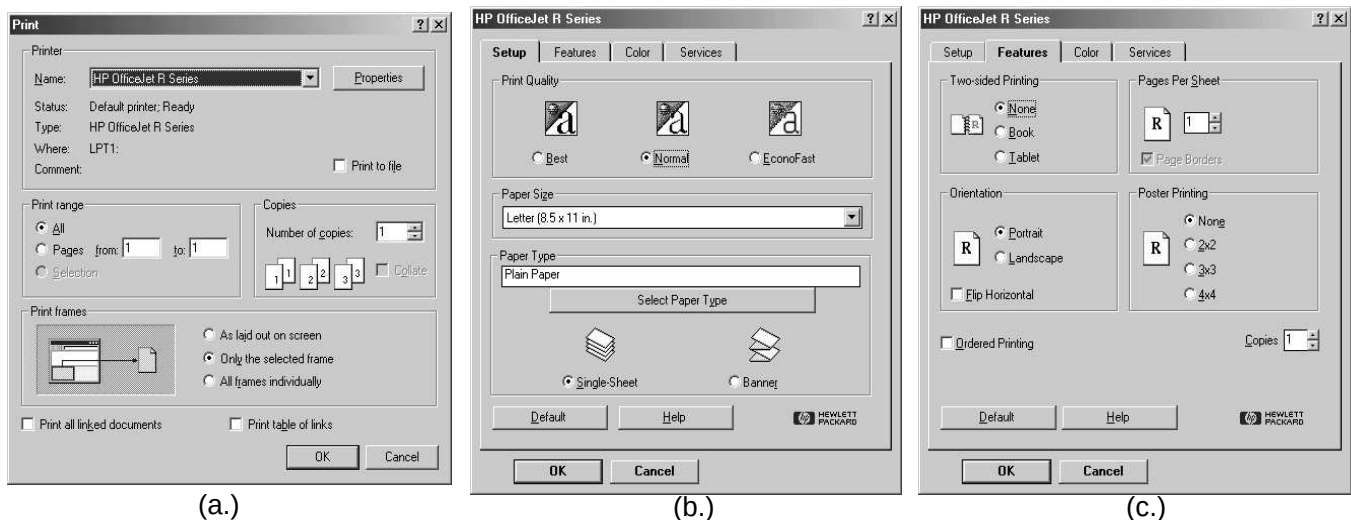


Figure 2: (a) Main dialog box for print command. (b) After clicking the “Properties...” button, the default “Setup” tab is displayed. (c) After clicking the “Features” tab, the user can select double-sided printing; the user must now click “ok” twice.

the problem of accurate behavior prediction is central to interface adaptation; we discuss it in the next section.

A second problem stemmed from the fact that frequently users didn’t find a shortcut link, even though PROTEUS had added it in the appropriate place. Shortcut-naming was usually to blame, and this raised the general issue of saliency, which we discuss in section 4.2.

4.1 Predicting User Behavior

High-quality adaptation requires the ability to learn an extremely accurate model of user behavior. Our comparison of existing techniques showed that a mixture of Markov models had the best predictive power [Anderson *et al.*, 2001a]. (Shortcuts generated by the mixture model saved users up to 40% of their navigational effort.) But all models performed poorly when data was sparse; hence none of these models would work well for situations where a user is exploring new parts of the interface — even if she is performing a task which is similar to one she has performed frequently in another context (*e.g.*, shopping cart checkout at a new store). To remedy this problem, we developed *relational Markov models* (RMMs), a generalization of Markov models that overlays relational structure on the states [Anderson *et al.*, 2002].

Intuitively, RMMs do for Markov models what Probabilistic Relational Models (PRMs) do for Bayesian networks [Friedman *et al.*, 1999]. RMMs generalize Markov models by partitioning the states into classes; each class has an associated k -ary relation, R , and members of the class correspond to R instantiated with distinct ground values. The domain of each variable can be hierarchically structured, and a smoothing technique called *shrinkage* is carried out over the cross product of these hierarchies. RMMs make it possible to generalize beyond the observed states. For example, suppose a user surfing through an e-commerce Web site goes from a page about Nirvana’s “Nevermind” CD to a page containing a biography of Nirvana. An ordinary Markov model

can infer nothing from this about what the user will do when visiting a page about a Pearl Jam CD, but an RMM would be able to infer that the user might next go to a Pearl Jam biography. RMMs make effective learning possible in domains with very large and heterogeneous state spaces, given only sparse data. Experiments on academic and e-commerce websites show that RMMs trained on only ten instances perform as well as Markov models which were trained on ten thousand instances [Anderson *et al.*, 2002]. We are currently extending RMMs by incorporating structure in the same way that dynamic Bayes nets extend Markov models; see [Sanghai *et al.*, 2003].

4.2 Partitioned Dynamicity

As our experience with unnoticed shortcuts showed, saliency is essential — any adaptive mechanism that introduces new commands or options to an interface, must ensure that the user finds them. Furthermore, the mechanism must take care not to obscure important existing functionality.

One way to increase saliency, while minimizing the cognitive dissonance associated with adaptivity, is to *partition dynamicism* — to segregate dynamic and static areas of an interface. Indeed, a number of well-designed adaptive interfaces exhibit this property. For example, the news stories and advertisements on Yahoo change continuously, but because the layout is fixed, the adaptation causes little distress to users. In contrast, the automatic menu shortening feature of Microsoft Office 2000 violates partitioned dynamicity, since the location of commands in the menus changes unpredictably.

Powerpoint XP’s “Insert Symbol” command provides another example. Previous versions required a long sequence of clicks to select the correct font and navigate to the desired symbol, but the dialog box in the XP version has a “Recently used symbols” area.

We conjecture that partitioned dynamicity is a useful general principle for adaptation in interfaces. Because

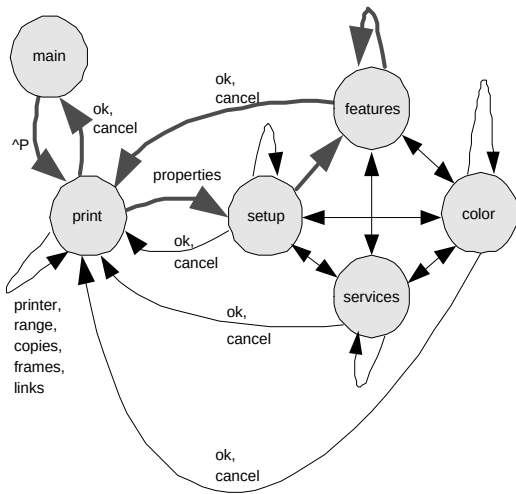


Figure 3: Abstract-state machine for the printing dialogs; the example user's path is darkened.

users become accustomed to changes in the dynamic area, these changes don't disturb their conceptual model [Norman, 1998]. And because existing navigational patterns remain unchanged, users maintain control even if adaptation is unhelpful. The alternative, dual-interface approach of [McGrenere *et al.*, 2002] shares many benefits with partitioned dynamicity, but trades convenience for screen real estate.

4.3 Example: Adapting Desktop Dialogs

We believe that the PROTEUS architecture, developed for adaptive websites, would be even more useful in a desktop application setting. As a story-board example, suppose that a user wishes to print a Word document, doublesided. As Figures 2 and 3 show this requires six user actions.² First, the user presses `ctrl-P` to print the current document. Next, she must click on "Properties..." which yields "Setup" options. Clicking on the "Features" tab uncovers two sided printing options; another click selects book-style duplex. Then "Ok" must be clicked twice, once to close the Properties window and the other to confirm printing.³ However, if an individual user executes this pattern hundreds of times, always choosing duplex printing, the system should recognize this and provide a shortcut.

Figure 4 shows one possible result, where the main screen has partitioned dynamicity, and the duplex options (with a new default value) have been added. In this example, we assume that the user had never chosen the "tablet" duplex value. For the shortcut, therefore, the system deleted the value, casting the duplex variable into a Boolean. As a result, the duplex option could be rendered with a simple, space-efficient checkbox. Of course, if the user later desired the tablet format, they

²Using Windows ME, Office XP, and a HP R80 printer.

³Of course, one can quibble with the design of this particular interaction, but it is a result of today's software engineering process, and many similar designs exist—that's why adaptation is so promising

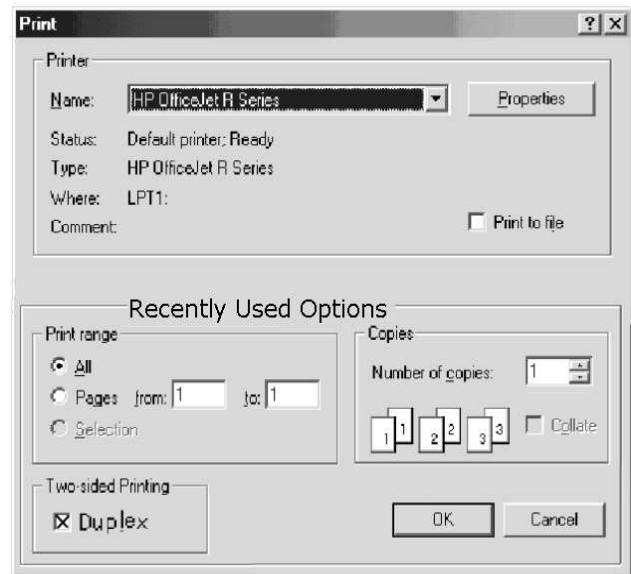


Figure 4: After adaptation, duplex printing is the default and visible on the main print dialog. By discarding the infrequently used "tablet" value, the variable can be cast as a Boolean and a simple, space-efficient check-box widget used.

could find it unchanged on the "Features" tab.⁴

Restricting a state variable's set of valid values and type casting are just two representative transformations; many more exist. Note that it is only possible to perform these transformations when the interface is represented in an abstract, declarative language. The next section discusses additional benefits from such a representation.

5 Adapting to Device Characteristics

The trend towards mobile and ubiquitous computing has resulted in an morass of device form factors and input mechanisms, and it is nearly impossible for a designer to target each of these manually. Furthermore, as shown in previous section, some parts of the interface may be generated dynamically in response to the particular usage patterns. Hence, we are building an automated solution, called SUPPLE. Others researchers have laid useful foundations. Starting with [Foley *et al.*, 1989], the model-based UI community has developed declarative, interface-representation languages; SUPPLE uses an extension of the representation developed for PEBBLES [Nichols *et al.*, 2002].

Researchers have explored many methods for "compiling" a declarative UI representation into concrete form for use on a specific target device. For example, PEBBLES uses a hand-constructed decision tree for this task; other approaches include hierarchical templates [Szekely *et al.*, 1993], stylesheets [Schreiber, 1994], and tools for facilitating

⁴Some might argue that this problem could be simply solved by using a MRU default for options like duplex, without restructuring the dialogs. But changing defaults is very dangerous unless they are visible to the user.

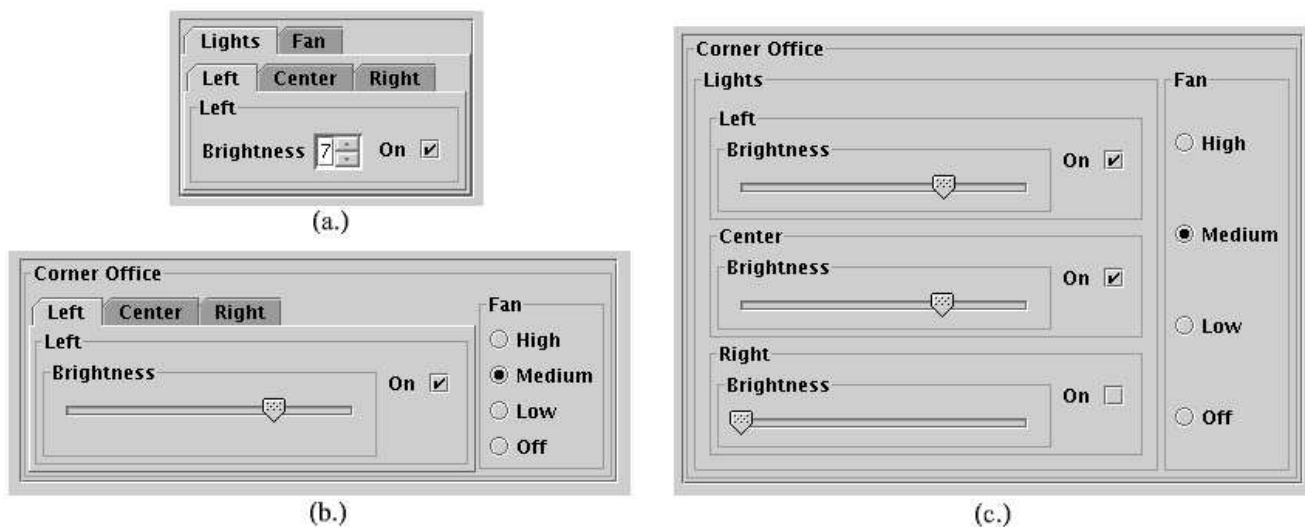


Figure 5: Given different PDA screen resolutions, SUPPLE generates different interfaces for a household controller.

manual design [Eisenstein *et al.*, 2001]. Yet another method is based on a higher-level encoding of the design in terms of *design patterns* [Lin and Landay, 2002].

In contrast, we adopt the decision-theoretic framework and search for the design with the highest expected utility among those that satisfy the device constraints. Our SUPPLE prototype focusses on screen-size constraints and uses a utility function that models the expected ease of interface operations for each widget. If multiple interface actions are necessary to complete a task (*e.g.*, if tabs must be selected, panes navigated, or confirmation buttons pressed) the costs are combined linearly. Figure 5 shows three different interfaces (from amongst the 3,888,000 possible) which were generated by SUPPLE and deemed optimal for differing amounts of available screen space.

While the current SUPPLE implementation accepts as input an arbitrary probability distribution over command usage, our framework supports user-based adaptation as described in the previous section. For example, at present SUPPLE makes an independence assumption regarding command use, but in practice people use commands in sequence and human designers recognize this by colocating the corresponding widgets. By computing the expected utility of an interface relative to a recorded trace of individual user commands, SUPPLE should be able to automatically duplicate this form of design optimization, but in a personalized manner.

6 Conclusions

Effective personalization requires improved methods for both adaptation (change based on implicit user behavior) and customization (change guided by explicit user requests). Interfaces should automatically adapt to the capabilities of the device at hand, to network connectivity, and to the user's activities, location, and context. If they wish to provide customization guidance, users should be able to control the adaptation process at any level.

This paper has briefly surveyed some of the projects at the University of Washington, which are investigating these issues, and codified several emerging principles:

- Because users find it hard to specify their preferences and goals, it is often more effective to induce them from user behavior. Version-space algebra allows an application designer to combine multiple strong biases to achieve a weaker one that is tailored to the application, thus reducing the statistical bias for the least increase in variance.
- Many benefits of model-based UI design have been long recognized [Foley *et al.*, 1989], but it is increasingly apparent that these declarative models are essential to a set of transformations (*e.g.*, Currying and casting) that facilitate adaptation.
- Since adaptive mechanisms are imperfect, the cost to users of errors must be considered alongside the advantages. Decision theory provides a powerful framework for such analysis [Horvitz, 1999], and the requisite utilities can be learned from user behavior or derived from device models.
- Certain interface mechanisms, such as defaults and automatic timeouts, can minimize the cost of errors and so increase the value of adaptation. Partitioned dynamicity offers a way to manage adaptivity while minimizing a user's cognitive load.
- Another way to enhance the user experience by improving prediction of user behavior. Because RMMs smooth, based on an abstraction lattice defined by the relational structure, they are exceptionally accurate when predicting sequential behavior (*e.g.*, a user's next action) from sparse data.

Acknowledgments

We thank James Landay for helpful discussions. This research was funded in part by Office of Naval Research Grants

N00014-98-1-0147 & N00014-02-1-0932, National Science Foundation Grants IRI-9303461, IIS-9872128, DL-9874759, and a CAREER Award.

References

- [Anderson *et al.*, 2001a] C. R. Anderson, P. Domingos, and D. S. Weld. Adaptive web navigation for wireless devices. *IJCAI-01*, Aug 2001.
- [Anderson *et al.*, 2001b] C. R. Anderson, P. Domingos, and D. S. Weld. Personalizing web sites for mobile users. *WWW-01*, May 2001.
- [Anderson *et al.*, 2002] C. R. Anderson, P. Domingos, and D. S. Weld. Relational Markov models. *KDD-02*, Aug 2002.
- [Cypher, 1993] Allen Cypher, editor. *Watch what I do: Programming by demonstration*. MIT Press, 1993.
- [Dent *et al.*, 1992] L. Dent, J. Boticario, J. McDermott, T. Mitchell, and D. Zabowski. A personal learning apprentice. *AAAI-92*, pages 96–103, July 1992.
- [Eisenstein *et al.*, 2001] J. Eisenstein, J. Vanderdonckt, and A. Puerta. Applying model-based techniques to the development of UIs for mobile computers. *IUI 2001*, pages 14–17, 2001.
- [Etzioni and Weld, 1994] O. Etzioni and D. Weld. A softbot-based interface to the Internet. *C. ACM*, 37(7):72–6, 1994.
- [Etzioni *et al.*, 1997] O. Etzioni, K. Golden, and D. Weld. Sound and efficient closed-world reasoning for planning. *Artificial Intelligence*, 89(1–2):113–148, January 1997.
- [Foley *et al.*, 1989] J. Foley, W. C. Kim, S. Kovacevic, and K. Murray. Defining interfaces at a high level of abstraction. *IEEE Software*, Jan 1989.
- [Friedman *et al.*, 1999] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. *IJCAI-99*, Aug 1999.
- [Golden and Weld, 1996] K. Golden and D. Weld. Representing sensing actions: The middle ground revisited. *KR-96*, pages 174–185, 1996.
- [Horvitz, 1999] Eric Horvitz. Principles of mixed-initiative user interfaces. *CHI-99*, pages 159–166. ACM Press, May 1999.
- [Lau *et al.*, 2000] Tessa Lau, Pedro Domingos, and Daniel S. Weld. Version space algebra and its application to programming by demonstration. *ICML-00*, pages 527–534, June 2000.
- [Lewis and Reiman, 1993] C. Lewis and J. Reiman. *Task-Centered User Interface Design: A Practical Introduction*. University of Colorado, Boulder, CO, 1993.
- [Lin and Landay, 2002] J. Lin and J. A. Landay. Damask: A tool for early-stage design and prototyping of multi-device user interfaces. *8th International Conference on Distributed Multimedia Systems*, pages 573–580, 2002.
- [Mackay, 1991] W. Mackay. Triggers and barriers to customizing software. *CHI-91*, pages 153–160. ACM Press, 1991.
- [Maes and Kozierok, 1993] Pattie Maes and Robyn Kozierok. Learning interface agents. *AAAI-93*, pages 459–465, 1993.
- [McGrenere *et al.*, 2002] J. McGrenere, R. M. Baecker, and K. S. Booth. An evaluation of a multiple interface design solution for bloated software. *CHI-02*, 2002.
- [Mitchell, 1982] T. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.
- [Nichols *et al.*, 2002] J. Nichols, B. A. Myers, M. Higgins, J. Hughes, T. K. Harris, R. Rosenfeld, and M. Pignol. Generating remote control interfaces for complex appliances. *UIST-02*, pages 161–170, 2002.
- [Norman, 1998] D. Norman. *The Invisible Computer*. MIT Press, 1998.
- [Perkowitz and Etzioni, 1997] M. Perkowitz and O. Etzioni. Adaptive web sites: an AI challenge. *IJCAI-97*, Aug 1997.
- [Perkowitz and Etzioni, 2000] M. Perkowitz and O. Etzioni. Towards adaptive web sites: Conceptual framework and case study. *Artificial Intelligence*, 118:245–276, 2000.
- [Popescu *et al.*, 2003] A. M. Popescu, O. Etzioni, and H. Kautz. High precision natural language interfaces to databases: a graph theoretic approach. *IUI-03*, Jan 2003.
- [Puerta, 1996] A. Puerta. The MECANO project: Comprehensive and integrated support for model-based interface development. *CADUI’96*, pages 19–36, June 1996.
- [Sanghai *et al.*, 2003] S. Sanghai, P. Domingos, and D. S. Weld. Dynamic probabilistic relational models. *IJCAI-03*, Aug 2003.
- [Schreiber, 1994] S. Schreiber. Specification and generation of user interfaces with the boss-system. *EWHCI-94*, pages 107–120, 1994.
- [Szekely *et al.*, 1993] P. Szekely, P. Luo, and R. Neches. Beyond interface builders: Model-based interface tools. *INTERCHI-93*, pages 383–390. ACM Press, 1993.
- [Weld, 1996] D. Weld. Planning-based control of software agents. *AIPS-96*, May 1996.
- [Wolfman *et al.*, 2001] Steven A. Wolfman, Tessa Lau, Pedro Domingos, and Daniel S. Weld. Mixed initiative interfaces for learning tasks: Smartedit talks back. *IUI-01*, pages 167–174, Santa Fe, USA, January 2001.
- [Yates *et al.*, 2003] A. Yates, O. Etzioni, and D. Weld. Reliable natural language interfaces to household appliances. *IUI-03*, 2003.

Intelligent Systems in Travel and Tourism

Hannes Werthner

e-Commerce and Tourism Research Lab (eCTRL)

ITC-irst and University of Trento, Italy

werthner@itc.it

Abstract

Travel and tourism is the leading application field in the b2c e-commerce, it represents nearly 50% of the total b2c turnover. Already in the past travel applications were at the forefront of Information Technology, i.e., the airline Computerized Reservation Systems in the early 60s. The industry and its product have rather specific features which explain this circumstance: the product is a confidence good, consumer decisions are solely based on information beforehand; and the industry is highly networked, based on world-wide cooperation of very different types of stakeholders. Consequently, this industry depends on advanced IT applications. As such travel and tourism may serve as an example of what happens and will happen in the emerging e-markets, pointing at structural changes as well as challenging application scenarios. The paper provides an overview about the industry, describes ongoing structural changes, outlines domain-specific requirements and discusses achievements and challenges in the field, following an AI and e-commerce point of view. It finishes with considerations regarding a future IT scenario.¹

1 Introduction

Despite unfulfilled business and stock market expectations, in some sectors such as the travel and tourism industry online transactions are rapidly increasing. This industry is the leading application in the b2c arena. The travel and tourism industry is witnessing an acceptance of e-commerce to the extent that the structure of the industry and the way business is conducted, is changing. The Internet is used not only for information gathering; there is an obvious acceptance of ordering services over the Internet. A new type of user is emerging; they become their own travel agents and build their travel packages themselves.

In the year 2002 European online bookings increased by 53 percent, accounting for about 3,5 % of all consumer spending in this domain, whereas in the US this number is

about 11 % (according to the Danish Center for Regional and Tourism Research). 64 Mio. Americans researched their travel options online (Travel Industry Association of America); and 32 percent of US travelers have used the Internet to book travel arrangements (see www.nua.com/surveys/). Forecasts state that by 2007 30% of all transactions in the European tourism domain will be done via the Internet, at least in the German speaking countries [Schuster, 1998].

This importance of e-commerce can be explained by the features of the industry, but it highlights also another issue, not less important: e-commerce and especially the Web are not only transaction and business oriented, it is also a medium of curiosity, of creating communities or having just fun, all of which may or may not result into business. Especially the tourism product has to do with emotional experiences; it is not just business [Werthner, 2001].

The paper is organized as follows: the following two sections describe the industry and look at the changing business environment. Section 4 provides an overview on the emerging IT landscape, it discusses domain specific requirements and related application examples, pointing at open research issues.

2 The Industry

The travel and tourism industry is a global (and a globalization) industry, with very specific features:

- Travel and tourism represents approx. 11% of the world wide GDP (following the tourism satellite account method of the World Travel & Tourism Council).
- There will be one Billion international arrivals in the year 2010 (following the World Tourism Organization). And tourism grows faster than the other economic sectors.
- It represents a cross-sectoral industry, including many related economic sectors such as culture, sport or agriculture, where over 30 different industrial components have been identified that serve travelers.
- This explains the industry's heterogeneity, and due to its SME structure it has a huge importance for regional development. For example, in the EU there are around 1.3 Mio. hotels and restaurants (9 % of all enterprises). And 95 % of them are very small, i.e., 1-9 employees.

¹ This work is partially funded by the CARITRO foundation and the European FP5 projects Dietorecs and Harmonise.

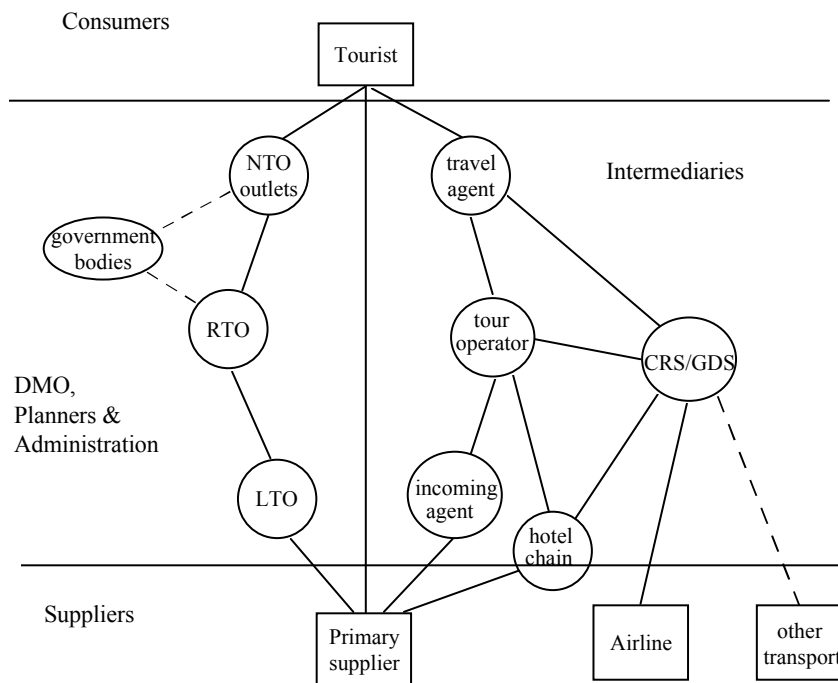


Figure 1: Structural view of the market [Werthner and Klein, 1999]

- The supply and the demand side form a worldwide network, where both production and distribution are based on cooperation.
- The product is perishable, complex and emotional: i) a hotel bed not sold for one night represents a lost income. Suppliers are in a risky situation, which can be reduced if access to information is available; ii) the tourism product is a bundle of basic products, aggregated by some intermediary. To support the rather complex bundling products must have well defined interfaces with respect to consumer needs, prices or also distribution channels; iii) vacations are emotional experience structures that involve cognitive and sensory stimulations as well as affective responses to certain events.

Tourism is an information based business, the product is a “confidence good”; an a priori comprehensive assessment of its qualities is impossible. Tourists have to leave their daily environment for consuming the product. At the moment of decision-making only a model of the product, its description, is available. This characteristic of tourism products entails high information search costs and causes informational market imperfections. Consequently, the industry has comparably long information and value chains.

Figure 1 differentiates between the supply and demand side and the respective intermediaries. Links mark the relationships as well as the flow of information, showing only the most relevant links. Nodes indicate the relevant types of players.

On the supply side we denote with “primary” suppliers enterprises like hotels, restaurants, etc., which are mostly SMEs. One should note that with respect to a functional differentiation these companies are at the same level as the “big” players like airlines. Tour operators can be seen as

product aggregators, travel agents act as information brokers, providing final consumers with the relevant information and booking facilities. CRS/GDS (Central Reservation Systems / Global Distribution Systems), stemming from the airline reservation systems already developed in the 60s, include also other products such as packaged holidays, or other means of transport. Whereas the intermediaries on the right side can be seen as the “professional” connection between supply and demand (mainly based on the electronic infrastructure of the CRS/GDS), the left side is relevant for the management, planning and branding of a destination. Normally, these entities have to act on behalf of all suppliers within a destination and are not engaged in the booking process. The links to governmental – as dotted lines – indicate that Destination Marketing Organizations are often governmental organizations.

The upstream flow of Figure 1 consists of product information, whereas the downstream flow reports on market behavior, mostly represented in terms of statistical aggregates. Both information flows create a tourist information network tying together all market participants and, apparently, reflecting the economic relationships between them.

3 The New Business Landscape

The Web facilitates new ways to meet changing consumer behavior and to reach new market segments, leading to an “informatization” of the overall tourism value chain. This allows different strategies to generate value [Sweet, 2001]:

- Value extraction: increases efficiency and reduces costs, e.g., automation of processes or outsourcing to clients

such as self check-in of hotel guests or airline passengers.

- Value capturing: client and sales data are used to support the marketing, e.g., data mining for forecast or yield management.
- Value adding: a linear combination of products / services to create richer product bundles, e.g., new service quality for consumers such as linking mobile services to existing Web services to advise tourists during their travel.
- Value creation: the focus is on network effects, e.g., tourists within a destination participate in service definition and planning.

Such strategies allow for the design of new products and services, enlarging the range of options to customize and to configure products. IT and, more importantly, improved organizational procedures lower the price of customization, enabling individualized offerings based on mass-customization. On the other side, configuration points at the bundling of different product or service components to integrated offerings. Core product can be combined with additional service elements in order to create integrated customer solutions.

Given the dynamics of the sector and the emerging already very competitive electronic market, nearly all stakeholders have implemented their Internet strategy. Travel and tourism has also become the playing field for new entrants, either start-ups or companies from the media and IT sector. Since tourism is an information-based business, it fits well with their respective background.

The overall trend points towards a further specialization and an ongoing deconstruction of the value chain paralleled by an integration of players and products. Companies will compete and cooperate at the same time, boundaries within the industry are blurring. All types of market players are affected:

- Tourists will be addressed by more players, and as they accept the Web as a booking channel, they will also play a more active role in specifying their services.
- Travel agents will see a diminishing power in the sales channel, as a consequence they will put more emphasis on consulting and complex products.

- Internet travel sites, as platforms with bundling of offerings will enhance by providing new market functionality and technology. This will lead to easier accessible price comparisons and market overviews, enabled by personalized intelligent tools for travelers.
- Destination management organizations will develop cooperation models within destinations and they will occupy a new role as consolidator and aggregator.
- Tour operator will blur the boundaries between individual and packaged tour, based on mass-customization and flexible configurations.
- CRS / GDS show an "INTEL inside" marketing strategy for major tourist Web sites to increase their transaction volume, and they also move into direct sales for the retail segment
- Suppliers will support electronic direct sales, increasing again price competition as well as price differentiation, and they will redefine customer processes (electronic ticketing, automated check-in, etc.).

This development leads to an evolution of the market best described as an ongoing interaction of concentration (e.g., as in the US with the major travel sites like Expedia, Orbitz or Travelocity) versus the simultaneous entering of new players. The related increased complexity, however, generates the need for new services such as providing transparent access or price comparisons. This enforces the design and deployment of more specialized services. Such a process will quite naturally create new business opportunities, accelerating competition and putting even more emphasis on technical innovation.

4 The IT Scenario

The described business scenario is based on flexible network configurations and the further integration of consumers into internal business processes. Adding the tourist life cycle – taking into consideration the mobility aspect of travelers – one can draw the following simplified figure of linking tourists' life cycles with companies' business processes.

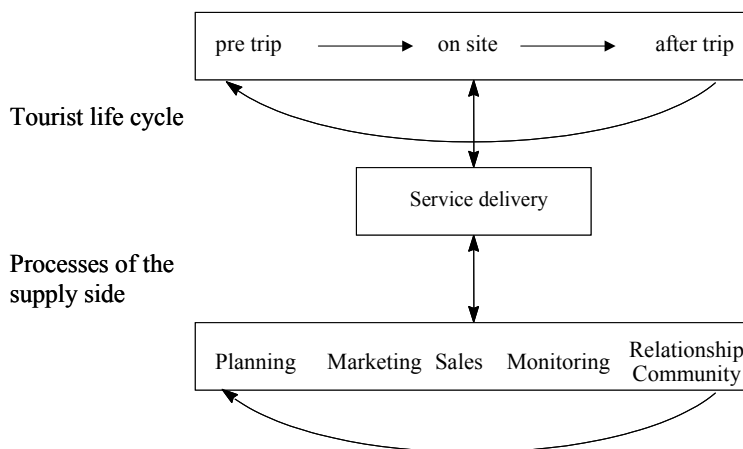


Figure 2: Tourist life cycle and companies' processes – both suppliers and intermediaries

Obviously, processes cross company borders, leading to distributed b2b2c applications, requiring cooperation between companies, and supporting also mobile communication with the consumer. Such a future business scenario is based on the assumption that technology – based on a common pervasive infrastructure – will become transparent, invisible for the consumer; information will be available at home, the work place and during travel. Given such a “holistic” approach interleaving business and technology perspectives, what are specific requirements when looking at the user – machine interaction? What is the intelligence needed to support collaboration among companies within a networked industry? These questions lead to interesting considerations regarding system architectures.

4.1 User Requirements

One could look at the human-computer interaction as a variant of the famous Turing test: as a competition between a human and an intelligent travel system.² Even if one thinks of it as an engineering task, and not as a cognitive science test, it poses hard questions: i) a simple problem like inquiring a flight from location A to B, with specific price and time constraints, complemented by a specific hotel, with similar constraints. In this case linking to a predefined database or to one of the CRS/GDS would be sufficient, if one ignores interoperability issues. This type of planning problem is already partially solved by existing online systems such as Expedia or Orbitz. But the need for more intelligent heuristics is obvious, when one thinks of the enormous number of scheduled flights and constraints such as different rates or different booking conditions within airline alliances, etc. ii) a complex problem which involves background knowledge of a specific traveler, e.g., the problem of traveling to Milan only when Milan plays. This needs the modeling of knowledge about the Italian city of Milan, the inference that in this case Milan refers to AC MILAN (a soccer club), the timing regarding soccer tournaments (normally at weekends). And other background knowledge might be needed such as specific weather conditions or cultural activities. In fact, the “leisure” domain might cover nearly all domains of daily life. And the information, stored in different bases with no common format and even unknown locations, needs to be extracted. Here wrapping techniques, learning accurate extraction rules and also adapting to structural changes in sites, are needed [Knoblock *et al.*, 2000; Kushmerick, 2000]. Other non-trivial tasks within this context are

- conversation with its semantic and pragmatic aspects,
- identification of problem domains,
- search and information retrieval with the related evaluation,
- negotiation including additional and even unsolicited offers, both having cooperative and opportunistic

² These ideas are based on discussions with W. Grossmann, University of Vienna, O. Stock, ITC-irst, A. Tjoa, TU Vienna, and W. Wahlster, DFKI.

behavior.

However, these problem types assume that users are able to express their needs, which is normally not the case. Systems have to “become active” and to provide advice, adapting to different travel decision processes as well as navigation styles [Moutinho, 1987; Ankomah *et al.*, 1996]. For example, Grabler and Zins [2002] have identified, based on a study on 200 human – human and human – machine counseling sessions, 6 different traveler groups such as highly pre-defined, recommendation oriented or geography oriented ones. These groups differ in the order in which information is searched, the type of information needed, the level of support required and the level of personal flexibility and knowledge. Two groups of factors are identified: personal, i.e., socioeconomic and psychological/cognitive factors (experience, personality, involvement), and travel features, i.e., travel purpose, travel party size, the length of travel or the distance to the destination.

An application example of such an approach is [Trip@dvce](#) [Ricci and Werthner, 2002]. Based on Case Based Reasoning (CBR), it adapts its dialogue as it learns more about the user, supports product aggregation for a given travel, provides personalized recommendations based on previous system experience, and it applies query refinement methods helping to adjust queries according to the data available in a given product catalogues. Thus, it integrates topical information and “good” examples of product bundling contained in previous travels built by a community of users.

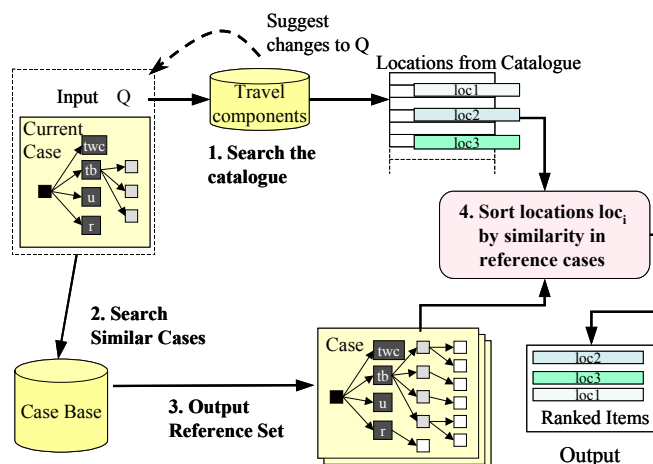


Figure 3: Sequence of interaction in [Trip@dvce](#)

Dietorecs, another example, extends this approach by integrating the iterative presentation and evaluation of alternative proposals [Fesenmaier *et al.*, 2003]. It assumes that by presenting specific images, which implicitly represent the most discriminating offers within a given product catalogue, the evaluation of the user provides indications about his/her references. At the beginning the systems performs a cluster analysis of the offers within a given catalogue. The user’s choice is the starting point of an iterative process of refocused clustering and presentation. At

the end, either the user directly takes the chosen product, or the obtained criteria serve as input for a CBR process.

These examples show, that different decisions styles as well as the seamless switching between them have to be supported. But the choice on tourism products is not rational; these products are complex experience structures that involve cognitive and sensory stimulations. Due to the multi-sensory nature of tourism products, textual ways of describing vacations are very limited to convey a complete picture that can be used to formulate product expectations serving as a basis for decision.

A possible approach is to look at product trails – these are sources of information for the formation of brand beliefs since they involve an experience of the product through multiple senses – describing leisure experiences and to draw conclusions on how to present tourism information and how to guide a user in a non rational navigation process. Gretzel and Fesenmaier [2003] looked at 3000 randomly selected persons who were asked to imagine a trip and to report the colors dominating their mental image, the scents they would like to smell, and the sounds they expect to hear. They identified a number of such sensory categories, which were clustered to experiences representing coherent bundles such as Autumn or Scenic / Nature. These findings indicate that sensory domains and associations thereof exist and that these follow specific patterns. Such bundles can be used to describe coherent experiences sought after by certain groups of travelers. These results can provide substantial input for interface design and user guidance.

Finally, travelers need access to information whenever and wherever they want, making tourism a perfect application field for mobile computing. Latest generations of mobile devices and wireless networks offer new opportunities, but mobile devices still suffer from restrictions compared to web based systems. Moreover, the mobile context imposes a very different interaction style and decision model. But, on the other side, mobile applications could take advantage when linked to existing web sites and refer to already existing personal profiles and selected travel products. The mobile Intelligent Travel Recommender (mITR) supports travelers to complement their product while traveling [Ricci *et al.*, 2002]. The objective of mITR is to improve iteratively the “fitness” of recommended products. But instead of asking the user for explicit input, mITR exploits the knowledge contained in the pre-travel plan to guess an initial set of candidate products. The user can provide critics to some of these products, such as “I like this feature of this product” or “I want something less expensive”, which is used for further interaction.

4.2 System Features – Intelligent Networking

System designed to support such applications should

- allow users to access information whenever and wherever they want,
- guide the user in the decision and traveling process, and
- allow the user to create product bundles.

Business services for the traveler may use Web services to be more easily integrated and bundled into new, ad-hoc

products. Web services provide an open, flexible, modular, layered architecture that can integrate multiple communication models. But these services do only facilitate the “manual” combination of components across different platforms. The vision is that this should be available (semi-) automatically, for example, allowing a customer to plan a trip in the city of Vienna visiting restaurants with at least 15 gourmet points and some classical concerts. Another enabling technology might be Peer-to-Peer (P2P) services to directly exploit resources present at other nodes without intervention of any central server, where nodes may join and be integrated in an ad-hoc manner. Such services need to be described for the respective peer device and discovered in an ad-hoc manner in a decentralized scheme, without a centralized registry as this may be unavailable in a mobile environment; to be composed ad-hoc and to interact not only with servers, but also with other peers; and to be connectable to legacy systems.

However, when taking the supply side’s point of view, the requirements become even more complicated: For instance, a middleman might search for 100 individual web services hosted by several individual, privately-owned hotels and with golf courses nearby in the Tyrol within a diameter of 100kms in order to bundle them with flights into tourist packages. In this case functions are needed to

- enable middlemen to create sets of bundled products,
- ease the participation of SMEs to larger networks, and
- facilitate networking leading to virtual organizations.

In this context networked business operations could be defined as P2P enabled semantic Web services that integrate business networks allowing for an n:m organization and communication between participating enterprises. This transfers the level from composing individual instances of services to composing sets of services; and from ad-hoc integration between two participants to an arbitrary number of cooperating enterprises. The orchestration of a large set of web services requires possibilities for business planning with more refined means. Techniques of constraint reasoning, multi-value optimization and relaxation might be used to aggregate web services at the set level and to achieve specific business goals, e.g., profit optimization or the equal distribution of income within a destination.

However, such a scenario is based on a non-existing assumption: there exists no agreed electronic standard to describe tourism services. Those existing such as the OTA (Open Travel Alliance) specifications are just adopted by few players. Given that the number of available tourism relevant information sites is huge, e.g., out of the 1.3 Mio European enterprises in this sector nearly 40% have already their Web site, it can be foreseen that also future standardization initiatives will fail.

One approach to this problem is the European project Harmonise, creating an ontology based mediation and harmonization tool [Dell’Erba *et al.*, 2002]. The project puts emphasis on the combination of a social consensus process with the application of new technologies. The goal is to allow participating tourism organizations to keep their proprietary data format while cooperating with each other.

Specific tourism mediators are dedicated to the “translation” needs between these data sources. Such a mediator looks at information from a higher conceptual semantic level using this level of abstraction for the mapping purpose.

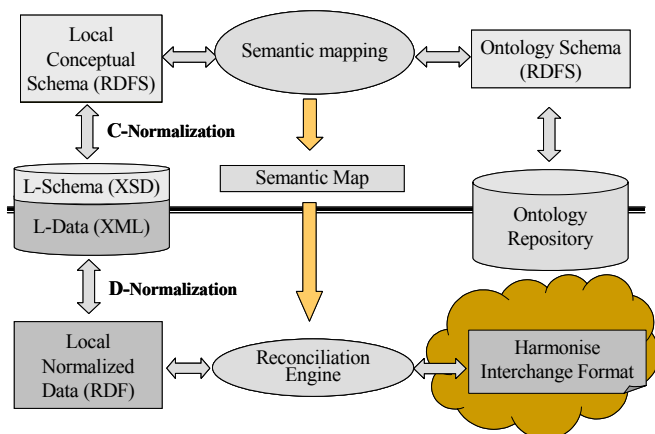


Figure 4: The Harmonise process

The data model of a source document, assuming XML, is first lifted to a local conceptual schema (C-Normalization) and then semantically mapped to the terminology specified by the shared ontology, which is built by domain experts. The output of the mapping process is a set of reconciliation rules, which are used in order to transform the local data and to code them according to the ontology content. Harmonise is based on RDF(S) as “language” for representing local conceptual schemata as well as the mediating ontology. But Harmonise has several limitations: i) it does not solve the problem of ontology versioning, which is severe in a domain with constantly new products; ii) and Harmonise, though mapping between different conceptual models, does not support ontology reasoning, or even more advanced concepts such as approximate reasoning, needed in a domain which is based not only on true axioms or facts.

5 Conclusions

The tourism business is changing. More specialized services, flexible network configurations and further consumer integration into internal business processes will lead to smart market places, integrating all stakeholders. The underlying IT scenario, enabling as well as enforcing this development, shows that tourism is an interesting field of application. But it also demonstrates that this industry poses hard challenges, which offer interesting research opportunities. This has also been acknowledged by the European Union, placing tourism as one of the preferred application fields of the European IT research.

References

[Ankomah *et al.*, 1996] P.K. Ankomah, J. L. Crompton and D. Baker. Influence of cognitive distance in vacation choice. *Annals of Tourism Research*, 23(1):138-150, 1996.

[Dell’Erba *et al.*, 2002] M. Dell’Erba, O. Fodor, F. Ricci and H. Werthner. Harmonise: a solution for data interoperability. In *Proceedings of IFIP I3E 2002 Conference*, pages 114-127, Lisbon, Portugal, Oct. 2002.

[Fesenmaier *et al.*, 2003] D. Fesenmaier, F. Ricci, E. Schaumlechner, K. Wöber and C. Zanella. DIETORECS: Travel Advisory for Multiple Decision Styles. In *Proceedings of the ENTER 2003 Conference*, pages 232-242, Helsinki, Finland, January 2003.

[Grabler and Zins, 2002] K. Grabler and A. Zins. Vacation trip decision styles as basis for an automated recommendation system: Lessons from observational studies. In *Proceedings of the ENTER 2002 Conference*, pages 458-469, Innsbruck, Austria, January 2002.

[Gretzel and Fesenmaier, 2003] U. Gretzel and D. Fesenmaier. Experience-based Internet Marketing: An Exploratory Study of Sensory Experiences Associated with Pleasure Travel to the Midwest US. In *Proceedings of the ENTER 2003 Conference*, pages 49-57, Helsinki, Finland, January 2003.

[Knoblock *et al.*, 2000] C. A. Knoblock, K. Lerman, S. Minton and I. Muslea. Accurately and reliably extracting data from the web: A machine learning approach. *IEEE Data Engineering Bulletin*, 23(4):33-41, 2000.

[Kushmerick, 2000] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence Journal*, 118(1-2):15-68, 2000.

[Moutinho, 1987] L. Moutinho. Consumer behavior in tourism. *European Journal of Marketing*, 21:2-44, 1987.

[Ricci and Werthner, 2002] F. Ricci and H. Werthner. Case-based querying for travel planning recommendation. *Information Technology and Tourism*, 4(3-4):215-226, 2002.

[Ricci *et al.*, 2002] F. Ricci, D. Cavada and Quang Nhat Nguyen. Integrating Travel Planning and On-Tour Support in a Case-Based Recommender System. In *Proceedings of the Workshop on Mobile Tourism Systems* (in conjunction with Mobile HCI’02), pages 11-16, Pisa, Italy, September 2002.

[Schuster, 1998] A. Schuster. A Delphi survey on electronic distribution channels for intermediaries in the tourism industry: the situation in German speaking countries. In *Proceedings of the ENTER 1998 Conference*, pages 224-234, Istanbul, Turkey, January 1998.

[Sweet, 2001] P. Sweet. *Designing Interactive Value Development*. PhD. Thesis, Lund University, Lund Business Press, Sweden, 2001.

[Werthner, 2001] H. Werthner. Just Business – Shouldn’t We Have Some Fun? In *Proceedings of the ECWEB – DEXA Conference*, pages 1-16, Munich, September 2001.

[Werthner and Klein, 1999] H. Werthner and S. Klein. *Information Technology and Tourism – A Challenging Relationship*. Springer Verlag, Wien, New York, 1999.

