University of Dayton
eCommons

Computer Science Faculty Publications

Department of Computer Science

2004

Program Transformations for Information Personalization

Saverio Perugini
University of Dayton, sperugini1@udayton.edu

Follow this and additional works at: http://ecommons.udayton.edu/cps_fac_pub

Part of the Databases and Information Systems Commons, Graphics and Human Computer Interfaces Commons, OS and Networks Commons, Other Computer Sciences Commons, Programming Languages and Compilers Commons, Systems Architecture Commons, and the Theory and Algorithms Commons

eCommons Citation

Perugini, Saverio, "Program Transformations for Information Personalization" (2004). *Computer Science Faculty Publications.* Paper 41.
http://ecommons.udayton.edu/cps_fac_pub/41

This Dissertation is brought to you for free and open access by the Department of Computer Science at eCommons. It has been accepted for inclusion in Computer Science Faculty Publications by an authorized administrator of eCommons. For more information, please contact frice1@udayton.edu, mschlangen1@udayton.edu.

Program Transformations for Information Personalization

Saverio Perugini

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Advisory Committee:

Naren Ramakrishnan, Chair
Edward A. Fox
Mary Beth Rosson
Manuel A. Pérez-Quiñones
A. Lynn Abbott

May 14, 2004
Blacksburg, Virginia

Keywords: Program Transformations, Personalization, Partial Evaluation, Program Slicing, Out-of-turn Interaction, Hierarchical Hypermedia, Web Modeling, Information Retrieval

Copyright © 2004 by Saverio Perugini
ALL RIGHTS RESERVED

Program Transformations for Information Personalization

Saverio Perugini

Abstract

Personalization constitutes the mechanisms and technologies necessary to customize information access to the end-user. It can be defined as the automatic adjustment of information content, structure, and presentation. The central thesis of this dissertation is that modeling interaction explicitly in a representation, and studying how partial information can be harnessed in it by program transformations to direct the flow of the interaction, can provide insight into, reveal opportunities for, and define a model for personalized interaction. To evaluate this thesis, a formal modeling methodology is developed for personalizing interactions with information systems, especially hierarchical hypermedia, based on program transformations. The predominant form of personalized interaction developed in this thesis is *out-of-turn interaction*, a technique which empowers the user to take the initiative in a user–system dialog by providing unsolicited, but relevant, information out-of-turn. Out-of-turn interaction helps flexibly bridge any mismatch between the user's model of information seeking and the system's hardwired hyperlink structure in a manner fundamentally different from extant solutions, such as multiple faceted browsing classifications and search tools. This capability is showcased through two interaction interfaces using alternate modalities to capture and communicate out-of-turn information to the underlying system: a toolbar embedded into a traditional browser for out-of-turn textual input and voice-enabled content pages for out-of-turn speech input. The specific research issues addressed involve identifying and developing representations and transformations suitable for general classes of hierarchical hypermedia, providing supplemental interactions for improving the personalized experience, and studying user's (out-of-turn) interactions with resulting systems.

For my parents and grandparents,
for love, support, and presence.

In Loving Memory of

Olimpia Verde DeVito
(1898–1994)

sister,
wife,
mother,
grandmother,
great-grandmother,
matriarch, and
friend.

Acknowledgments

> "You can't run with the hares *and* hunt with the hounds."
>
> Anonymous, but popularized by N. Ramakrishnan

I pay all homage to Jesus Christ, my Lord, Savior, and Redeemer, through whom all things are possible.

Pursuing a Ph.D. is like a boxing match. During the course of the match, the boxer is expected to take some really good shots (the agony of the quals, the sleepless nights spent preparing a proposal), but persevere, and prevail victoriously (bringing structure to amorphous research issues). This match has been a labor of love, the best experience of my life, one which no money or fancy company stock options could buy, one which I would trade for no other. As the popular advertising slogan goes: 'completing an undergraduate degree: expected; aspiring to get a Master's: ambitious; pursuing a Ph.D.: *priceless.*' Moreover, there is a certain intangible and irresistible camaraderie and solidarity among Ph.D. students which permeates and enriches the soul. Few things in life are as beautiful. Thus, special thanks are due to several instrumental people for making this process possible and so enjoyable.

Chief among them is my thesis advisor, Naren Ramakrishnan. Naren is a special person. Anyone who has ever interacted with him knows that he is cut from a different cloth. I admire Naren's wisdom and creative approach to research and thank him for sharing it with me. Our relationship flourished from the traditional advisor–student nature, when I first joined his group, to assume a three-dimensional nature today: advisor (mentor), father-figure (role model), and friend (peer). He has been my best friend throughout my graduate studies. I would have never survived my graduate program were it not for the energy and inspiration that I draw from him, mostly by example. He always challenged me to ask tough questions of myself and aspire for 'high-hanging fruit.' His dedication to his profession is reified by his concern for his graduate students. He has been a tremendous source of ideas, especially by recommending that I look at program slicing which ultimately became a major ingredient in my research. In addition, Naren knew exactly when to let me struggle with my research and when to intervene. I'd be remiss if I did not mention the countless hours he spent, often into the early morning or at the Mill Mountain café (Naren's second office), working with me to push out papers; especially working patiently with me for several hours, despite having many of his own personal issues to tend to, in the Math Emporium the night before this document went to the committee. I will never forget those sessions Naren; they were my best times in school. I hope that some day I can express my appreciation by mentoring my students in his image. This world needs more Naren Ramakrishnans.

I also have been fortunate to work with an energetic, supportive, and encouraging advisory committee, comprised of researchers regarded as experts in their fields. Professor Ed Fox is the most down-to-earth professor I have ever met. His concern for the well-being of his students is not only reflected in his support for and feedback on their work, but also in his development of respectable computing professionals. Professor Fox went out of his way during the holidays to review my application materials for faculty positions. His feedback was invaluable to presenting a well-organized, focused, and polished dossier. I also thank Professor Fox for the suggesting the usage of 'structural' and 'terminal' information to qualify interaction sequences.

I thank Professor Mary Beth Rosson for ideas and feedback on the HCI aspects of my work, especially the study presented in Chapter 6. I have always left Professor Rosson's office enamored by the connections she draws. I also thank her for introducing Mary Pinney, an ISE graduate student, funded by NSF SGER grant IIS-0136182, to our research. I thank Dr. Manuel A. Pérez-Quiñones for bringing his expertise in dialog-based computer systems, especially from an HCI perspective, to bear upon my work. He has been actively engaged in our research since its birth and has contributed to papers describing the study in Chapter 6. Dr. Pérez-Quiñones also lent me his Ph.D. dissertation for several months. Since his dissertation already marries dialog notions with programming language concepts, it was foundational to my work. Lastly, I thank Dr. A. Lynn Abbott for joining my committee as an external faculty member and contributing suggestions from an intelligent systems perspective.

Several other faculty and colleagues have helped in various ways. Thanks are thus due to Don Goelman (Villanova University); Charlie Aull, Reza Barkhi, Rex Hartson, Lenny Heath, T. M. Murali, Alexey Onufriev, Cal Ribbens, Layne Watson, and Chad Wingrave (Virginia Tech); and Craig Struble (Marquette University). Special thanks to John Lewis at Villanova University who mentored me as an undergraduate and encouraged me to pursue a Ph.D. John has always believed in me and supported my endeavors. Much of my success is a reflection of his guidance. I thank Mary Pinney for conducting the user sessions for the study presented in Chapter 6 and all the Virginia Tech students who participated. Many thanks are due to Chris Williams, MS grad student and programmer extraordinaire, for building the interaction manager through which my transformation engine personalizes interactions for end-users. I also thank him for the many hours he spent debugging code as well as babysitting me on how to produce slick screen captures for several of my papers, including this dissertation. I also would like to thank Rob Capra, a fellow Ph.D. student, who occupied the carrel adjacent to mine. He has continually provided feedback on my work, especially its HCI aspects, and been a great person off whom to bounce ideas. My thanks go to my dear friend and fellow graduate student Padmapriya Kandhadai who not only proofread several of my manuscripts, many of which she was not a co-author, but also provided needed emotional support and love. Thanks to Dan Moisa, an MS grad student, who assisted me with my TA grading duties during the most demanding times of my final year. I also thank the administrative assistants in the CS Department, particularly the soft-spoken, lovely, and generous Carol Roop, who always went out of her way to help me, especially in preparing my faculty applications. Finally, I express my thanks to the illustrious director of the Terascale Computing Facility at Virginia Tech, Srinidhi Varadarajan ('the hatted one'), for nonchalantly recommending that we name our toolbar *Extempore*, highlighting its, in addition to his, *extemporaneous* nature.

I acknowledge NSF SGER grant IIS-0136182 for funding this research. In addition, NSF DUE-0136690 and DUE-0121679 grants supported the CITIDEL project which I used as a case study for my techniques. Moreover, several computing organizations have recognized my work with awards and associated grants. Acknowledgments are thus due to ΥΠΕ/IEEE-CS for the Academic Excellence Award, the Washington, D.C. Chapter of the ACM for the Samuel N. Alexander ACM Fellowship, the Virginia Tech College of Engineering for the P. E. Torgersen Graduate Student Research Excellence Award, and the Virginia Tech Graduate Student Assembly for Research Development Project and Travel Fund Program grants.

Apart from those directly engaged in my work, my life has been influenced by interactions with several other amazing people who have helped in various ways. I have been blessed with the best friends one could ever wish for. This thesis would not have been brought to successful fruition without my *strong* support network of friends—Chris Barbee, Greg Burnett, Tim Costello, Reverend Jim Griffin, Surya Kodukula, Joe Ruscio, Dimitri Savvas, and Omar Vasnaik (local to Blacksburg); Joe Ciarleglio, Nicky Guerrera, Dan McMillan, and Ryan Murphy (local to Connecticut and Villanova)—who I could always count on when faced with a crisis. I am proud to say that the best times of my life were shared with you.

Three of them merit special remarks. Dimitri Savvas has been in my corner since my first day at Virginia Tech in 1998 when I met him. He has invested a lot of time and energy in believing in me. Thank you Dimitri for keeping me focused by reminding me, often and when necessary, that getting my Ph.D. was the most important priority in my life. Many thanks to Dan McMillan, the closest I have to a brother, whose unrelenting support and encouragement was vital to this thesis. Thank you Dan, most of all, for your presence in my life, the experiences shared, and ‘the wonderful memories.’ Thanks for implicitly opening my eyes to the paradox: ‘While life is short, it must be approached as a marathon, not a sprint.’ And special thanks to my spiritual advisor, Reverend Jim Griffin who has shepherded me beyond my wants, beyond my fears, from death unto life. I never thought I would find the most influential Catholic priest in my life at a large state university in southwestern Virginia.

Lastly, I extend my deep appreciation to my family for everything they have done for me. I was fortunate to grow up with seven primary role models—my parents (Saverio and Georgeanna Perugini), four grandparents (Nicola and Giuseppina Perugini; and George and Lucia Daloia), and uncle (Guido Guerrera)—who have unconditionally loved and supported me throughout my life. I am proud that you all are here in good health to see me pass this milestone. My success is a testament to your lives. Thank You.

Saverio Perugini
Blacksburg, Virginia
May 14, 2004

This document was typeset in LaTeX 2_{ε} on a 15” Apple PowerBook G4.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> "I believe that the role of the successful engineering researcher is to understand developers' problems, but to use the luxury of not having to meet short-term deadlines, to look for the underlying causes and fundamental cures rather than immediate, symptomatic, relief. Developers, who must meet pressing market driven deadlines, do not have the time to look for long-term solutions. That is the researcher's job."
>
> D. L. Parnas, in [Par98]

The rapid growth of the World Wide Web (WWW) and the concomitant increase in online content have greatly exacerbated information overload and ignited research in personalization. Personalization constitutes the mechanisms and technologies necessary to customize information access to the end-user. It can be defined as the automatic adjustment of information content, structure, and presentation. Commercial websites increasingly employ personalization to help retain customers and reduce information overload. For example, Amazon's e-commerce site is estimated to have at least 23 different types of personalization [Rie01]. Elements of the personalization landscape include search engines [LG98], recommender systems [PGF04], and adaptive websites [Bru01]. Even a cursory survey of the articles in the August 2000 *Communications of the ACM* issue on this topic [Rie00b] reveals that the scope of personalization extends to many different forms of information content and delivery, and transcends diverse application domains. The advent of multimodal and mobile computing devices, with varying input and output (display) capabilities, has made personalization not only attractive, but necessary; and as a result personalization technologies are now widely believed to be critical to sustaining the Internet economy [SV99].

What does it mean for an information system to be *personable*? The WWW community has interpreted personalization in multiple ways [Rie00b]. For example, some websites, especially e-commerce sites, welcome returning users and remember personal details (e.g., credit card numbers). Other sites track purchase patterns and recommend specific products. Still others provide browsing aids, such as top-10 visited links. Similarly, there are many ways of studying personalization, e.g., by the level at which content is customized, relevance of information displayed to the user, speed of information access, or more qualitative criteria such as utility and customer satisfaction.

In this thesis, I argue that an information system is personable if a user can interact with it in an expressive manner to achieve her information-seeking goals. Thus, my view of personalization is oriented toward *personalizing interaction.* This approach is holistic and has been influenced by the seminal work of Marchionini [Mar97]. At a time (circa 1995) when information system researchers were focused on information retrieval (IR) models and algorithms, Marchionini championed the study and representation of interaction as a basis for designing systems. Pednault has declared that 'representation is everything' [Ped00]. My broad research goal is to bring this viewpoint to bear upon personalization and develop a modeling methodology for information personalization.

1.1 My Thesis

It is helpful to think of interaction with an information system as a dialog between the user and the system. The goal of personalization then is to support a flexible dialog between the user and information system, and to support the user in achieving his information-seeking goals. For instance, dialogs are useful in refining imprecise requests [Roc71], clarifying task goals [BCST95], and in general bridging the user's mental model of the document space with the information system's representation [Wil84].

The central thesis of my work is that modeling interaction explicitly in a representation, and studying how partial information can be harnessed in it by program transformations to direct the flow of the interaction, can provide insight into, reveal opportunities for, and define a model for personalized interaction. I evaluated this thesis by casting various forms of personalized interaction with hierarchical hypermedia as program transformations over suitable representations and by studying user experiences with systems affording this interaction.

The context for my work arises from the PIPE (Personalization is Partial Evaluation) project [Ram00]. PIPE is a modeling methodology for personalizing information-seeking dialogs by transforming programmatic representations of such dialogs. In particular, PIPE models personalization as a form of *partial evaluation* [Jon96], an automatic technique for specializing programs given some, but not all, of their input. A sequence of such partial evaluations corresponds to a personalized interaction for the user. Thus, the main theme of my research is to *programmatically represent* interactions with information systems, and use *program transformation techniques* for personalization based on *partial user input*:

Representation × Transformation × Partial Input ⇒ Personalized Interaction

One of the fundamental research issues in information systems research is the development of models which allow the specification and realization of information-seeking interactions. Besides formalizing important operations, such models provide a vocabulary to think and reason about the information-seeking activity. In this sense, the PIPE methodology allows us to model interaction with an information system as a program, uses partial assignments of program variables to capture user input, and employs partial evaluation to realize user-specified interactions. In addition, PIPE enables us to reason about how the programmatic model should be structured, in order to realize a targeted set of interaction sequences.

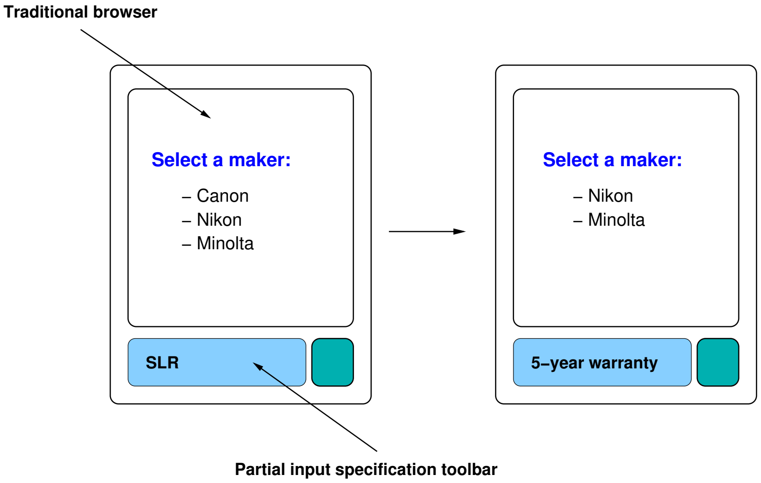


Figure 1.1: Example of a personalized interaction with a website using a toolbar embedded into a traditional browser. The top window of the interface supports traditional browsing functionality. At any point in the interaction, in addition, the user has the option of supplying personalization parameters out-of-turn (using the toolbar at the bottom) to conduct personalization.

1.2 Setting

While the next chapter provides details of PIPE and specifics on the form of personalization it achieves, I give a sneak peak into the basic idea here. I focus primarily on hierarchical hypermedia (e.g., websites). In this domain interaction is typically characterized by a progressive mode of information seeking (i.e., drill-down) where the content sought (e.g., a document, webpage, or value) lies at the end of an interaction sequence. For instance, Fig. 1.1 depicts a website which solicits choices of camera attributes to complete the specification of a camera. The site initially solicits a choice of maker from the user. The user, however, decides to not pursue any of the presented hyperlinks and instead supplies unsolicited, but nevertheless relevant, information about lens type using a toolbar embedded into the browser (Fig. 1.1, left). Such a partial, *out-of-turn* input from the user is used to partially evaluate a representation of the camera dialog, revealing that the Canon manufacturer does not carry a camera with the desired lens. This causes the manufacturer choices to be pruned (Fig. 1.1, right). The site again prompts the user for his choice of maker, since there are still two options left. Once again, the user opts to utilize the toolbar to provide a specification of warranty information out-of-turn (results not shown). I call this technique of supplying unsolicited information, *out-of-turn interaction*. It is a technique which empowers the user to take the initiative in a web dialog by supplying unsolicited, but expected, information while browsing. A sequence involving a mixture of responsive (via hyperlink clicks) and out-of-turn (by typing into the toolbar) inputs corresponds to a personalized interaction for the user.

This seemingly simple example is actually a powerful demonstration of the importance of an explicit representation of interaction used to *stage* the interaction. As I show in the next chapter, both responsive and out-of-turn inputs are partial, and hence the same transformation technique (i.e., partial evaluation) can support both browsing and out-of-turn interactions through the same representation. In addition, since the transformation operator is closed, the user is able to interleave these basic modes of information seeking in any manner he desires, leading to a *mixed-initiative* mode of interaction, without the designer anticipating all the forms of interactions which must be supported. Finally, by explicitly recognizing the role of a representation, I am led to the attractive possibility of investigating alternative programmatic representations and transformations and formally characterizing the classes of personalized interactions which they enable. This aspect constitutes the creativity in my work.

1.3 Research Questions

The crux of my research thus involves designing representations of interaction and developing program transformation techniques to realize a desired form of personalization.

The semantics of an out-of-turn interaction are salient in a website such as the camera example where levels of the site correspond to distinct facets of camera classification, namely maker and model, and where values for these attributes are mutually-exclusive (e.g., 'a camera cannot be made by both Canon and Minolta'). The benefits of supplying partial information out-of-turn are apparent, since these aspects can be communicated in any order. Less obvious is how to support out-of-turn interaction in websites which are not organized in such a levelwise, mutually-exclusive manner. In addition to being not-levelwise and not-mutually-exclusive, many websites are properly modeled as DAGs (Directed Acyclic Graphs; one simple example is in the presence of crosslinks such as those in Yahoo! whose hyperlink labels are augmented with '@'). The primary research question of this dissertation is hence aimed at formalizing interpretations of out-of-turn interaction in generalized websites:

RQ1: How can we model out-of-turn interaction with general classes of hierarchical hypermedia?

I studied this question by capturing the processing of partial input in general classes using generalized program transformations such as program slicing [BG96] and modeling crosslinks through program factoring techniques (e.g., procedural abstraction). This allowed me to formally cast out-of-turn interaction with many classes of websites in a single framework [NWPR04]. In addition, I built a purely functional (i.e., side-effect free), stateful, and robust web transformation engine based on this more general concept.

In studying out-of-turn experiences of users, I was led to the importance of supporting alternative dialog options, in particular *meta*- and dynamic dialog restructuring capabilities. The next research question aims to supplement out-of-turn interaction with new personalization primitives, without disturbing the program transformation viewpoint:

RQ2: What other program transformations are useful for personalizing interaction?

This question was studied by developing a taxonomy of program transformation techniques, studying the interactions they support, and investigating how they can be combined to augment out-of-turn interaction.

It is important to note that the use of an out-of-turn interaction interface as in Fig. 1.1 allows the user to circumvent any originally intended navigation flow at a website. This functionality provides a form of personalization which flexibly reconciles any mismatch between the user's model of information seeking and the site's hardwired organization in a manner fundamentally different from multiple faceted browsing classifications [HEE$^+$02] (e.g., Epicurious.com) and site-specific (e.g., power search of Amazon.com) or general search tools (e.g., Google). Out-of-turn interaction is different from faceted browsing interfaces because it does not hardwire all possible interaction sequences in the hyperlink structure. It also is different from a search engine since, unlike the latter, it does not curtail the interaction. As a result, an out-of-turn interaction interface supports a form of interaction not familiar to web users. I built two interaction interfaces—a toolbar embedded into a traditional web browser [PR03b] and a multimodal voice interface [NWPR04]—to capture out-of-turn input and communicate it to the underlying system. My final research questions are aimed at exposing users to out-of-turn interaction, and exploring how they employ it in conjunction with their browsing interactions with a site.

RQ3: How do users employ out-of-turn interaction?

RQ4: What is their rationale for choosing out-of-turn interaction in information seeking?

I approached these questions through studies with users in a targeted website to determine if users employ out-of-turn interaction for information-finding tasks, and gathering rationale for doing so through think-aloud and retrospective protocols. I found that users were adept at discerning when it is necessary to interact out-of-turn. I also observed that out-of-turn interaction is better assimilated when introduced using multiple modalities [PPR$^+$03]. In addition to studies with users, to assess the research results, I developed evaluation metrics which formally characterize many aspects of representations and transformations, e.g., their soundness, completeness, sufficiency, compression ratio, and representational adequacy.

1.4 Outcomes

By studying personalization from a representational and transformational perspective, my research (i) brings a formal, theoretical, and original approach to the subject, (ii) provides a systematic and functional approach to designing systems, and (iii) provides an implementation-neutral way to study software frameworks for personalization. The previous three properties are absent from the nascent field of personalization and thus the resulting modeling methodology for information personalization is my most significant contribution. In addition, supporting out-of-turn interaction via multiple and multimodal interaction interfaces in devices with varying capabilities allows us to cast personalization in a variety of contexts.

1.5 Reader's Guide

This document is organized as follows:

Chapter 2: An introduction to the program transformation approach to information personalization [PR03b].

Chapter 3: A survey of related research identifying three progressive tiers of systems, classified by the interaction they afford to users [PR03a].

Chapter 4: A formalization of out-of-turn interaction, involving several lemmas, examples, and evaluation criteria [PRa, PRb].

Chapter 5: A description of interactions supplemental to out-of-turn interaction [PRF04], and a software framework for out-of-turn interaction [NWPR04].

Chapter 6: An account of a study exploring out-of-turn interaction in a website and users' rationale for employing it [PPR$^+$03].

Chapter 7: An overall discussion of the research presented, its contributions, and future work.

Appendices:

A. XSchema OTML Language Definition.
B. A survey of program transformations.
C. A survey of program transformation systems.
D. A problem-solving scenario for evaluating interaction interfaces.

Chapter 2

Personalizing Websites with Mixed-Initiative Interaction

> "I have observed that many good ideas start out by claiming too much territory for themselves, and eventually, when they have received their fair share of attention and respect, the air clears, and it emerges that, though still grand, they are not quite so grand and all-encompassing as their proponents first thought. But that's all right. .... That would be a fine start."
>
> D. Hofstadter, in *Analogy as the Core of Cognition* [Hof02]

Mixed-initiative interaction between two participants is one where the parties can take turns at any time to change and steer the flow of interaction. In this chapter, I show how mixed-initiative interaction with websites can be achieved using the novel transformation approach alluded to in the previous chapter. This approach leads to a personalized experience for users and, at the same time, naturally lends itself to a simple implementation strategy for the website designer. This chapter describes the underlying approach, implementation experiences, and many potential directions for supporting rich forms of personalization. It serves as a foundation for the more complex interactions studied later.

2.1 Dialogs: Fixed- and Mixed-initiative

To see what a personable interaction with a website can be, consider the following human–human dialogs between a camera consumer and a dealer:

Dialog 1

1 **Dealer:** How may I help you?
2 **Consumer:** I am looking to purchase a camera.
3 **Dealer:** Sure, is there a particular manufacturer you are interested in?
4 **Consumer:** Nikon.
5 **Dealer:** What type of Nikon camera would you like?
6 **Consumer:** An SLR model.

7 **Dealer:** Sure, we have those. Now, . . .
(conversation continues to ascertain more details)

Dialog 2

1 **Dealer:** How may I help you?
2 **Consumer:** I am looking to purchase a camera.
3 **Dealer:** Sure, is there a particular manufacturer you are interested in?
4 **Consumer:** Not really, but it has to be SLR.
5 **Dealer:** I see. Only Nikon and Minolta make SLR cameras.
6 **Consumer:** Okay, in that case, . . .
(conversation continues)

Both conversations involve the specification of camera attributes but differ in important ways. In the first dialog, the consumer responds to the questions in the order they are posed by the dealer. The dealer has the initiative at all times and such an interaction is thus referred to as a *directed* or *system-initiated dialog.* In the second dialog, when the dealer prompts the consumer about camera manufacturer, the consumer responds with information about his choice of lens type instead (SLR (Single-Lens Reflex); see line 4 of *Dialog 2*). I say that the consumer has taken the conversational initiative from the dealer. Nevertheless, the conversation is not stalled and the dealer continues soliciting other aspects of the information-gathering activity. In particular, the dealer registers that the consumer has answered a different question than the one he was asked, and the dealer refocuses the dialog in line 5 to the issue of manufacturer (this time, narrowing down the available options). Such a conversation—where the dealer and consumer exchange initiative—is called a *mixed-initiative interaction* [ABD$^+$01, NS97].

My goal is to provide the user with an interaction interface to take the initiative while browsing a website. Such an interface, when used in conjunction with hyperlink clicks, helps bring mixed-initiative interaction to the web.

2.2 Mixed-initiative Interaction

How can we have a similar flexibility of interaction with a website? More fundamentally, what does it mean to take the initiative from a website? Users predominantly interact with websites by clicking on presented hyperlinks. Any time a user clicks on a hyperlink, she is responding to the choices already put forth by the website. In other words, interactions between a user and a website are primarily system-initiated and reflect directed dialogs. Browsing is, hence, not mixed-initiative, because the initiative always resides with the site. As a result, browsing only supports a strict, one-size-fits-all interaction paradigm. Site designs which are hardwired to disable some interaction sequences can be called 'unpersonalized' w.r.t. the user's mental model of information seeking.

Given this handicap, in order to support rich interactions, website designers have traditionally anticipated every type of interaction sequence beforehand and hardwired multiple (customized) browsing interfaces (or algorithms) to cover all possibilities (e.g., see Fig. 2.1).

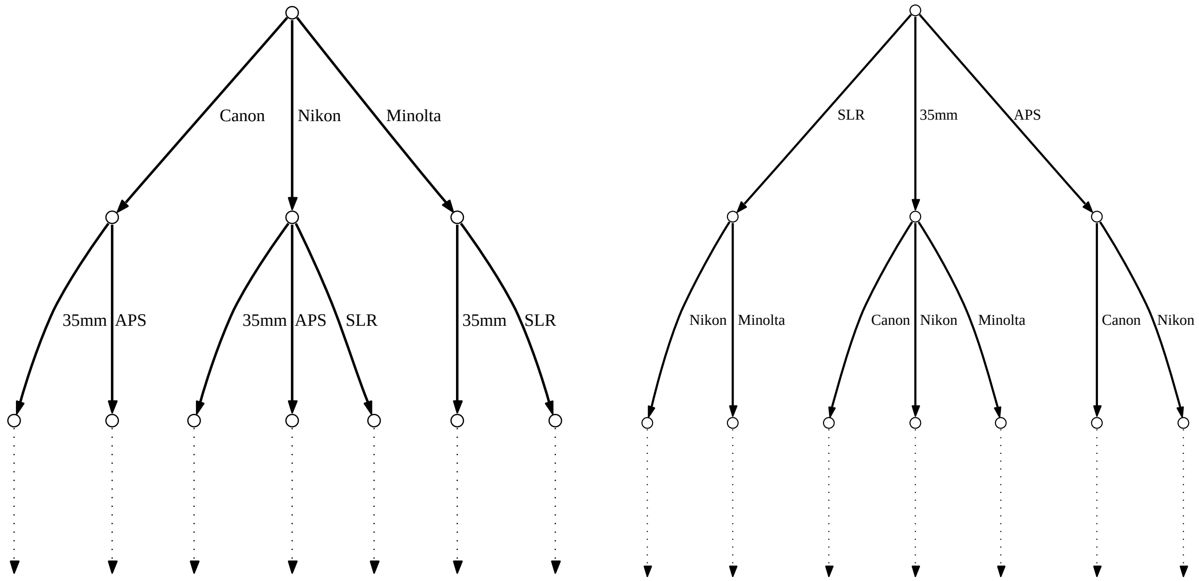


Figure 2.1: Two organizations of a camera catalog: by maker–type (left) and by type–maker (right). Only two levels are shown for ease of illustration. The vertices are webpages, the edges denote hyperlinks, and labels on edges represent the text anchoring the hyperlinks or selections made by a navigator.

This usually implies creating and storing separate information hierarchies. Sometimes, the site designer chooses an intermediate solution which places a prior constraint on the types and forms of interaction sequences supported. This is frequently implemented by directing the user to one of several predefined categories (e.g., 'to search by Lens, click here.').

For example, a website organization such as that shown in Fig. 2.1 (left) would be appropriate for the first consumer, who thinks of cameras primarily by their maker and only secondarily by lens type. Conversely Fig. 2.1 (right) would be appropriate for the second consumer, who thinks of cameras by lens type first. Many websites are indeed organized along such multiple facets and shift the responsibility to the user, who must employ the right interface for his information-seeking activity.

Such designs present several problems. For independent levels of classification, such as in Fig. 2.1, there exists a combinatorial explosion of scenario possibilities. If cameras are distinguished by, say, six independent attributes of classification, then we have to support $6 \times 5 \times 4 \times 3 \times 2 = 720$ possible browsing organizations! Some websites, such as Epicurious.com, a site which organizes recipes, described in [Hea00], actually take such an exhaustive approach and support all possible ways of interacting with it. Such a solution also leads to cumbersome site designs. In an attempt to customize information access, these solutions exacerbate information overload and thus run contrary to the goals of personalization. The more fundamental problem with such designs is that they over-specify the personalization goals by anticipating in advance all the forms of interactions that the site must support.

Websites are not traditionally designed for mixed-initiative interaction; this is because historically web interactions, in general, started out as simple means for retrieving pages from a server. Moreover, the underlying HTTP access protocol is *stateless* because it does not

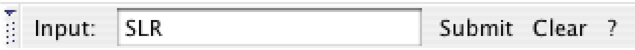


Figure 2.2: The Extempore out-of-turn interaction toolbar interface for personalized interaction with websites. I embed this interface into extant web browsers to augment hyperlink interaction. At any point in the interaction, the user has the option of supplying personalization parameters (in the toolbar's textfield) and conducting personalization. Here, the user has supplied SLR presumably out-of-turn.

retain information about the current user interactions for future use. Because of this traditional usage paradigm and statelessness, few interaction interfaces exist for mixed-initiative interaction on the web. Arguably the only interaction interface that allows the user to take the initiative is the 'Location URL' box in web browsers – the user can choose to discard the current site and enter a different site's URL to browse. This form of mixing initiative is restrictive and does not let users take the initiative *within* the current website.

2.3 Solution Approach: Out-of-turn Interaction

Out-of-turn interaction is my solution to support the user in taking the initiative in web interactions. It is a technique which empowers the user to take the initiative in a user–website dialog by providing unsolicited, but relevant, information out-of-turn.

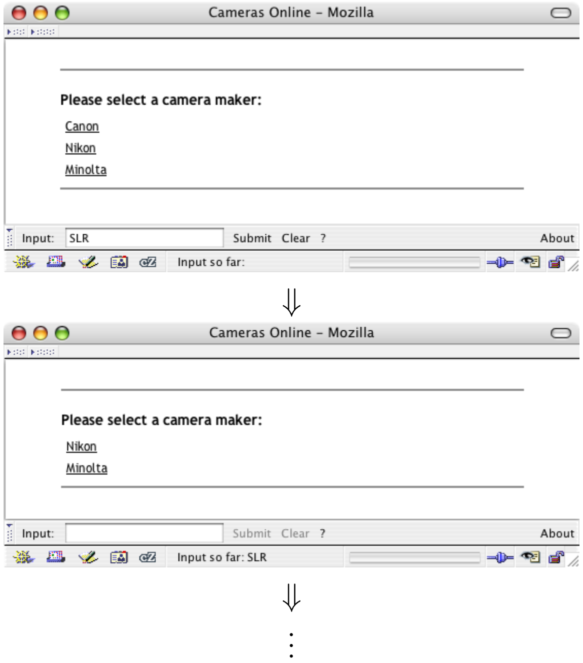
2.3.1 Extempore

To realize out-of-turn interaction, I developed a toolbar, which I call *Extempore* (see Fig. 2.2), to capture and communicate the supplied out-of-turn information to the underlying system. I implemented Extempore using XUL (XML User-interface Language) [BSD01], a cross-platform language for describing user interfaces of applications, as a plug-in to the Netscape/ Mozilla web browser. Extempore can be implemented for other traditional web browsers, such as Internet Explorer, as well. However, IE currently does not support XUL.

An out-of-turn enabled website need hardwire only *one* design, but because a user, armed with this toolbar interface, can take the initiative, the site can support any mixed-initiative interaction. For the site designer, the advantage is that he is no longer plagued by the thought of having to support all possible interfaces directly in the hyperlink structure. For the website user, the interface appears less cluttered and the interaction resembles more of a real dialog, with all its attendant advantages.

2.3.2 Camera Dialog Revisited

Fig. 2.3 illustrates how out-of-turn interaction works. Fig. 2.3 (top) illustrates the top level of Fig. 2.1 (left) at the outset, which shows the three choices available for camera maker. This site trivially supports the first consumer, since he can proceed to click on 'Nikon' first and then specify that he is interested in an SLR camera. This amounts to simple browsing or *in-turn interaction* (and I call 'Nikon' an *in-turn* input). Because the second user does not wish to specify a maker at the outset, he uses Extempore to supply 'SLR' out-of-turn. The



Cameras Online – Mozilla

Please select a camera maker:

Canon
Nikon
Minolta

Input: SLR

Input so far:

Cameras Online – Mozilla

Please select a camera maker:

Nikon
Minolta

Input:

Input so far: SLR

Figure 2.3: Example of a personalized web interaction for a user with a hypothetical camera catalog. At the beginning of the interaction, the user decides to not pursue any of the presented hyperlinks for camera maker. Instead, he uses the Extempore interaction toolbar to specify his choice of lens type out-of-turn (left). The results of processing this input cause the Canon option to be removed from the list of available makers (right) because Canon does not offer the specified type of lens. Once again, the user opts to employ Extempore to provide warranty information out-of-turn (results not shown).

next stage of the dialog (Fig. 2.3, bottom) shows that this input has been taken into account by revising the list of makers. Thus, out-of-turn interaction via Extempore now makes one site support multiple modes of interaction. Similarly, I could have used Fig. 2.1 (right) to support both users. Out-of-turn interaction makes an enumerative classification [AKB91] appear to be an unenumerative faceted classification [Wyn00].

2.3.3 What does it mean to interact out-of-turn?

One interpretation of out-of-turn interaction is that, when the user enters 'SLR', she is desiring to experience an interaction sequence through the site involving 'SLR.' Later in this thesis (Chapter 4), I develop and formalize multiple interpretations of out-of-turn interaction. The implicit assumption in the current implementation is that what is entered into Extempore is a hyperlink label (or variation thereof) nested deeper in the site, and hence an *in-vocabulary* utterance. In other implementations, a more elaborate modeling of the vocabulary could be conducted. Therefore, out-of-turn interaction is merely a mechanism to address alternate aspects of the given activity, while postponing the specification of currently solicited aspects.

2.3.4 Why would users interact out-of-turn?

There are several reasons why users might desire to interact out-of-turn. For instance, what the site is requesting from the user may actually be what the user is seeking in the first place! In Figure 2.3 (left), the site is soliciting camera maker, but the user might be looking for makers with a certain property (those that offer SLR cameras). Being able to supply information out-of-turn in an otherwise hardwired site permits the user to experience interaction sequences not describable by browsing. This means that the site can support all permutations of specifying camera attributes, without explicitly enumerating all in-turn choices. If the specification of a camera involves n independent attributes of classification, then we would require $n!$ faceted browsing classifications to support all tasks (i.e., browse by maker–lens–. . . , by lens–maker–. . . , and so on). Incorporating out-of-turn information does not curb the interaction, i.e., the hierarchical organization is preserved, and situates future interactions in the context of past ones.

More fundamentally, out-of-turn interaction is a novel technique for flexibly bridging any mismatch between the user's model of information seeking and the site's hardwired hyperlink structure. The site's layout and design influences how a user interacts with it. However, a user's mental model indicates how her information-seeking goals are best specified and realized. Notice that out-of-turn interaction reconciles this mismatch in a manner fundamentally different from the existing state of the art solutions, such as multiple faceted browsing classifications and search tools. It is different than the interaction websites employing multiple faceted browsing classifications (Epicurious.com) afford because it does not enumerate all possibilities in the hyperlink structure. Extempore is not a site-specific (or general) search tool that returns a flat list of results, akin to the power search functionality provided at Amazon.com or the Google toolbar. It is important to note that Extempore is embedded in the web browser, and not the site's webpages. This design property thus makes the interface and resulting out-of-turn interaction unintrusive and optional. In addition, these features of

```
if (Cannon)
    if (35mm)
      ...
    else if (APS)
      ...
else if (Nikon)                          if (Nikon)
    if (35mm)       if (35mm)                  ...
      ...             ...
    else if (APS)   else if (APS)
        ...             ...
    else if (SLR)   else if (SLR)
        ...             ...
else if (Minolta)                        else if (Minolta)
    if (35mm)                                  ...
      ...
    else if (SLR)
        ...
```

Figure 2.4: Modeling website interactions in a program. Original website (left) can display differently as the result of browsing for 'Nikon' (center) or supplying 'SLR' out-of-turn (right).

Extempore permit the user to take the initiative at multiple points in a browsing session, at his, rather than the site's, discretion. Extempore is thus an interface which users can independently bring to bear upon their browsing experience at multiple points and multiple sites. Further, while search engines index webpages, Extempore relies on an internal, explicit representation of the website and, when the user supplies out-of-turn input, uses transformation techniques to stage the interaction, pruning the website accordingly. In summary, out-of-turn interaction is fundamentally different because it does not anticipate, via enumerated hyperlinks or any other ad-hoc mechanism, when the mismatch might happen.

2.4 Program Representation and Transformation: a Model for Out-of-turn Interaction

How exactly does out-of-turn interaction via Extempore work? It is helpful to think of information as being organized along a motif of interaction sequences. With this thought I can model interaction with a website as a sequence of conditionals to be evaluated, as shown in Fig. 2.4 (left). Imagine these conditionals written in any programming language, such as C. Notice that the nested structure of the program models the hierarchical hyperlinked structure of the site. The hyperlink labels are represented as program variables and semantic dependencies between links are captured by the mutually-exclusive `if..else` dichotomies. The program's control-flow, as written, models the browsing interaction within the site which is, in this case, progressively drilling-down the hierarchy by making individual selections.

For the user who clicks on 'Nikon' (and, hence, responds to the initiative), Fig. 2.4 (center)

```
int pow (int base, int exponent) {
  int product = 1;
  for (int i = 0; i < exponent; i++)
    product = product * base;
  return product;
}
```

```
int pow2 (int base) {
  return (base * base);
}
```

Figure 2.5: Illustration of the partial evaluation technique. A general purpose `pow`er function written in C (left) and its specialized version with `exponent` statically set to 2 to handle squares (right). Such specializations are performed automatically by partial evaluators such as C-Mix [JGS93c].

models what I want to happen. That is, the choices now reflect the three types of cameras under Nikon: 35mm, APS (Advanced Photography System), and SLR. On the other hand, Fig. 2.4 (right) models what the user who enters 'SLR' out-of-turn wants to happen. That is, the choices should reflect the choices of makers, and only two, Nikon and Minolta, should be available. We can think of Fig. 2.4 as capturing requirements for program transformations. *Interestingly, the same program transformation algorithm can support both browsing and out-of-turn interaction!*

This transformation algorithm is called *partial evaluation* [Jon96]. Partial evaluation is a program specialization technique, familiar to compiler writers, which simplifies portions of programs given some (but not all) of their input. The input to a partial evaluator is a program and a partial (static) assignment of values to its variables. Its output is a specialized version of this program (typically in the same language), which uses the assignments to 'pre-compile' as many operations as possible. A simple example is how we can specialize the C function `pow`er to create a new function, say `pow2`, which computes the square of an integer. Consider the definition of `pow`er shown in Fig. 2.5 (left). If we knew that a particular user will utilize it only for computing squares of integers, we could specialize it (for that user) to produce the `pow2` function (right). Note that `pow2` is obtained *automatically* (not by a human programmer) from `pow` by pre-computing all expressions which involve `exponent`, unfolding the for-loop, and by various other compiler transformations such as *copy propagation* and *forward substitution*. It is automatic, in that there exist off-the-shelf partial evaluators that take programs such as that shown in Fig. 2.5 (left) and a partial assignment of its variables as input and produce programs such as those illustrated in Fig. 2.5 (right) as output. Dozens of partial evaluators are available for specializing programs written in languages such as C, FORTRAN, Scheme, Haskell, Java, PROLOG, and several other important languages. I refer the interested reader to [Jon96] for an introduction to partial evaluation.

As a concept in computing, partial evaluation is at least 30 years old. While the traditional motivation for partial evaluation is to achieve speedup or remove interpretation overhead [Jon96] in highly parameterized environments, it also can be viewed as a technique for simplifying program presentation, by removing inapplicable, unnecessary, and uninteresting information (based on user criteria) from a program. In this vein, the approach shown here helps relate it to information personalization. In particular, I use partial evaluation to *non-sequentially* evaluate expressions nested deeper in the program without evaluating their lexical predecessors. Fig. 2.4 (center) is then the result of partially evaluating the input program (Fig. 2.4, left) w.r.t. the variable '`Nikon = 1`' (and '`Cannon = 0`' and '`Minolta = 0`'). Similarly, Fig. 2.4 (right) results from partially evaluating Fig. 2.4 (left) w.r.t. '`SLR = 1`' (and '`35mm = 0`' and '`APS = 0`'). Since partial evaluation subsumes interpretation, I also can use it to *sequentially evaluate* the program given an in-turn input. Hence, I use the same program transformation to support both interaction techniques in a single representation. This is beneficial because the representation my approach affords (notice the nesting of conditionals in Fig. 2.4, left) is typically much smaller than expressing the same as a union of all possible interaction sequences.

Since partially evaluating a program results in another program, the transformation operator is *closed*. For interaction, this means that any modes of information seeking (such as browsing, in Fig. 2.4) originally modeled in the program are preserved. In the above example, personalizing a browsable hierarchy returns another browsable hierarchy. The closure property also means that the original information-seeking activity (browsing) and personalization can be interleaved in any order. Evaluating the program in the form and order in which it was modeled (by using partial evaluation to sequentially evaluate it) amounts to the system-initiated mode of browsing. 'Jumping ahead' to nested program segments (by using partial evaluation to non-sequentially evaluate the program) amounts to the user-directed mode of personalization. We can render and browse, in the traditional sense, the simplified programs in Fig. 2.4 (center and right), or partially evaluate further with additional user inputs. My use of partial evaluation is thus central to realizing a mixed-initiative mode of information seeking [RCPQ02], without explicitly hardwiring all possible interaction sequences. With this approach, it also is possible to encode miscellaneous application logic (about the interaction) and use it to drive the personalization.

To summarize, I have reduced the problem of personalizing websites to partially evaluating a representation of interaction [Ram00]. My approach to mixed-initiative interaction is to create a representation of the space of possible interaction sequences, and then to use the technique of partial evaluation to realize individual (in-turn or out-of-turn) interaction sequences. I model the information space as a program, partially evaluate it w.r.t. a partial assignment of its variables representing user input, and recreate the personalized information space from the specialized program. In addition, by working programmatic representations, I can employ alternate program transformations, such as dead-code detection and elimination [CXY01, WZ91], to, for example, remove the `Canon` conditional after partially evaluating w.r.t. `SLR = 1` (and `35mm = 0` and `APS = 0`). Although I can use any partial evaluation software to implement a web personalization engine, I present an approach here using the transformation capabilities of XSLT (eXtensible Stylesheet Language for Transformations).

2.5 Using XSLT for Personalization

XSLT is a mature technology for transforming XML documents (to XML). It can implement many transformations [Tid01], but I specifically use it here to support the partial evaluation transformation. An XSLT transformation is specified as a set of pattern-action rules in a stylesheet, which are then recursively applied, starting from the root of a tree-structured XML document. Whenever a match is encountered, the stylesheet describes the particular actions to take.

```
<site>
  <Cannon>
    <35mm>
      ...
    </35mm>
    <APS>
      ...
    </APS>
  </Cannon>
  <Nikon>
    ...
  </Nikon>
  <Minolta>
    ...
  </Minolta>
</site>
```

Figure 2.6: Modeling website interactions in an XML format.

```
<?xml version="1.0"?>

<!-- Template for the input:  SLR -->
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:output method="xml"/>

<xsl:template match="SLR">
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="APS"/>

<xsl:template match="35mm"/>

<!-- matches any node, including the root -->
<xsl:template match="*|@*">
  <xsl:copy>
    <xsl:apply-templates select="@*|node()"/>
  </xsl:copy>
</xsl:template>

</xsl:stylesheet>
```

Figure 2.7: Stylesheet generated for a user interested in SLR cameras (specified by XML element `SLR`). Running this stylesheet through an XSLT processor with the XML document shown in Fig. 2.6 emulates partial evaluation for personalization and transforms the document to reflect the site shown in Fig. 2.4 (right).

```
<?xml version="1.0"?>

<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:output method="xml"/>

<xsl:strip-space elements="*"/>

<xsl:template match="@* | *[child::node()]">
  <!-- prunes dead-end nodes, -->
  <!-- only retains nodes that lead to at least one text node -->
  <xsl:if test="descendant::text()">
    <xsl:copy>
      <xsl:copy-of select="@*"/>
        <xsl:apply-templates select="@* | node()"/>
    </xsl:copy>
  </xsl:if>
</xsl:template>

</xsl:stylesheet>
```

Figure 2.8: Stylesheet for pruning dead-ends. I use this stylesheet in conjunction with other stylesheet transformations to post-process a resulting specialized XML document.

To use XSLT for personalization, I must first represent programs such as that in Fig. 2.4 (left) in XML. Fig. 2.6 shows one possible approach to modeling the cameras website. Then, for each user input, I prepare a suitable XSLT stylesheet, outlining the transformation, and apply it on the XML source. For instance, the stylesheet of Fig. 2.7 would be appropriate for the user interested in 'SLR' cameras. This stylesheet specifies that the 'SLR' hyperlink can be simplified and the '35mm' and 'APS' hyperlinks can be removed (because they are mutually exclusive with respect to 'SLR'). I then apply additional post-processing transformations to prune dead ends. For example, Canon has no SLR models so the transformation can remove it. Fig. 2.8 shows an XSLT stylesheet for pruning dead ends.

XSLT also facilitates other post-processing activities, many aesthetic and originally handled by ad-hoc mechanisms such as shell scripts. For example, typical partial evaluators will rename variables in a specialized program. Although such conventions do not affect the semantics of the resulting program, they do shatter the original symmetry between hyperlink labels and program variables [Van01]. XSLT obviates the need to reconcile such differences because it does not rename XML elements unless told to do so. In addition, using reflective capabilities to extract element names, I can easily implement an 'Input so far: ...' label (see Fig. 2.3) at every stage of the interaction, to help orient users. Lastly, high-level XSLT functions, such as sorting, simplify tasks such as link label ordering on re-created webpages.

The transformation can be implemented as part of a proxy server that is designed to process input events communicated from the browser (either clicking on hyperlinks or specifying inputs out-of-turn). Recall that one must convert the transformed XML document back into a website for a user to browse. The user might then proceed to click on any remaining hyperlinks or might require a further degree of personalization, both of which another XSLT transformation can handle.

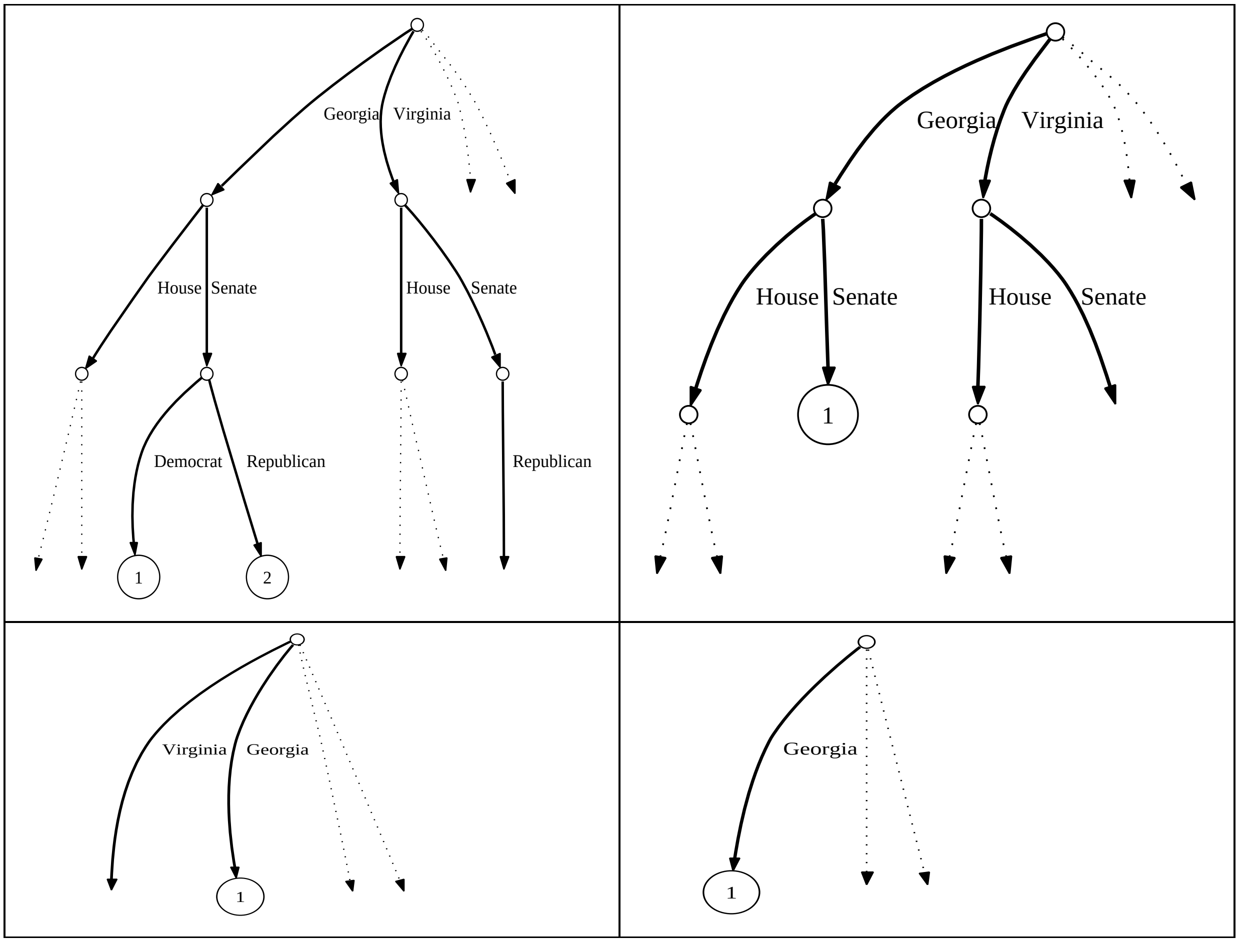
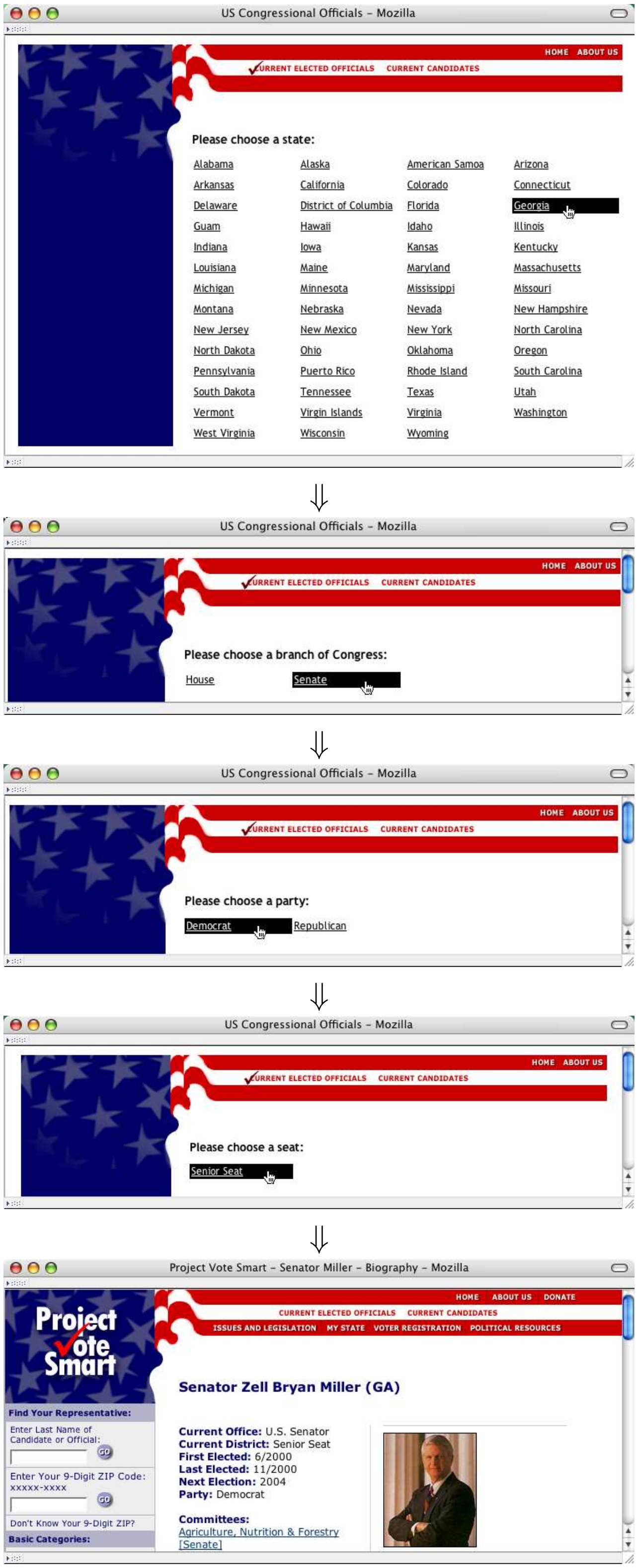


Figure 2.9: Personalizing a browsing hierarchy w.r.t. ‘Democratic Senators.’ (top left) Original schematic of the Project Vote Smart Congressional website, depicting information about members of the US Congress (only the first three levels are shown for ease of presentation). (top right) Personalized hierarchy w.r.t. the criterion ‘Democrat.’ Notice that not only the pages, but also their structure is customized for (further browsing by) the user. (bottom left) The hierarchy in the top right personalized w.r.t. ‘Senate.’ Notice that the hyperlink labeled ‘Virginia’ is a dead-end. This is because Virginia has only Republican Senators. (bottom right) A post-processing step removes all dead-ends.


US Congressional Officials – Mozilla
HOME
ABOUT US
CURRENT ELECTED OFFICIALS
CURRENT CANDIDATES

Please choose a state:

Please choose a branch of Congress:

House
Senate
⇓
US Congressional Officials – Mozilla
HOME
ABOUT US
CURRENT ELECTED OFFICIALS
CURRENT CANDIDATES

Please choose a party:

Democrat
Republican
⇓
US Congressional Officials – Mozilla
HOME
ABOUT US
CURRENT ELECTED OFFICIALS
CURRENT CANDIDATES

Please choose a seat:

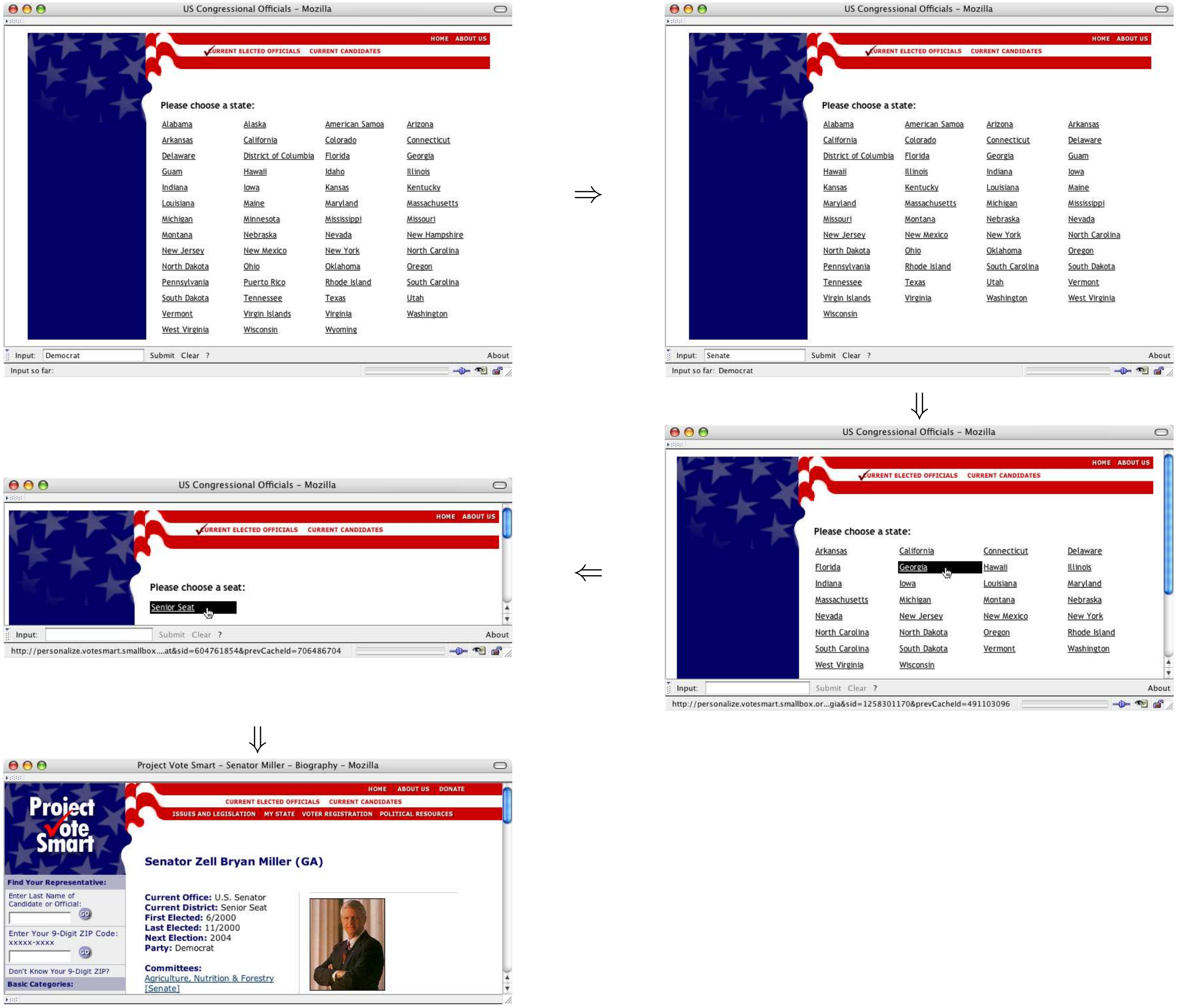
Senior Seat
⇓
Project Vote Smart – Senator Miller – Biography – Mozilla
Project Vote Smart
HOME
ABOUT US
DONATE
CURRENT ELECTED OFFICIALS
CURRENT CANDIDATES
ISSUES AND LEGISLATION
MY STATE
VOTER REGISTRATION
POLITICAL RESOURCES
Find Your Representative:
Enter Last Name of Candidate or Official:
GO
Enter Your 9-Digit ZIP Code: xxxxx-xxxx
GO
Don't Know Your 9-Digit ZIP?
Basic Categories:

Senator Zell Bryan Miller (GA)

Current Office: U.S. Senator
Current District: Senior Seat
First Elected: 6/2000
Last Elected: 11/2000
Next Election: 2004
Party: Democrat

Committees:
Agriculture, Nutrition & Forestry [Senate]

Figure 2.10: An in-turn interaction experience with Project Vote Smart.



US Congressional Officials – Mozilla
HOME ABOUT US
CURRENT ELECTED OFFICIALS CURRENT CANDIDATES
Please choose a state:
Alabama Alaska American Samoa Arizona
Arkansas California Colorado Connecticut
Delaware District of Columbia Florida Georgia
Guam Hawaii Idaho Illinois
Indiana Iowa Kansas Kentucky
Louisiana Maine Maryland Massachusetts
Michigan Minnesota Mississippi Missouri
Montana Nebraska Nevada New Hampshire
New Jersey New Mexico New York North Carolina
North Dakota Ohio Oklahoma Oregon
Pennsylvania Puerto Rico Rhode Island South Carolina
South Dakota Tennessee Texas Utah
Vermont Virgin Islands Virginia Washington
West Virginia Wisconsin Wyoming
Submit Clear ? About
⇒
⇓
⇐
⇓
Please choose a seat:
Senior Seat
Project Vote Smart – Senator Miller – Biography – Mozilla
HOME ABOUT US DONATE
ISSUES AND LEGISLATION MY STATE VOTER REGISTRATION POLITICAL RESOURCES
Project Vote Smart
Find Your Representative:
Enter Last Name of Candidate or Official:
Enter Your 9-Digit ZIP Code: XXXXX-XXXX
Don't Know Your 9-Digit ZIP?
Basic Categories:

Input: Democrat
Input so far:

Input: Senate
Input so far: Democrat

http://personalize.votesmart.smallbox.or...gia&sid=1258301170&prevCacheId=491103096

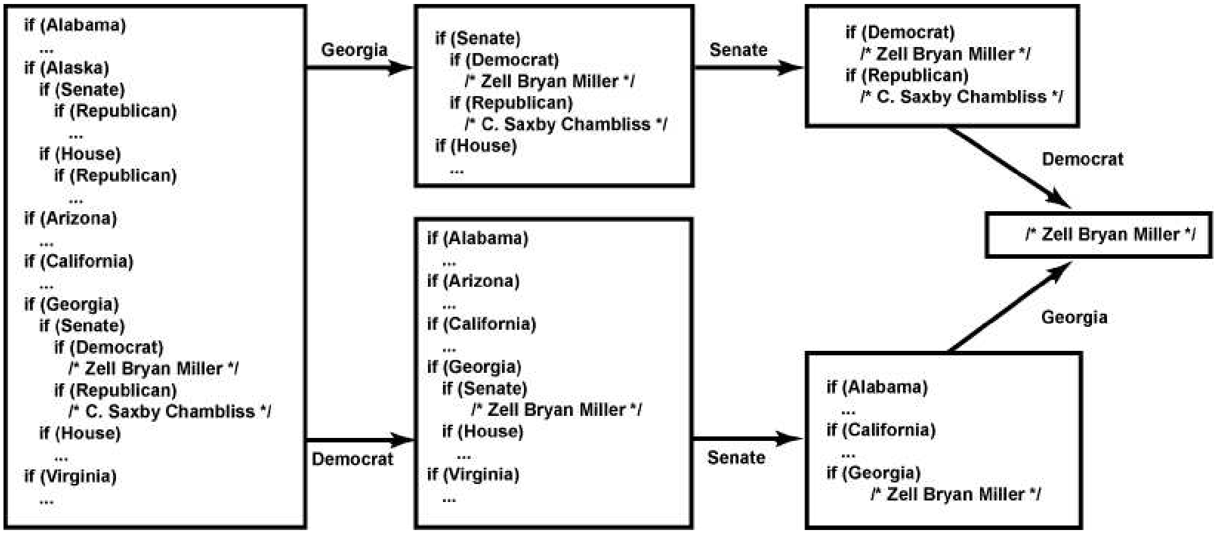
http://personalize.votesmart.smallbox....at&sid=604761854&prevCacheId=706486704

Senator Zell Bryan Miller (GA)

Current Office: U.S. Senator
Current District: Senior Seat
First Elected: 6/2000
Last Elected: 11/2000
Next Election: 2004
Party: Democrat

Committees:
Agriculture, Nutrition & Forestry
[Senate]

Figure 2.11: An mixed-initiative interaction experience with Project Vote Smart.



```
if (Alabama)
  ...
if (Alaska)
  if (Senate)
    if (Republican)
      ...
  if (House)
    if (Republican)
      ...
if (Arizona)
  ...
if (California)
  ...
if (Georgia)
  if (Senate)
    if (Democrat)
      /* Zell Bryan Miller */
    if (Republican)
      /* C. Saxby Chambliss */
  if (House)
    ...
if (Virginia)
  ...
```

```
if (Senate)
  if (Democrat)
    /* Zell Bryan Miller */
  if (Republican)
    /* C. Saxby Chambliss */
if (House)
  ...
```

```
if (Democrat)
  /* Zell Bryan Miller */
if (Republican)
  /* C. Saxby Chambliss */
```

```
/* Zell Bryan Miller */
```

```
if (Alabama)
  ...
if (Arizona)
  ...
if (California)
  ...
if (Georgia)
  if (Senate)
      /* Zell Bryan Miller */
  if (House)
    ...
if (Virginia)
  ...
```

```
if (Alabama)
  ...
if (California)
  ...
if (Georgia)
      /* Zell Bryan Miller */
```

Figure 2.12: Staging dialogs using partial evaluation. The top series of transformations mimic an in-turn dialog with the user specifying (Georgia: Senate: Democrat), in that order. The bottom series of transformations correspond to a mixed-initiative dialog where the user specifies (Democrat: Senator: Georgia), in that order.

2.6 Prototype Implementation

To demonstrate these ideas, I designed a prototypical personalization system for the US Congressional portion of the Project Vote Smart website (vote-smart.org). The site is organized in a hierarchical manner, where it requires the user to progressively make choices for politician attributes—*state* at the first level, *branch of Congress* at the second level, followed by levels for *party*, and *district/seat*—by browsing (see Fig. 2.9, top left). I extracted the site's hierarchy for presenting pages using a depth-first crawl of the site. A depth-first crawl is merely a way to capture a site's browsing hierarchy. While I represented the extracted hierarchy in an XML notation, I use the programmatic rendition of the site for presentation to reinforce my programmatic model of personalization. Thus, I abstract the situation in Fig. 2.9 (top left) by the program of Fig. 2.12 (left). Figs. 2.10 and 2.11 show how you can solve the following two tasks:

1. Find the webpage of the Democratic politician from Georgia holding the Senior Senatorial seat.
2. Find the webpage of each Democratic Senator.

A demo of this system is available at http://pipe.cs.vt.edu.

Fig. 2.12 illustrates how the *same* programmatic rendition of the PVS website is transformed by the *same* transformation technique, partial evaluation, to stage the in-turn interaction sequence in Fig. 2.10 (top series of transformations) *as well as* the mixed-initiative interaction sequence in Fig. 2.11 (bottom series of transformations) in a single framework.

I also am exploring the design of multimodal web interfaces to communicate partial information where, for instance, the user interacts using both the traditional (textual) interaction interfaces (e.g., hyperlinks) and via voice. A voice interface more naturally resembles a dialog where the user 'takes the initiative' by speaking to the browser.

In this chapter I provided the reader with an overview of the program transformation approach to information personalization. The concepts I introduced and discussed here serve as a foundation for the remainder of this document. Armed with the foundation provided by this chapter, we are now ready to see how the program transformation approach is situated in the landscape of personalization research, discussed in the following survey of related research. To reinforce that out-of-turn information can be captured and communicated using multiple modalities, I demonstrate the use of a voice interface to supply out-of-turn inputs in the survey. My survey also addresses a wide range of support tools and technologies for use at various points in the lifecycle of a personalization system, from requirements gathering to implementation (e.g., the transformation support provided by XSLT presented here).

Chapter 3

Survey of Related Research

> "So, what is personalization?"
>
> D. Riecken, in guest editor's introduction to the
> August 2000 *Communications of the ACM*
> issue on personalization [Rie00b]

This chapter surveys personalization research. It covers mechanisms for information-finding on the web, advanced information retrieval systems, dialog-based applications, and mobile access paradigms. Specific emphasis is placed on studying how users interact with an information system and how the system can encourage and foster interaction. This helps bring out the role of the personalization system as a facilitator which reconciles the user's mental model [Bor86] with the underlying information system's organization. Three tiers of personalization systems are presented, paying careful attention to interaction considerations. These tiers show how progressive levels of sophistication in interaction can be achieved. The chapter also surveys systems support technologies and niche application domains.

3.1 Background

While the roots of personalization can be traced back to information filtering [BC92] and recommender systems [RV97], the web has propelled personalization into a highly studied and legitimate research area. The explosion of online content and the advent of ubiquitous computing devices and information appliances [Ber00] have made personalization critical to the success of Internet applications. Personalization is achieved in information systems which afford complex, compelling, and user-adapted interactions. Studying how users interact with information systems and understanding the frustrations they experience provides ample motivation for personalization.

3.1.1 Why Personalize?

Let us begin with the quintessential information access paradigm on the web – browsing [Mar97]. Bush is regarded as the godfather of browsing as known today [Bus45]. In browsing, two distinct roles are seen.

> 'The role of the author was to create the hypertext and the role of the user/reader was to browse through it. Thus, the reader was faced with the task of understanding the author's mental model [Bor86] of the hypertext documents in order to navigate the collection of linked nodes (hyperbase) effectively' [BC99].

Pre-defined, hardwired browsing interfaces in information systems have been succinctly characterized with phrases such as 'static hypertext' [BC99], 'strong authoring' [BC99], and 'one-size-fits-all' [Bru01, Chi97]. Such a rigid model assumes that the author's viewpoint is correct. The resulting mental mismatch problem has been identified as a legitimate research issue in [Bor86, Suc87]. The goal of personalization technologies is to help overcome this mismatch. Essentially the same issue arises in other information access paradigms, and a variety of delivery mechanisms.

3.1.2 Approaches to Studying Personalization

Operationally, the word 'personalization' is broad and open to many interpretations. Many surveys of personalization focus on technical distinctions of how information is tailored to end-users and the level at which it is targeted. Business schools have adopted terms such as 'real time,' 'one-to-one,' and 'check-box' personalization. Therefore, there are 'personalized views of personalization' [Rie00b]. For instance, the articles in the August 2000 *Communications of the ACM* issue on personalization range from topics such as natural language dialogs, to website restructuring, to manually customizable portals.

In this survey, I approach personalization from the viewpoint of personalizing *interaction*. Interaction with an information system is thus the common thread among all systems surveyed in this chapter. Distinctions are only made when they reveal differences among interaction paradigms. For instance, Amazon's recommender system might make better recommendations of books than another bookseller's but if they afford the user the same interaction, they are considered equivalent for my purposes. In fact, many personalization solutions do not even explicitly recognize the issue of interaction with a user; needless to say they are not surveyed here. Distinctions such as *content-based* and *collaborative*—popular in the recommender systems community—thus do not find place in this chapter. For a survey making these types of dichotomies, please see [PGF04].

I posit that surveying personalization according to interactions of users [Mar97] is a more holistic approach to studying this subject. To the best of my knowledge, this survey is the first to employ this approach. The reader should keep in mind that I use the term *personalization* synonymously with personalized interaction.

3.1.3 Organization of this Survey

Three main approaches to personalizing interaction are outlined (see Fig. 3.1). The first approach is the terminal case where the system provides no support for maintaining interaction and the onus of personalization is shifted to the user. As shown in Fig. 3.1(a), the system effectively behaves as a functional engine mapping users' specification aspects into results. It does not recognize the fact that information access occurs in the context of an interaction.

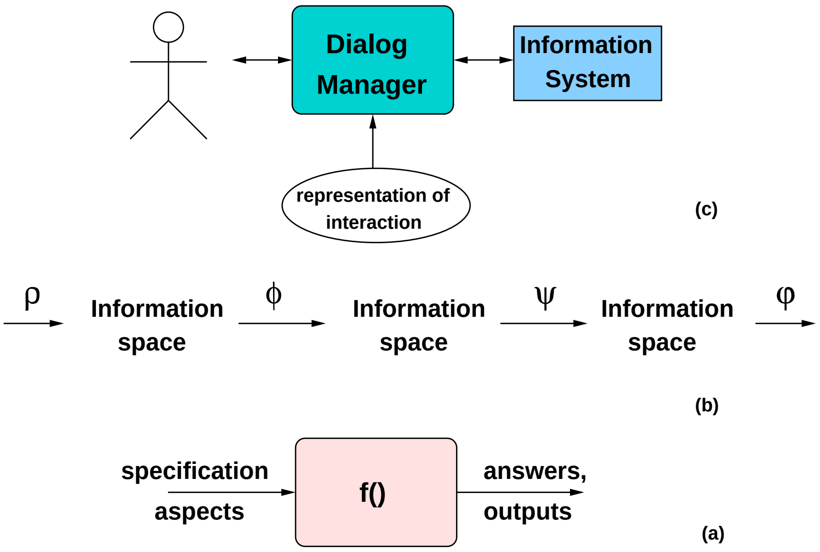


Figure 3.1: Three approaches to personalized interaction surveyed in this chapter: (a) templates for personalization, (b) operators for personalization, and (c) representing and reasoning about interaction.

Writing SQL queries in a database context is an example of a functional modeling. Although the user might interactively explore the database through a sequence of such queries, the system *per se* does not provide any support for interaction. These approaches, which I refer to as template-based, have a *functional-emphasis*. I survey a sample of systems in this tier as they relate to information access on the web. I pay particular attention to systems which combine distinct modalities of information seeking and which are especially relevant to the primary Internet access paradigms today.

Systems in the next tier provide a set of basic primitives for sustaining interaction. As shown in Fig. 3.1(b), these primitives are typically in the form of operators which successively transform an information space. The user is encouraged to apply these operators in a form which is suitable to his information-seeking activity. I say that such systems recognize and encourage interaction, but with an *operational-emphasis*.

The third tier of systems are novel in that they explicitly represent and capture interaction. As Fig. 3.1(c) shows, interaction here resembles more a dialog between the user and the information system. A natural dialog is one where both parties interact to achieve the desired information-seeking goals. Systems in this tier are characterized by their representations, whose expressiveness and capabilities directly relate to the quality of personalized interaction. They have a *representational-emphasis* and are most capable of reconciling the mental mismatch issue introduced earlier.

Notice that systems in a given tier can trivially support functionalities at lower tiers.

3.2 Templates for Personalized Interaction

It can be argued that being able to set the background color for a desktop screen is a rudimentary form of personalization. Here, the goals of personalization have become so

```
SELECT Name, Count
FROM States, WebCount
WHERE Name = T1
ORDER BY Count Desc
```

Figure 3.2: A WSQ query to rank states by how often they appear on the web. This query has traditional SQL semantics. States and WebCount are relations. The schema of States is States (Name, Population, Capital). The WebCount relation, whose schema is WebCount (SearchExp, T1, T2, . . . , TN, Count), is populated by the results of a web search request. T1, T2, . . . , TN are values for parameters in SearchExp. Notice that all aspects of information seeking necessary to determine an answer are provided in one stroke.

over-specified that the responsibility of achieving the personalization is shifted to the user, who must specify the settings.

A typical form of over-specification involves templates that are meant to be customized by the user. Another involves providing an expressive web query language, not unlike SQL. Their salient feature is a ‘one-shot’ [Bru01, MM00] style of personalization. This section surveys such approaches. Specifically, I start from a database perspective and describe the WSQ/DSQ project and probabilistic relational algebra. I next discuss web queries as templates and the use of templates for constructing personal information spaces.

3.2.1 WSQ/DSQ

The WSQ/DSQ (pronounced ‘wisk-disk’) project [GW00] at Stanford University attempts to bridge the gap between structured relational databases (DBs) and the unstructured web in support of an information retrieval request. WSQ (Web-Supported Queries) incorporates web search results into SQL queries over a database to enrich an answer. On the other hand, DSQ (Database-Supported (Web) Queries), its complement, leverages DB relations to enhance and explain web search results.

For the purposes of this chapter, it is sufficient to focus on the WSQ component. WSQ leverages web search results to provide a richer set of input parameters for a query against original relational data sources. In other words, the output of one query (the web search) is provided as input, along with the extant DB relations, to a master SQL query. In WSQ the primary mode of information seeking is thus an SQL query and the secondary mode is web search.

The basic idea behind WSQ is to permit users to make references to web search requests within a traditional SQL query. A user writes a query that makes reference to a web search engine (WSQ/DSQ uses AltaVista and Google) and search terms, obtains an answer – a new relation, and proceeds to the next independent query that may (e.g., join the resulting relation with itself) or may not involve the resulting relation. A typical WSQ query (from [GW00]) is shown in Fig. 3.2. Interaction in WSQ is hence limited to issuing a query and obtaining an answer. This is referred to as a one-shot interaction [Bru01] or a one-shot task [MM00]. Furthermore, traditional query processing cannot proceed until all attribute values initially populated with calls to a particular web search engine are replaced with corresponding URL answers.

Thus, interaction in WSQ is best modeled as a template for personalization. The order of the two information-seeking interactions in WSQ are determined *a priori* at query-creation time. This design of over-specification means that information-seeking parameters are provided in one stroke.

In fairness to the designers, the design in WSQ/DSQ is commensurate with the targeted applications (i.e., answering questions regarding comparisons or frequencies of items on the web, e.g., 'Rank all countries in North America by how often they are mentioned by name on the web.'). Nonetheless, WSQ is an interesting research project to study with respect to personalization and combining aspects of information seeking. I view it as a limiting case of a personalization system.

3.2.2 Probabilistic Relational Algebra

In [FR97], Fuhr and Rölleke approach the problem of integrating aspects of information seeking from a different angle. Specifically, the designers weave canonical IR parameters (e.g., weights, rankings, and probabilities) into a DBMS in order to enhance and improve retrieval.

Integration here is motivated by the lack of seamless methods to incorporate IR parameters into (relational) database management systems (DBMSs). For instance, DBMSs do not adequately address vagueness, imprecision, and uncertainty which IR systems are designed for. DBMSs are however strongly grounded in theory and relations afford expressive query languages (QLs). IR systems, on the other hand, incorporate parameters well but have problems incorporating ground facts. In addition, there is limited expression in IR QLs that is currently addressed with ad hoc methods.

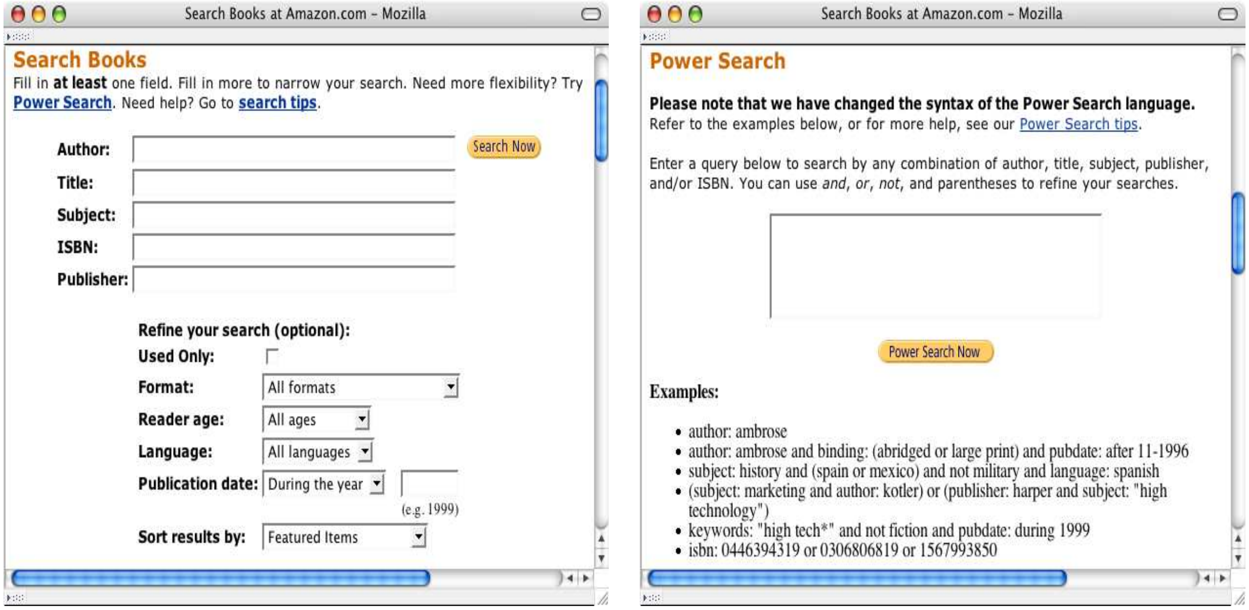
In order to weave standard IR parameters into a DBMS, Fuhr and Rölleke generalize traditional relational algebra, where probabilities are either 0 or 1, to the continuous range $[0 \ldots 1]$. Incorporating probabilities into tuples of DB relations is relatively straightforward. Ensuring that these probabilities are correctly propagated in an answer (after possibly complex joins or other operations) is difficult, due to uncertainty about the independence/dependence of tuples at query formulation time. Additional data could be modeled in relations to make such constraints explicit. With large DBs, however, such constraints and annotations embedded into relations may approach exponential levels.

Due to the explicit requirement to specify all information-seeking aspects at query formulation time, I classify Fuhr and Rölleke's work as a template for personalization. The system they developed does not allow users to specify parameters over time or leave parameters residual. Information-seeking sessions of this system resemble interaction with WSQ, with minor adaptations, e.g., instead of specifying which search engine to employ, the user indicates a probability threshold or an index weight.

3.2.3 Web Query Languages

Search Interfaces: Precursors to Web QLs

In order to combat cognitive frustrations experienced in browsing, many sites provide within-site search interfaces. I present three common user interface designs here. Fig. 3.3 (left) il-


Search Books at Amazon.com - Mozilla

Search Books

Fill in **at least** one field. Fill in more to narrow your search. Need more flexibility? Try Power Search. Need help? Go to search tips.

Author:
Title:
Subject:
ISBN:
Publisher:

Refine your search (optional):
Used Only: ☐
Format: All formats
Reader age: All ages
Language: All languages
Publication date: During the year (e.g. 1999)
Sort results by: Featured Items

Search Books at Amazon.com - Mozilla

Power Search

Please note that we have changed the syntax of the Power Search language.
Refer to the examples below, or for more help, see our Power Search tips.

Enter a query below to search by any combination of author, title, subject, publisher, and/or ISBN. You can use *and*, *or*, *not*, and parentheses to refine your searches.

Examples:

- author: ambrose
- author: ambrose and binding: (abridged or large print) and pubdate: after 11-1996
- subject: history and (spain or mexico) and not military and language: spanish
- (subject: marketing and author: kotler) or (publisher: harper and subject: "high technology")
- keywords: "high tech*" and not fiction and pubdate: during 1999
- isbn: 0446394319 or 0306806819 or 1567993850

Figure 3.3: (left) A book search interface at Amazon.com. This interface contains multiple category-labeled text-fields, expecting input to belong to a category. Such a design attempts to hide hyperlink enumeration in websites. (right) A power-search facility at Amazon.com that allows multiple query terms from different categories, but still requires categorical information.

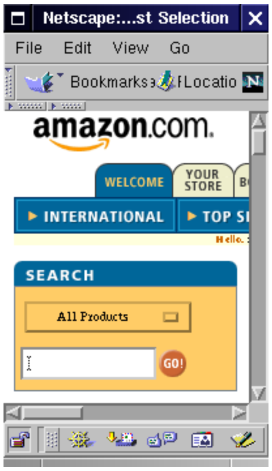


Figure 3.4: A search facility of Amazon.com that allows the entry of free-form text.

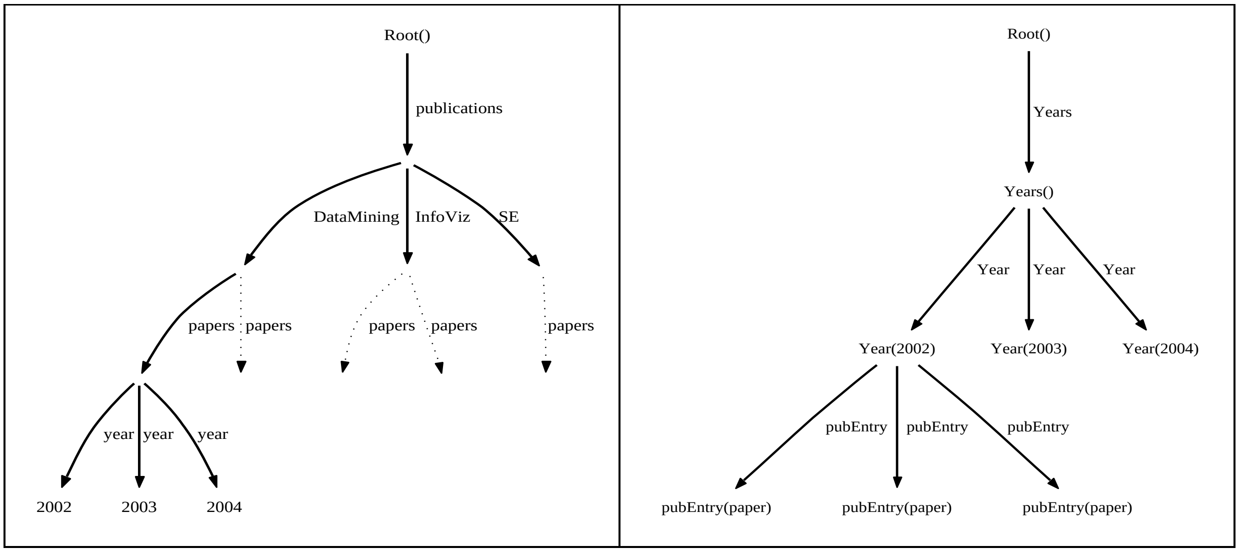


Figure 3.5: (left) Directed graph model of XML input to the StruQL query shown in Fig. 3.6. The publications are ordered by research area. (right) Directed graph model of XML output from the StruQL query shown in Fig. 3.6. Notice that publications are now ordered by year. Such XML data sources can be easily converted into a set of browsable webpages with tools such as XSL/XSLT [Cha99, Wid99].

lustrates part of a book search tool available at Amazon.com. This type of search interface is typical and requires a user to associate search terms with categorical information (e.g., author, title, and publisher). The goal of such search interfaces is to avoid enumerating multiple browsing paths to terminal information (in this case, a book webpage). An alternative design, shown in Fig. 3.3 (right), is called a 'power-search' and has gained popularity in many e-commerce sites. A power-search more closely resembles a web query language. In other words, such tools include a small language for communicating inputs involving multiple query fields (possibly combined via `AND`s or `OR`s). The power-search of Amazon.com shown in Fig. 3.3 (right) still requires a user to specify categorical information, however. From a user perspective, a less restrictive interface is a free-form text box (Fig. 3.4) that does not require categorical information. In such a design, users' query terms are matched against all attributes of an information nugget (e.g., webpage, book, or movie). The search facilities involved in the interfaces outlined above do not employ expressive QLs.

The systems presented below provide users with a sophisticated QL. Through the invocation of a query, such systems combine aspects of information seeking with reconstruction properties of an information space.

Restructuring Semistructured Data

The semistructured data [ABS00, FLM98] and XML communities have embraced the idea of building, restructuring, and managing information spaces (e.g., websites) with adaptations of traditional SELECT-FROM-WHERE queries. In this context, new and personalized information spaces may be constructed from DB relations, structured files, or semistructured data (e.g., XML). In addition, extant information spaces, such as websites, may be

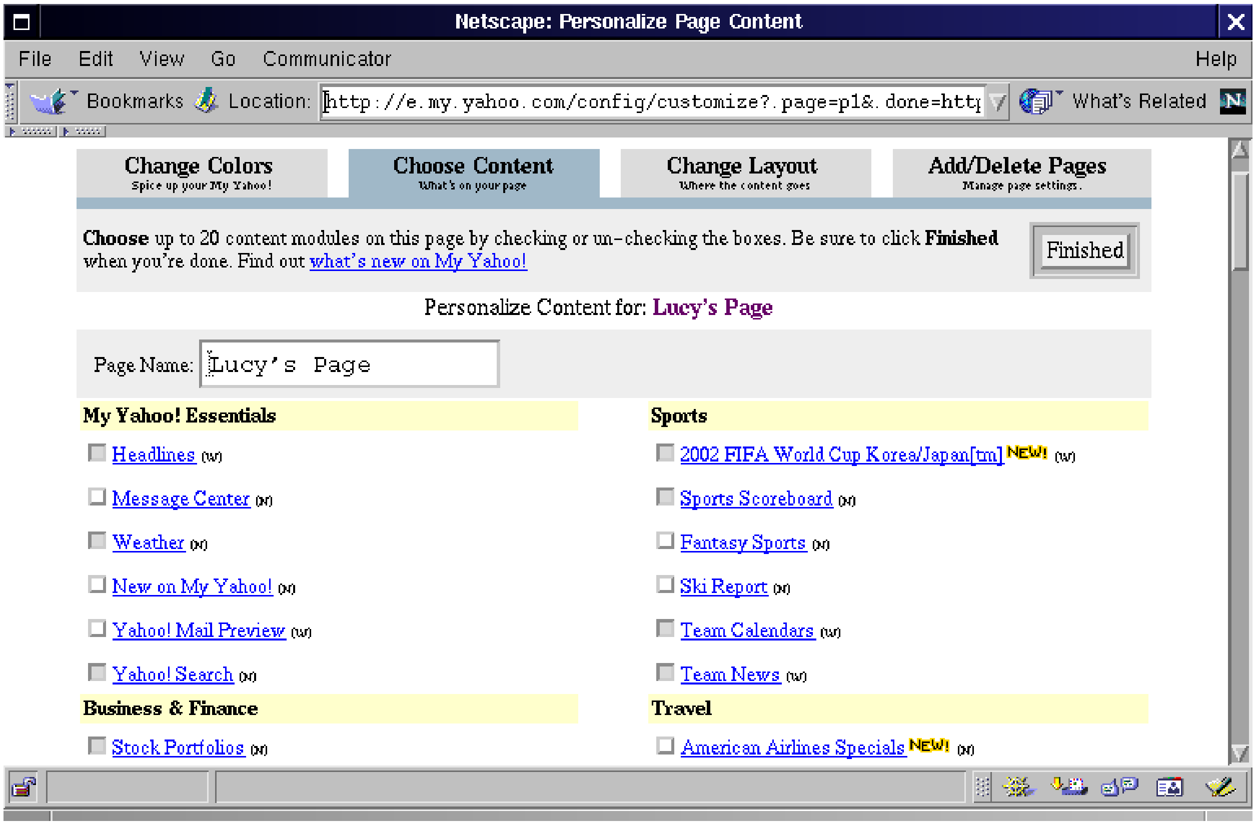
```
{ WHERE root in publications.xml,
        root -> "publications".
        ("DataMining" | "InfoViz" | "SE")."paper" -> paper,
        paper -> attribute -> attributeValue

  COLLECT Root(), pubEntry(paper)

  /* group by year */
  { WHERE attribute = "year",
          attributeValue -> "PCDATA" -> yearValue
    COLLECT Years(), Year(yearValue)
    LINK Root() -> "Years" -> Years(),
    Years() -> "Year" -> Year(yearValue),
    Year(yearValue) -> "pubEntry" -> pubEntry(paper),
    Year(yearValue) -> "year" -> yearValue
  }
}
```

Figure 3.6: A StruQL query. Notice that enough parameters have been specified in order to produce a reconstructed answer.

restructured via a declarative query. This latter application is more interesting to study for my purposes.

For example, consider a researcher who disseminates his publications on his webpage via a research area browsing dichotomy. The original data may be stored in XML files (see left side of Fig. 3.5). If this researcher desires to restructure the hierarchical presentation with respect to year, he could write a semistructured data query (see Fig. 3.6). The output of the query is another XML file containing the publications of the researcher ordered by year of publication (see right side of Fig. 3.5). I use the StruQL query language [FFK$^+$98, FFLS97] to illustrate the query example in Fig. 3.6, but there exist several other semistructured and XML QLs such as Araneus, Florid, Lorel, WebOQL, WIRM, YAT, XSL/XSLT, and XML-QL [FLM98].

A query to restructure an information space actually mixes two distinct modalities of information seeking. Typically, the data to restructure is a subset of an information space and retrieved via the WHERE clause of a semistructured data query. The WHERE clause thus serves as a match operator. Once data is bound to variables in a WHERE clause, manipulation of those variables within the CONSTRUCT clause of a query (the LINK clause in the case of StruQL) restructures the space. Thus, reconstruction activities take place following a retrieval or match operation. These operations are performed by the information system at query execution time. Analogous to WSQ [GW00], user intervention is unnecessary to realize the mixture of aspects of information seeking. Systems supporting reconstruction via querying are best classified as templates for personalization, because users specify all aspects of information seeking at query formulation time.



Location: http://e.my.yahoo.com/config/customize?.page=p1&.done=htt

Choose up to 20 content modules on this page by checking or un-checking the boxes. Be sure to click **Finished** when you're done. Find out what's new on My Yahoo!

Personalize Content for: **Lucy's Page**

Page Name: Lucy's Page

My Yahoo! Essentials
- ☐ Headlines (w)
- ☐ Message Center (n)
- ☐ Weather (n)
- ☐ New on My Yahoo! (n)
- ☐ Yahoo! Mail Preview (w)
- ☐ Yahoo! Search (n)

Business & Finance
- ☐ Stock Portfolios (n)

Sports
- ☐ 2002 FIFA World Cup Korea/Japan[tm] NEW! (w)
- ☐ Sports Scoreboard (n)
- ☐ Fantasy Sports (n)
- ☐ Ski Report (n)
- ☐ Team Calendars (w)
- ☐ Team News (w)

Travel
- ☐ American Airlines Specials NEW! (n)

Figure 3.7: The content template for personalization of My Yahoo!. In this form webpage, users select desired content within categories to appear on a My Yahoo! personalized webpage. Users may similarly customize layout and color in a personalized webpage.

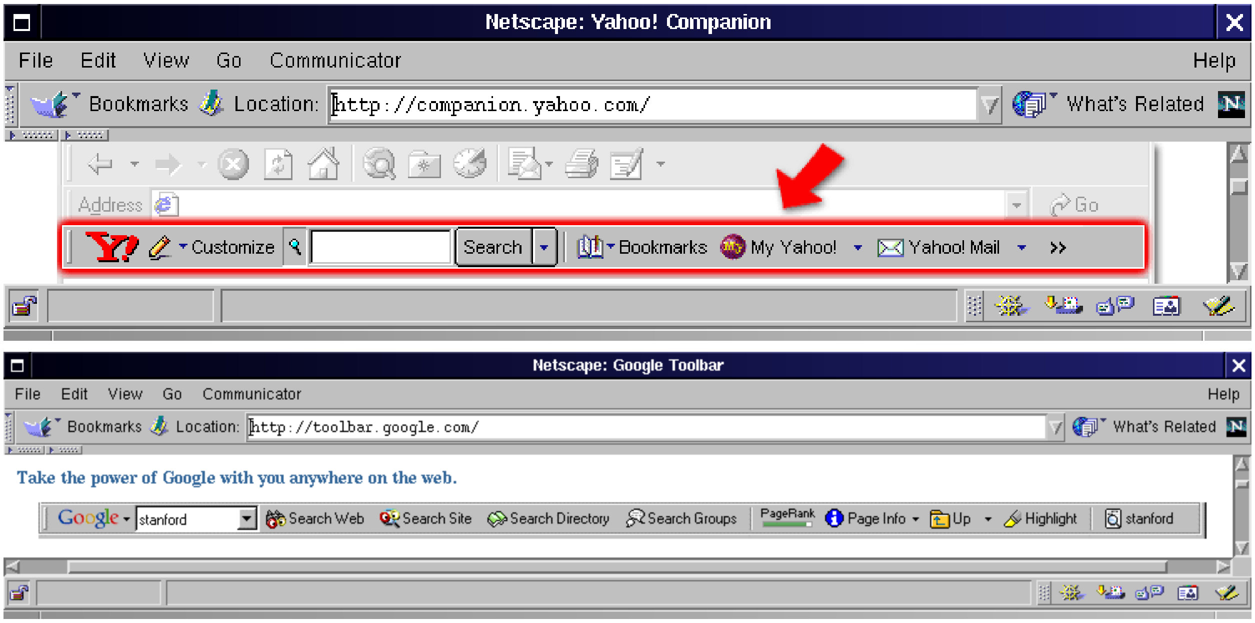
3.2.4 Personal Information Spaces

Yahoo! provides many tools, e.g., My Yahoo!, Yahoo! Companion, and Inside Yahoo! Search, for managing one's personal information space [MPR00]. My Yahoo! [MPR00], a manually customizable web portal, has been freely available since 1996. With My Yahoo! users may customize the content and layout of a personalized webpage. See Fig. 3.7 for the content template for personalization of My Yahoo!. Interaction here entails filling in pre-defined templates and is referred to as check-box personalization.

There are many such sites which provide templates for creating My sites. These types of templates are simply an abstraction of a personal webpage with infrastructure provided by a third party (e.g., Yahoo!). After exploring these tools for personal use, I conclude that their usefulness is limited by the absence of interactivity and interaction.

Nevertheless, one of the main attractions to My sites is the ease with which they permit users to manage centralized bookmarks. Personal user bookmarks provide fertile ground for collaborative filtering [THA$^+$97] if bookmarks may be shared. The Siteseer system [RP97] mines overlap in bookmark folders to deliver personal recommendations of webpages to users. In addition to serendipitous webpage recommendation, users who interact with many computer systems and clients on a daily basis need central access to bookmarks.

Therefore, beyond providing templates for personalization, many of the My site providers,



Location: http://companion.yahoo.com/

Location: http://toolbar.google.com/

Take the power of Google with you anywhere on the web.

Google: stanford

Figure 3.8: Static browser toolbar plugins: (top) The Yahoo! toolbar called Yahoo! Companion provides ubiquitous access to bookmarks, e-mail, and web search. (bottom) The Google toolbar provides direct access to web search operators such as within site search, search term highlighting, and word-find.

including Yahoo! and Google, implement web browser toolbars. The main goals of these toolbars are to provide ubiquitous access to bookmarks (stored in the My page), e-mail, and web search. See Fig. 3.8 for examples of popular embedded toolbars for web browsers. Interacting with a toolbar template for personalization is useful, but again limited. These toolbars are static and only provide direct access to stored information. A toolbar that facilitates a dialog between a user and browser is a vision for personal interaction.

3.3 Operators for Personalized Interaction

Recently, supporting the seamless integration, combination, and composition of many atomic operators by the end-user in compelling ways has become popular [Rie00a]. In this section I analyze several systems and projects which provide this functionality.

3.3.1 Search and Results Refinement

Many search systems provide users with operators to refine searches and improve search results. Typically, such operators are iteratively invoked during the course of an information-seeking session. Some operators such as relevance feedback are broad and directed toward helping users focus an initially imprecise query. Other operators, such as the 'search with results' functionality provided in many web search engines, are focused to reduce a results space. Some systems provide a hybrid of the two with a clustering operator. Users may cluster to prune results or cluster an original information space to facilitate query formulation.

I expound on all three operators below.

Relevance Feedback

Relevance feedback is concerned with addressing the mental mismatch issue in query formulation. Namely, the vocabulary which a user employs to specify an information-seeking goal may not match the terms in the system representing the desired information. This should not come as a surprise, since information seeking itself is ultimately concerned with resolving a problem for which existing knowledge is inadequate [Bel00]. This problem has been identified by many in the information systems community.

> "The major problem in interaction for naive users is therefore the large semantic gap between the user model (concepts) and the system model (words)" [Sac00].

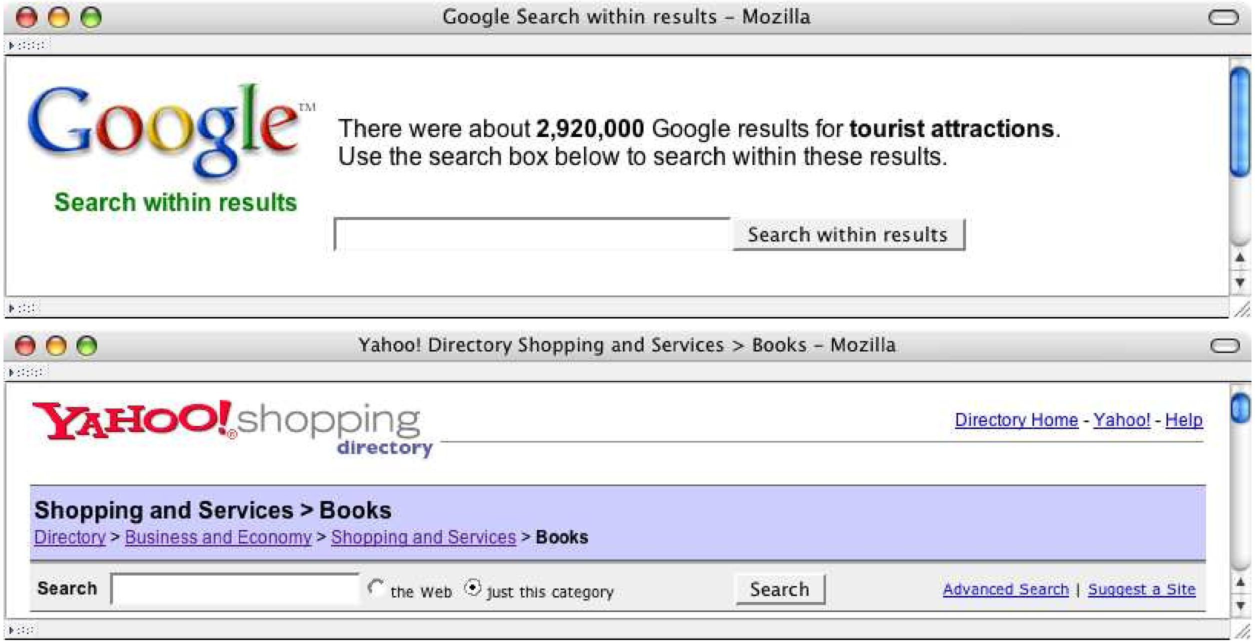
Some systems provide static functionality supporting mnemonics to address this problem. Other researchers however contend that interaction is an ideal vehicle by which to formalize an information-seeking goal.

> "...the essence of 'interactive retrieval' lies in the constant adjustment between 'answer evaluation' and the 'command formulation' tasks to achieve user satisfaction" [Chi97].

In the mid-1960s, Rocchio developed an interactive technique for tackling this problem called 'relevance feedback' [Roc71]. Relevance feedback entails iteratively ranking search results by the user in order to correctly reformulate an information-seeking query. This helps to distinguish relevant results from irrelevant results and aids in query refinement. The process terminates when the user is satisfied that the query is ideal. Since the problem of finding the correct words for a successful search is still endemic to information systems today, much research has been conducted on interaction styles for relevance feedback. Belkin contends that information foragers would rather take a laissez-faire approach (i.e., uncontrolled term suggestion) toward query reformulation than explicit relevance feedback [Bel00]. I direct the interested reader to [CCTL01, HR01] for treatment of relevance feedback in the context of recommendation and personalization. Relevance feedback also has been employed as a technique to model user interests [MMLP97].

Web Search

While visions for future web search engines include dynamically directing users with computed links [Hea00], currently refinement operators are employed to provide aspects of personalization. Another results refinement operator, quite complementary to relevance feedback, is 'search-within.' While relevance feedback addresses a broader problem, a correctly formulated query is implicit in search-within. The operator simply reduces search to the scope of a particular information space, typically results. Search-within operators are predominantly seen in web search engines such as Google, HotBot, and Lycos. Fig. 3.9 (top) shows Google's interface design for searching within results. On the other hand, web taxonomies such as LookSmart and Yahoo! provide search capabilities at every step while



There were about **2,920,000** Google results for **tourist attractions**.
Use the search box below to search within these results.

Google Search within results - Mozilla
Google™
Search within results
Search within results
Yahoo! Directory Shopping and Services > Books - Mozilla
YAHOO! shopping directory
Directory Home - Yahoo! - Help
Shopping and Services > Books
Directory > Business and Economy > Shopping and Services > Books
Search
○ the Web ● just this category
Search
Advanced Search | Suggest a Site

Figure 3.9: (top) Free form query interface for the search-within results operation in Google. (bottom) The interface to Yahoo!'s search-within category. Designs such as these provide a simple form of integrating personalization and browsing.

drilling-down categories in a hierarchical fashion. The interface design of Yahoo!'s free form categorical search is shown in the bottom of Fig. 3.9.

Such search functionality integrates browsing and personalization; to support an interactive experience, however, search-within operators should be closed and applicable at any point in the information-seeking session. The search-within results operators available in Google, HotBot, and Lycos are closed. The search-within category operator, such as that seen in Yahoo! and LookSmart, is however not closed. For example, if a user initiates a search while browsing a category hierarchy in Yahoo!, interaction via hierarchical browsing is disrupted and the user is returned a flat list of results without further search or hierarchical browsing capabilities.

Lastly, integrating modes of information seeking is seen at other levels in Yahoo!. Since Yahoo! provides a suite of specialized webpages (e.g., travel pages at travel.yahoo.com, movie pages at movies.yahoo.com, and maps at maps.yahoo.com), designers envision personalizing searches according to the category of the request [MPR00]. For example, if one searches for 'Mission Impossible,' Inside Yahoo Search can direct one to the appropriate page within movies.yahoo.com.

Clustering

Clustering elegantly reduces information overload and prevents users from sifting through many similar results. Results clustering can aid answer examination while initial clustering familiarizes a user with an information space. In many search engines, including AltaVista and Google, clustering of results is done by default so users do not see more than two pages

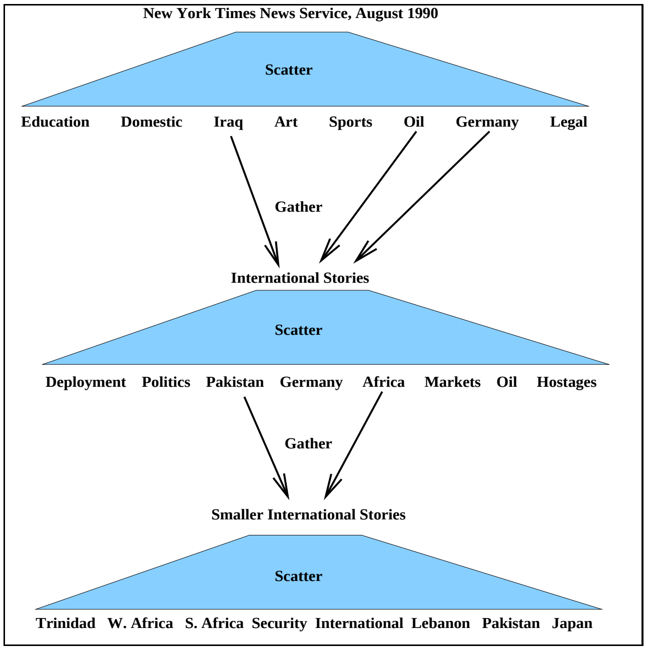


Figure 3.10: Interaction with Scatter/Gather.

from the same site.

Search-within functionality and clustering capabilities are just two of the many operators available in web search engines. Others include similarity and 'from links' searches. I omit discussion of these here and refer the interested reader to Search Engine Watch at seachenginewatch.com for details and comparisons. A cursory look at implementation details and structural differences in search engines is given in [Tho98].

3.3.2 Scatter/Gather

I present the Scatter/Gather project [CKPT92] as an example of a system that provides operators for personalized interaction. The two interactive information-seeking operations being integrated are scattering (clustering) and gathering (browsing). I begin my discussion with some motivation for the work in [CKPT92].

Several research projects have addressed the use of document clustering algorithms to improve information retrieval. Due to accuracy constraints however, such algorithms have poor, quadratic, run-time complexities. Therefore, these algorithms have not been widely accepted by the IR community. The Scatter/Gather project employs document clustering for different objectives. Instead of attempting to improve information retrieval via clustering, it aims to enrich browsing experiences via clustering. Clustering facilitates the formulation of an information-seeking goal by the user. Clustering in the context of Scatter/Gather is more sophisticated that the clustering for web search results described above. For instance, it entails more than collapsing webpages from the same site.

Interaction with the Scatter/Gather system is as follows. Essentially, a one-time, offline clustering of a document corpus is performed. This initial step is expensive. Afterward, clustering is done in an online, iterative, and interactive fashion. Clustering is the scatter-

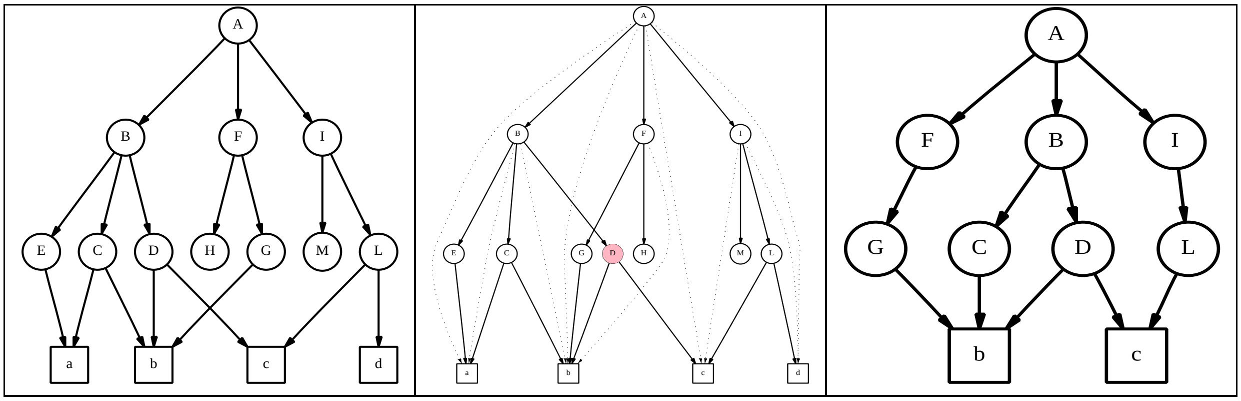


Figure 3.11: Illustration of the zoom operation in Dynamic Taxonomies. (left) A multidimensional dynamic taxonomy. (center) Extensional inference of all concepts related to node D. (right) The reduced taxonomy after a zoom operation on concept D.

ing component of Scatter/Gather. Clusters are described to users via terms and succinct summaries. Thus, in addition to employing clustering algorithms, Scatter/Gather makes use of summarization algorithms. These algorithms essentially consider the central words of a cluster (i.e., those which appear most frequently in the group as a whole). After an initial scatter, a user selects clusters which she wishes to explore further. This step comprises the gather phase. After gathering clusters, the documents of those selected clusters are merged and re-clustered. Then, the scatter phase resumes. This interactive and iterative process continues until a user has honed in on a desired set of documents. The interleaving of scattering and gathering operations drives the information exploration process. During this process, themes of the corpus are extracted and presented to the user. One advantage of this approach is that no browsing hierarchy is hardwired *a priori*. Rather, a hierarchy is created quite naturally, on-the-fly, via clustering. Interaction with Scatter/Gather is illustrated in Fig. 3.10 (regenerated from [CKPT92]).

The interactive nature of the personalization operators available in Scatter/Gather leads us to categorize the project here. Modes of information seeking in Scatter/Gather follow a strict, ordered sequence dictated by operation semantics. Interaction begins with a gather operation and proceeds in a scatter, gather, scatter, gather fashion. One cannot arbitrary intermix these operations. Two scatter operations in succession produce the same set of clusters. Furthermore, gathering does not make sense if it is not immediately followed by a scatter. While these two modes of information seeking may be specified and performed over time, they are complementary and dependent on each other. Neither have semantics in isolation because no hardwired hierarchical schema is in place from the onset.

3.3.3 Dynamic Taxonomies

Another project closely related to Scatter/Gather is Dynamic Taxonomies [Sac00]. The motivation here is personalizing a taxonomy with set-theoretic operations (e.g., union and intersection). In this context a dynamic taxonomy is a model of an information space which can be browsed and simplified by set-theoretic operations. A user may drill-down a taxonomy to arrive at an interesting node. At this point in the interaction the user may continue to browse or perform a 'zoom' operation.

The adaptation, reduction, and dynamic nature of a taxonomy via the zoom operation is computed by extensional inference. The zoom can reveal relationships in the original taxonomy which even the designer may be unaware of. One caveat to this approach is that the original taxonomy must be multidimensional (i.e., each atomic data item is classified under more than one concept).

Interaction with a dynamic taxonomy, and adaptation and reduction of it proceed as follows. Consider the multidimensional taxonomy shown on the left side of Fig. 3.11 (adapted from Figs. 6 and 7 of [Sac00]). The zoom operation begins with extensional inference. When a particular concept is selected (D in the case of Fig. 3.11), all the data atoms under this concept are computed. Performing a zoom on concept D of the taxonomy in Fig. 3.11 (left) infers the intensional relationships illustrated with dotted arcs in Fig. 3.11 (center). As is shown, the zoom operation reduces the taxonomy to all the data items (i.e., concept nodes and atomic nodes) classified directly under the node which the zoom was performed on (in this case, node D). In addition, the taxonomy retains the other nodes and paths which lead to the atomic nodes classified under the zoomed node. All nodes which do not lead to those atomic data items are pruned from the taxonomy yielding a reduced taxonomy or a conceptual summary (see Fig. 3.11, right). The zoom operator is closed.

With multidimensional taxonomies it is easy to see that a conjunction of the sets of ancestor nodes of the corresponding atomic objects under which a zoom is performed thins an information space. Furthermore, multidimensional taxonomies yield all set-theoretic operations applicable and useful (of which intersection is the most powerful). If the information base is restricted to monodimensional taxonomies (referred to as 'conventional taxonomies' in [Sac02]), conjunctions result in null sets, yielding set union as the only applicable operator. Union operations however do not simplify the taxonomy, but rather expand it and thus do not reduce information overload. Sacco compares conventional (monodimensional) and dynamic (multidimensional) taxonomies and provides experimental results in [Sac02].

The two modes of information seeking mixed in Dynamic Taxonomies are browsing and zooming (also referred to as 'taxonomic retrieval' in [Sac00]). Sacco refers to this paradigm of information access as 'information thinning' [Sac02]. While decision points at which to browse or zoom are determined by the user, there is an ordering on such activities dictated by operation semantics. For instance, two zoom operations in succession yield the same taxonomy present prior to the second zoom operation. On the other hand, performing the second zoom operation on a different node transforms the taxonomy. The new node being zoomed upon must however be arrived at via browsing. It is clear that the zoom operation is subservient to browsing. Browsing operations, however, can be performed independently of zooming.

Since browsing and zooming are performed interactively and subject to constraints, the interaction model of Dynamic Taxonomies is similar to that of Scatter/Gather [CKPT92]. In other words, in both systems, the application of available operators for personalization is constrained. It is interesting to note that Sacco does not explicitly allude to this interaction constraint in [Sac00].

A byproduct of Sacco's approach is that dynamic taxonomies can be nicely integrated with other retrieval methods (e.g., IR and DB queries). For example, Sacco states that 'extensional inference can be applied to any subset of the information base, no matter how generated, and thus guarantees a tight, symmetric coupling with other retrieval methods' [Sac00]. Such integration and associated distinctions do not alter my classification of Dynamic Taxonomies as affording operators for personalization. In conclusion, Dynamic Taxonomies is simply a set-theoretic model to realize combinations of information-seeking activities. In the following systems I investigate, no constraints exist on the composition or application of available operators for personalization.

3.3.4 RABBIT

RABBIT is a novel information system that was well ahead of its time (circa 1984). Many of the ideas motivating RABBIT are related to several of the papers and systems I analyze in this section. There appears to have been a gap in the literature addressing the pertinent issues (mental mismatch and combinations of interaction operators) from the time that RABBIT was published up until nearly 1995.

Essentially, RABBIT provides a unique interface to a DB. Browsing an information space is the main interaction motif. While affording compelling browsing experiences, the interface is based on the paradigm of 'retrieval by reformulation' [Wil84]. Retrieval by reformulation allows a user to incrementally specify and formalize an information-seeking goal. Specifically, a user may interleave six closed transformation operators (called critiques) with browsing. The idea is to iteratively refine a query following an operation based on how the system responds to the previous operation. A user query is implicit in the interaction with the RABBIT system. RABBIT distinguishes itself from other IR systems by exploiting partial information. Therefore RABBIT is useful to novices in a particular domain. Specifically, RABBIT assumes that a user knows more about the generic structure of the information space than RABBIT does. RABBIT however knows more about the particulars. The six critique operators available in RABBIT—require, prohibit, alternatives, describe, specialize, and predicate—are expounded in [Wil84].

The most interesting aspect of the RABBIT system is that its reformulation operators (i.e., the critiques) may be specified and invoked at arbitrary points in the interaction. Thus, in contrast to the personalization operators available in Scatter/Gather and Dynamic Taxonomies, RABBIT's operators may be applied in an unbiased fashion. An early interactive information retrieval system similar to RABBIT, which embraces the idea of integrating operators such as browsing and searching, is presented in [CT87].

The systems I present below also exhibit personalization operator independence. After a long absence from the information systems literature (over 10 years after RABBIT appeared), approaching mental model mismatches from an operation-combination perspective resurfaced in [MTW95]. DataWeb motivates the need for personalization operator integration.

3.3.5 DataWeb

In 1995 researchers from IBM Almaden and the Ohio State University wrote a visionary paper which outlines the issues surrounding the mental mismatch problem [Suc87] between the designer and users of an information system [MTW95]. In addition to identifying and expounding on a legitimate cognitive problem, the authors identify approaches to solving the problem. Without using the phrase, the authors discuss aspects of ‘mixed-initiative interaction’ [HM97], in the context of the interface and browsing taxonomies of Yahoo!, as chief among possible approaches.

Mixed-initiative interaction is a flexible dialog management strategy where participants may take turns at any time to seize initiative and direct the flow of interaction without disrupting its contiguous and smooth flow. Mixed-initiative interaction is most commonly observed in human conversations. For instance, the following conversation between a travel agent and traveler illustrates one particular form of mixed-initiative, called ‘unsolicited reporting’ [AGH99].

Conversation
1 **Agent:** Where would you like to travel today, Sir?
2 **Traveler:** New York.
3 **Agent:** Which airline would you like to fly with?
4 **Traveler:** Actually, I’d like to sit in a business-class, aisle seat.
5 **Agent:** Very well.
6 **Traveler:** I also need a vegetarian meal please.
7 **Agent:** Sure.
(conversation continues)

At the beginning of the conversation the agent has the initiative (line 1) and the traveler responds to this initiative (line 2). In line 4, however, the traveler specifies seat preferences ‘out-of-turn’ and hence takes the initiative. Notice that although the traveler does not answer the agent’s airline question (line 3), the conversation progresses smoothly. The traveler could have responded directly to the agent’s airline inquiry (as in line 2), but instead shifted gears and addressed another aspects of the reservation. Such an interaction, where the two parties can ‘mix’ these two modes of inquiry in such arbitrary ways, is referred to as a mixed-initiative interaction; scenarios and systems which do so constitute the landscape of mixed-initiative interaction.

Mixed-initiative interaction with the envisioned DataWeb system [MTW95] is as follows. A user may initially enter a keyword query. The ensuing navigation and summarization of an answer is used to refine the initial and possibly imprecise query. Thus, querying and navigation activities are weaved to facilitate query refinement. One can browse (drill-down or roll-up) or query to attain a different hierarchy at any point while interacting with the DataWeb system. Transition from one operation to another is seamless. While in this context queries induce hierarchies, there are also an initial set of pre-existing hierarchies available as exemplars for a user to browse prior to querying. Similar functionality exists in RABBIT where a user can browse pre-cached hierarchies to exploit ‘find one’ [Wil84] search techniques. Thus, a user may begin an information-seeking activity in the DataWeb system with a query or browse an extant hierarchy. As is seen, DataWeb is a highly interactive system.

The authors make it clear that a user may invoke the available information-seeking operators on demand. There is no pre-determined ordering on the operations. For these reasons and the interactive nature of the outlined system, I view DataWeb as a system affording operators for personalization. The authors partially recognize that no constraints exist on the application of their information-seeking operators.

3.3.6 Web Browser Command Shells

The UNIX operating system comes bundled with many useful, focused, and atomic software development tools such as `cat`, `grep`, and `sed`. While these tools have merit in isolation, much of the success of the UNIX can be attributed to the command shell which supports the composition and communication of such powerful tools via pipes. Such composition supports user interaction in creating a compelling and personal experience with the system while developing software. In other words, the design of tools in UNIX has been carved up at a comfortable and personable level of granularity. Furthermore, the communication mechanism, which is provided by the shell, allows end-users to become programmers on-the-fly. A similar approach to personalization is advocated in [Smi00]. The ideas presented here are motivated in [Rie00a].

Interaction with a web browser also entails invoking atomic functions (e.g., clicking on a hyperlink). Furthermore, many popular browsers integrate access to other tools through fancy user interfaces. For example, many web browsers today provide one-click access to an e-mail application. What browser vendors are yet to provide is a communication mechanism to support the composition of these atomic web tools. Consider the following scenario of interaction to motivate this idea.

> Lucy launches her favorite web browser. The browser opens to her startpage – the homepage of CNN.com. The headline highlights the summer heat wave on the west coast and reminds Lucy of her trip to the Grand Canyon next week. This reminder compels Lucy to open her mail utility from within the browser to retrieve an e-mail sent to her last week regarding heat precautions. Upon opening the mail client, Lucy uses the find command in the browser to retrieve the message. After locating it, she opens the mail message and immediately begins clicking on the URLs provided therein. These clicks spawn page loads in her browser. After a series of mouse clicks on URLs, page loads, and invocations of the find utility of the browser (to scan the page), Lucy realizes she has found a webpage of interest. She next prints the webpage so she can take it with her on the trip. Lucy closes her browser and terminates the information-seeking session.

The above scenario of interaction demonstrates that the web browser has provided easy and central access to all the tools needed to complete the information-seeking interaction (i.e., e-mail, HTTP requests, find, and print). Interaction with the browser is however discrete and discontinuous in the information-seeking episode. Although Lucy knows what she is looking for from the start, she has to go through a series of individual and painstaking tasks. Providing a mechanism within the browser to coordinate the communication between these autonomous tasks on demand would permit a user to create personal interactions. It is the interleaving of these autonomous commands that is currently done by manual invocation, and which would benefit from personalization.

LAPIS: Engaging Your Browser

Many of these ideas were first introduced in [MM00] and implemented in a browser shell called LAPIS (Lightweight Architecture for Processing Information Structure). The capabilities of LAPIS include a pattern language, a scripting language, and the ability to

```
SELECT y.URL
FROM   x in Fragment, y in Fragment
WHERE  x.URL = ‘‘http://www.yahoo.com/headlines/tech/’’
       x.HREF = y
       y.CONTENT = ‘‘*Microsoft*’’;;
```

Figure 3.12: An AKIRA query. The semantics of this query are to i) locate and fragment the specified webpage, ii) load each webpage that the specified webpage references, and iii) search all collected fragments for the text ‘Microsoft.’

invoke external programs. Collectively these entities help cultivate a new shell interaction model [MM00]. Extensions to this research include providing support in an interface for a user to create a script ‘by example’ (also called ‘automation by demonstration’) [MM00] and enriching captured context such as browsing history. With the advent of the XML suite of technologies, I expect such approaches to become more feasible and subsequently gain widespread acceptance. While toolbars such as LAPIS are a step in the direction toward engaging a browser in a dialog, high levels of sophistication are not seen. I surmise that more research on representing and reasoning about user interaction in information systems will aid future systems [Mar97]. The following system employs elaborate data modeling to facilitate combinations of information-seeking activities.

3.3.7 AKIRA

The AKIRA project [LSCS97] at the University of Pennsylvania has a theme similar to that of WSQ. The project attempts to incorporate data on the web into a canonical DB query. Instead of simply dealing with web search results as URLs and associated frequencies, the AKIRA project models webpage content. Modeling webpage content gives users the freedom to be expressive in queries. Within-webpage modeling can also affect the granularity of answers. The model employed to capture the webpage data in AKIRA is object-oriented. The information-seeking operators which are mixed in an interactive manner are browsing, querying, and output restructuring. While I classified WSQ and web query languages as templates for personalization, I view AKIRA as providing operators for personal interaction. Information-seeking sessions with AKIRA are interactive and no constraints exist on the order in which operators may be invoked.

A user interacts with the AKIRA system as follows. After the user poses a query (see Fig. 3.12, a modified version of the query presented in [LSCS97]) and receives an answer, she may browse the resulting pages or write another query to restructure the output. Furthermore, points at which these information-seeking activities are engaged may be mixed in any order. User interaction with AKIRA is similar to that with RABBIT. As opposed to RABBIT however, querying (including output restructuring) and browsing are the only two valid information-seeking operations available in AKIRA.

3.3.8 Complete Answer Aggregates

Meuss and Schulz's complete answer aggregates [MS01] are tree based data structures used to facilitate the integration of browsing, querying, and reformulation in an information-seeking session. Meuss and Schulz define a complete answer aggregate as 'a complete and non-redundant view on all the possible target nodes, for each of the query variables, and on all links between these candidates that contribute to some answer' [MS01]. The approach of complete answer aggregates is based on sets, relations, and tree theory.

Interaction with the system proceeds as follows. A user writes a tree structured query, whose answers map tree query nodes to DB nodes. Since the number of answers to a tree query may be exponential and thus possibly lead to information overload, a method by which to summarize and compact the answer is required. The solution adopted is factorization, which not only compacts the answer, but also arranges relevant data elements of the answer in context. Such qualification was also the primary motivation for Dynamic Taxonomies [Sac00]. A terse but expandable answer is preferred over a long, flat, and monolithic list of hits.

Aspects of information seeking are supported by information previews (e.g., counters) to facilitate decisions on whether to construct and issue another query, drill-down, or reformulate. Reformulation here is considered as a special case of querying. Meuss and Schulz provide two closed reformulation operations: node rank by counter values and compaction by attribute values [MS01]. Surprisingly, the two useful operations on answer aggregates do not directly correspond to any of the six critique operations in RABBIT [Wil84]. The main idea is that an initial tree query will present a useful starting point for active exploration of an answer space. Meuss and Schulz contend that such exploration facilitates 'interactive knowledge discovery and hypothesis testing' [MS01]. Subsequent browsing and reformulation is employed to refine/enhance an initial, possibly under specified, query.

Connections from complete answer aggregates to Dynamic Taxonomies [Sac00] and RABBIT [Wil84] is seen in that all three projects model an information resource and provide canned, closed operations (including browsing) on that resource to transform, simplify, and personalize it. The zoom operation is available in Dynamic Taxonomies. In RABBIT, available operators are reformulations. Closure preservation in complete answer aggregates fosters both an exploratory style of browsing and seamless integration with further query type activities (reformulations). This browsing style is similar to that in some OLAP (Online Analytical Processing) systems [HAC$^+$99].

It is clear that the operators here (i.e., the specification of attribute values to collapse by and the specification of counter values to rank by) are independent of each other and need not arrive in an ordered or predetermined fashion. Furthermore, although not explicitly mentioned by the authors, I believe that another tree query may be written against a complete answer aggregate (at a different time in the interaction). Such interaction exemplifies the interleaving of information foraging activities with browsing. While the authors do not explicitly address this aspect of their approach, they do stress the exploratory nature of complete answer aggregates. At different points in time, different aggregates may be viewed via certain attributes. Since the available operations may be specified in an unbiased fashion over time, interaction with complete answer aggregates is similar to that in the RABBIT system.

3.3.9 BBQ and MIX

XML as a data format provides opportunities for mixing operations for personal interaction, especially browsing and querying. Typically XML data elements are nested, making XML documents conducive to browsing via drill-down and roll-up metaphors. In addition, most XML query languages such as XML-QL are closed [DFF$^+$99]. Thus, interactively blending browsing and querying of XML is quite natural.

Blending Browsing and Querying (BBQ) [MLP00, MP00] is an information system which achieves precisely this objective. There are no system semantics dictating the order in which a user may apply the two information-seeking operations. Querying in BBQ [MLP00], as opposed to more traditional XML query languages [DFF$^+$99], may be performed by example via a drag and drop interface. Thus, querying in the BBQ system is interactive, as opposed to the one-shot style of interaction seen in other systems [FR97, GW00]. After a query is answered, the system infers a document type definition (DTD). This DTD assists the processing of subsequent queries.

I view BBQ as an information system which affords operators for personalization since querying is interactive and combined independently and at any interaction point with browsing in BBQ. The designers of BBQ do not recognize this aspect of their system. BBQ is currently absorbed by a larger project called MIX [MP02]. MIX is a mediator-based approach to integrating querying and navigation. While BBQ incorporates visualization, there also exist systems, focused solely on visualization, which afford operators for personalization as a convenient by-product.

3.3.10 Operators for Interactive Visualization

Interactive information visualization is the main thrust of the systems I discuss in this section. These systems provide operators to bring aspects of interactivity to bear upon a visualization. Ultimately and as a by product, such operators tackle the mental mismatch issue [Suc87], which is endemic to personalization research. Therefore, I showcase these operations here in the context of the personalization they achieve. While there are several interactive information visualization systems, I focus on three which provide operators which affect user perception of an information base. Specifically, I analyze three data structures: user-defined hierarchies [WB99], polyarchies [RCCR02a, RCCR02b], and treemaps [Shn92, SW01]. A unifying theme among these systems is their ability to provide visualizations and views which expose semantic relationships in an information base.

User-Defined Hierarchies

User-Defined Hierarchies (UDHs) are dynamic hierarchies. Systems incorporating UDHs champion multiple visual layouts of a single hierarchy and therefore support dynamic hierarchy specification and visualization. Multiple layouts facilitate the discovery of semantic relationships in data. Various different layout algorithms, each with support for discovering different properties (e.g., level of clustering) efficiently, are discussed in [WB99]. Such algorithms modify a hierarchy dynamically based on user interaction. Dynamic hierarchies are generated directly from data and not as a result of operations or transformations on

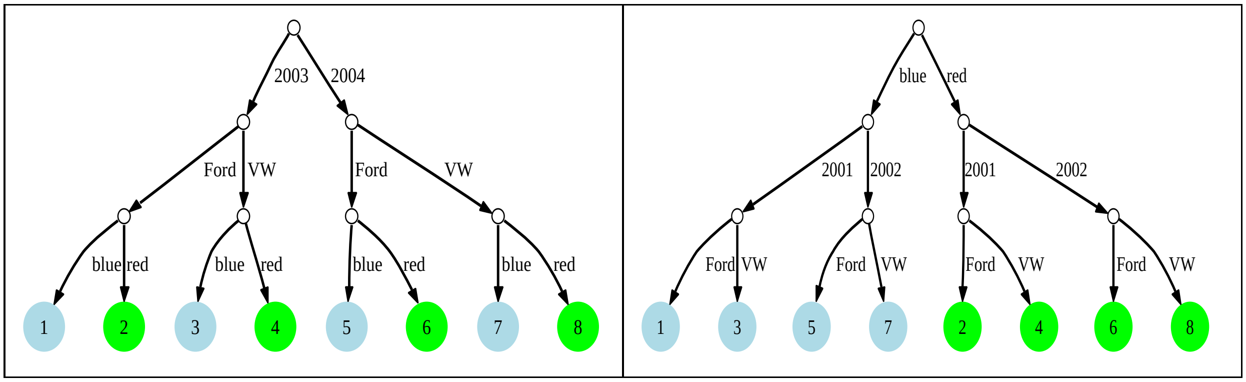


Figure 3.13: Illustration of a possible reconstruction operator on a UDH. The UDH description of the hierarchy on the left is modified to restructure the levels of the hierarchy to that shown on the (right).

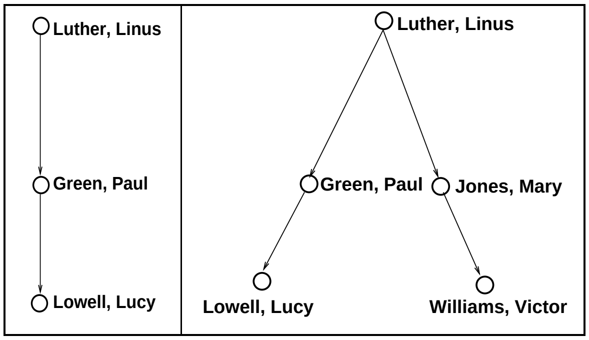


Figure 3.14: Adding a person to a management polyarchy. (left) The path from the root 'Luther, Linus' to 'Lowell, Lucy.' (right) The polyarchy resulting from adding 'Williams, Victor.' This figure illustrates how a user can incrementally add entities to a polyarchy which reveal resulting relationships. Such relationships are difficult to observe with a general overview of an extremely large hierarchy.

an 'unpersonalized' hierarchy or representation. Modeling interaction is thus not stressed in [WB99]. Fig. 3.13 illustrates a possible reconstruction operator. A UDH, whose first, second, and third levels pertain to automobile year, model, and color, respectively, is shown in Fig. 3.13 (left). Fig. 3.13 (right) might be the output of a goal-oriented reconstruction of the UDH description of Fig. 3.13 (left). This reconstruction reorganizes the hierarchy by making automobile color, year, and model the first, second, and third levels, respectively.

Polyarchies

Polyarchies [RCCR02a, RCCR02b] deal with predetermined (static) hierarchical structures. A polyarchy groups multiple intersecting hierarchies which share at least one node into a single hierarchical structure. Again, the main focus here is on visualization. A polyarchy helps to visualize both a single hierarchy and understand the relationships between multiple entities within that single hierarchy. In addition, a user may visualize more than one hierarchy simultaneously for a clear understanding of the relationships between multiple

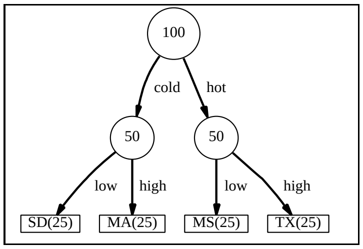


Figure 3.15: A tree containing data about states.

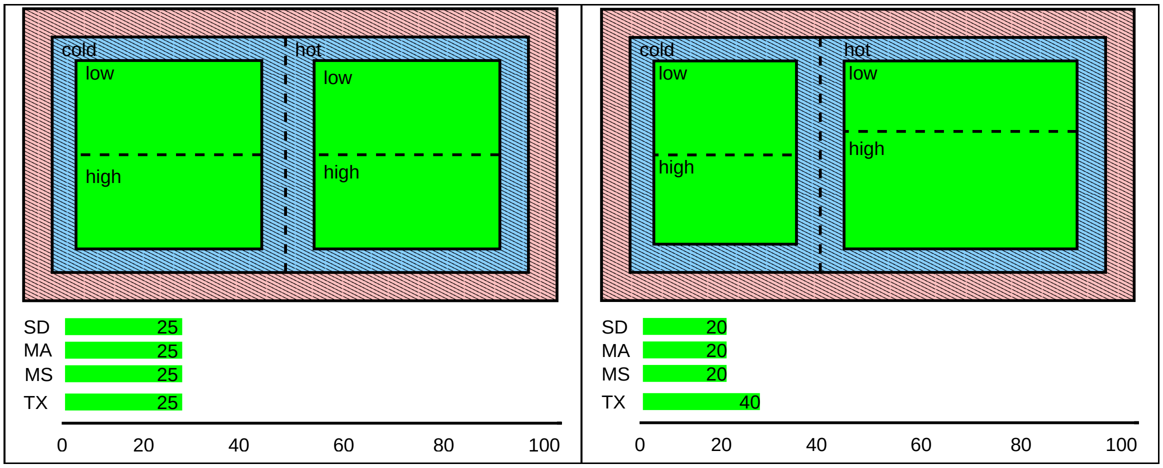


Figure 3.16: Illustration of the slide operator to adjust weights of attributes in treemaps. (left) A possible treemap for the hierarchical data shown in Fig. 3.15. (right) Resulting treemap, which displays the recalculated state weights, from moving the sliders in (left).

hierarchies. To facilitate these goals, manipulations such as sliding and pivot points are provided. Fig. 3.14 (adapted from [RCCR02a]) illustrates adding a new object to a polyarchy. Fig. 3.14 (left) shows one path in a management hierarchy – the path from the root 'Luther, Linus' to 'Lowell, Lucy.' Fig. 3.14 (right) shows the result of adding 'Williams, Victor' to the polyarchy – an additional path to the root. The use of this addition operator in this example illustrates the discovery of relationships between the selected entities. This approach distinguishes relationship discovery in polyarchies vs. that from a general overview of an extremely large hierarchy.

Treemaps

Treemaps are yet another data structure approach to inferring relationships in an information base. Similar to polyarchies, treemaps deal with predetermined structures – trees in this case. The traditional two dimensional treemap approach is discussed in [Shn92]. The treemap3 system (see www.cs.umd.edu/hcil/treemap3/) extends this by allowing users to choose the aggregation order to form a tree of their choice. Layout difficulties in visualizing treemaps as opposed to supporting multiple aggregation orders are discussed in a newer article [SW01].

The tree in Fig. 3.15 (adapted from [Jon98]) models attributes of states. The first level of the tree involves values for climate while the second level contains values for population.

The number in a node represents the weight of the node which is equal to the sum of the weights of all the descendants of the node. Fig. 3.16 (adapted from [Jon98]) illustrates direct manipulation of treemap attribute weights to recompute the value of objects (e.g., the weight of states in Fig. 3.15). The left of Fig. 3.16 displays a possible treemap and state weights for the data in Fig. 3.15. Users may adjust the weight of attributes in this treemap by manipulating the dotted sliders. A user may move the sliders to explore the cumulative effect that different attribute weight values have on the objects (states, in this case). In Fig. 3.16, moving the sliders corresponds to adjusting the relative importance of preferences. Such an interface helps the user decide on, for example, relocation options. After manipulation, the value of each object (e.g., state) is automatically recalculated as illustrated in Fig. 3.16 (right). Such manipulations are critical to decision support systems as seen below. I now turn to interaction as a vehicle to analyze massive data sets.

3.3.11 Interactive Data Mining and Analysis

Close examination of data analysis in DBMSs, decision-support systems, and data mining packages from a user perspective reveals that analysis calls for iteration, intuition, and exploration. I have established that a query in a DBMS is a one-shot activity. Such an approach is effective when a user knows what he is seeking, but is not conducive to exploration. Thus, a user experiences frustration when using a query information-seeking strategy to search for information that the user does not know [Bel00]. This problem is endemic when DBMSs and IR systems are used as interactive systems. As discussed above, results refinement techniques such as relevance feedback are typically employed to combat this problem.

This issue is exasperated in decision-support systems and data mining applications for the following reasons. Typically batch analysis of large data sets is costly and time consuming. Often the success of algorithms which discover patterns in data is predicated on and highly sensitive to algorithm-specific parameters (e.g., support and confidence) tuned by users. A poor choice of parameters may lead to useless results. The results of the first few runs on massive data sets may be correct, but undesirable or difficult to interpret. Furthermore, knowledge that a choice of parameters is poor is often unknown until results are returned. In summary, in traditional analysis systems, not only is querying, computation, and analysis one-shot, it also takes place in a 'black-box' [HAC$^+$99]. Computation is conducted as efficiently as possible, but users have no control over it, once begun.

A classic chicken and egg problem ensues. The difficulty is that users can neither precisely formulate their analysis goals nor tweak algorithmic-sensitive parameters until implicit properties of a dataset (e.g., dimensionality) are progressively revealed to them. While much research has been conducted on improving the efficiency of data analysis and mining algorithms in decision-support systems, little research has addressed improving usability and personalized interaction in such systems.

Applying techniques from human-computer interaction to data analysis is an approach. The goal of the Control project [HAC$^+$99] is to afford users direct interaction with computation in order to refine results and control processing 'just-in-time.' Interaction tightens the data analysis process loop. Analogous to the nature of the operations of personalization of the information systems discussed in this section, Hellerstein *et al.* [HAC$^+$99] provide users of analysis tools with canned operations to facilitate interactive exploration. Rather than computations assuming a black-box model, operators for personalized interaction in the Control project afford users direct insight into the ongoing analysis. Such operators trade quality and accuracy of results for direct control. A data mining user is typically willing to accept approximate and partial results in return for a handle into the computation.

The Control project supports many interactive algorithms for data analysis. The supported operations include online aggregation or drill-down online enumeration through user interface widgets to support ‘eyeballing’ and ‘panning,’ online data visualization through a technique called ‘clouds,’ and online data mining. Control employs random sampling and reordering to achieve online interactivity. In addition, Control implements ripple join algorithms to tackle online query-processing problems entailing multiple inputs.

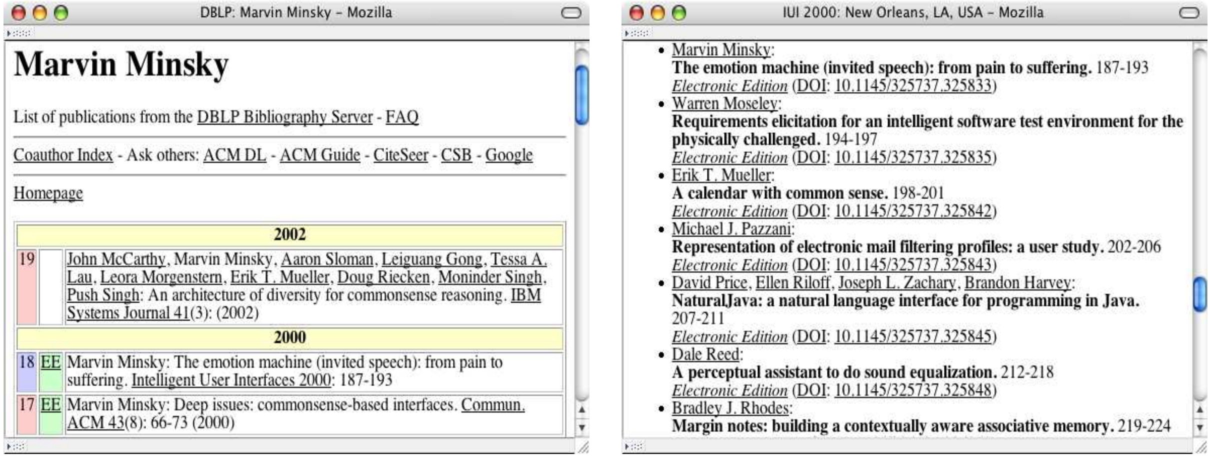
Projects such as Control lie within the scope of data warehousing and OLAP. Data warehousing and OLAP technologies are critical to the success of decision-support systems which currently constitute a large segment of the database industry [CD97]. As OLAP technologies and resulting systems gain widespread acceptance, I expect the need for personalized interaction with them to increase. I view the design of systems such as Control as initial steps in this direction.

3.3.12 Social Network Navigation

While many sites on the web are organized along a hierarchical browsing motif, sites in certain domains are more effectively based on a social network navigation metaphor. A social network is a graph in which nodes represent entities (e.g., people, books, or movies) and edges represent relationships between entities (e.g., is-a-friend-of or have-co-authored-a-paper). Social networks are characterized by heterogeneous nodes and homogeneous edges. A simple example of a social network is one’s network of family and friends. Examples of websites based on a social network navigation metaphor are the Internet Movie Database at imdb.com, Barnes and Noble at bn.com, and the online computer science bibliography DBLP at www.informatik.uni-trier.de/∼ley/db/ (see Fig. 3.17). The increased popularity in the use of social networks has led to projects such as FOAF (Friend of a Friend; foaf-project.org) which attempt to build a web of machine-readable homepages describing people, the relationships between them, and the things they create and do. FOAF utilizes several W3C recommendations, such as RDF (Resource Description Framework) and OWL (Web Ontology Language) and is an application of the ‘Semantic Web.’

Social networks can be induced from an existing information base for later exploration and exploitation. An early project on social network analysis induced a communication network from e-mail logs in order to discover shared interests [SW93]. For purposes here, I am interested in operators to explore and exploit pre-induced social networks, in order to discover products of interest, serendipitous collaborations, or network resources [SYV01]. Such operators enhance personalized interaction and expedite the personalization process.

ReferralWeb [KSS97] is a collaborative filtering recommender system which provides users with operators for exploration and exploitation in a person-person social network. Associations between people nodes are mined from close proximity of names in web documents subject to a set of heuristics. An induced network facilitates the search for experts, communities, or documents. ReferralWeb contains operators for several types of searches including a referral chain search (e.g., a user may be interested in finding the relationship chain be-


DBLP: Marvin Minsky – Mozilla
IUI 2000: New Orleans, LA, USA – Mozilla

Marvin Minsky

List of publications from the DBLP Bibliography Server - FAQ

Coauthor Index - Ask others: ACM DL - ACM Guide - CiteSeer - CSB - Google

Homepage

		2002
19		John McCarthy, Marvin Minsky, Aaron Sloman, Leiguang Gong, Tessa A. Lau, Leora Morgenstern, Erik T. Mueller, Doug Riecken, Moninder Singh, Push Singh: An architecture of diversity for commonsense reasoning. IBM Systems Journal 41(3): (2002)
		2000
18	EE	Marvin Minsky: The emotion machine (invited speech): from pain to suffering. Intelligent User Interfaces 2000: 187-193
17	EE	Marvin Minsky: Deep issues: commonsense-based interfaces. Commun. ACM 43(8): 66-73 (2000)

- Marvin Minsky:
  The emotion machine (invited speech): from pain to suffering. 187-193
  Electronic Edition (DOI: 10.1145/325737.325833)
- Warren Moseley:
  Requirements elicitation for an intelligent software test environment for the physically challenged. 194-197
  Electronic Edition (DOI: 10.1145/325737.325835)
- Erik T. Mueller:
  A calendar with common sense. 198-201
  Electronic Edition (DOI: 10.1145/325737.325842)
- Michael J. Pazzani:
  Representation of electronic mail filtering profiles: a user study. 202-206
  Electronic Edition (DOI: 10.1145/325737.325843)
- David Price, Ellen Riloff, Joseph L. Zachary, Brandon Harvey:
  NaturalJava: a natural language interface for programming in Java. 207-211
  Electronic Edition (DOI: 10.1145/325737.325845)
- Dale Reed:
  A perceptual assistant to do sound equalization. 212-218
  Electronic Edition (DOI: 10.1145/325737.325848)
- Bradley J. Rhodes:
  Margin notes: building a contextually aware associative memory. 219-224

Figure 3.17: An association in the social network at DBLP. Jumping from a author webpage (left) to a conference webpage (right).

tween herself and a colleague and thus ask, ‘What is my relationship to Graham Smith?’), an expert search (e.g., by specifying a topic and a social radius a user may ask ‘What friends of mine or friends of friends of mine know about tourist attractions in Italy?’), and an expert controlled search (e.g., ‘List documents on the topic ‘human factors and user interface components’ close to Don Norman.’). The examples of searches given here have been adapted from those in [KSS97].

Consider how an editor of a journal may exploit a social network of authors in computer science to find an unbiased committee of reviewers for a communicated article. The editor surely does not desire individuals within close proximity of the author under review. The editor does however seek effective reviewers who must be close enough to the reviewees’ research area to be qualified. The editor may therefore apply the available operators on a social network induced from a corpus to find all individuals within three degrees of separation from the author subject to review. Systems consisting of a social network and a suite of expressive operators foster relationship discovery and are thus classified here.

3.4 Representing and Reasoning about Interaction

Thus far this chapter has espoused the theme that personalization is advantageously approached by studying and understanding interaction [Mar97]. In the previous two sections, the onus of personalization was on users. Templates are so over-specified that interaction is limited to filling out a form or writing a query to communicate an exact level of customization. While operators for personalization afford more freedom, interaction remains stifled by constraints on the applicability and composition of the available operators. If interaction is to guide the design of personalization systems, then beyond understanding and studying it, interaction must also be explicitly modeled and exploited. In other words, personalization should be approached from a user-centered design perspective [KNV00]. In my opinion, representing and reasoning about interaction is the holy grail of personalization. The main premise of this section and chapter echoes that of Marchetti *et al.* [MVPB93], namely that

	Dimensions							
	Method		**Goal**		**Mode**		**Resource**	
ISSs	**Scan**	**Search**	**Learn**	**Select**	**Recognize**	**Specify**	**Information**	**Meta-information**
1	√		√		√		√	
2	√		√		√			√
3	√		√			√	√	
4	√		√			√		√
5	√			√	√		√	
6	√			√	√			√
7	√			√		√	√	
8	√			√		√		√
9		√	√		√		√	
10		√	√		√			√
11		√	√			√	√	
12		√	√			√		√
13		√		√	√		√	
14		√		√	√			√
15		√		√		√	√	
16		√		√		√		√

Table 3.1: Four-dimensional information-seeking strategy space.

'...information retrieval is an inherently interactive process, and that support of users should be support of their interaction, with all of the system resources.' The following two chapters reinforce the theme of representing and reasoning about interaction as I illustrate advantages to my representations, in addition to those described here, w.r.t. the interaction they enable.

3.4.1 Why Model Interaction?

Ultimately, models of interaction serve as a representational basis to design an interactive system. They are more expressive than templates and operators and are thus at a finer level of granularity. Care must be taken however to ensure that interaction is not modeled too tightly. In other words, over-representation and excessive modeling can lead to bulky designs. Systems which fall victim to this trap run contrary to the goals of personalization. Representations which are too general should be avoided for obvious reasons as well. This problem suggests the need for structures of interaction at a personable level of granularity. Pednault motivates this issue as:

> "The representation should be as rich and fluid as the interaction itself, but at a level of abstraction that allows the relationships among stimuli and responses to be readily observed in the data collected" [Ped00].

3.4.2 Information-Seeking Strategies

Prior to designing an interactive system, one must first study, understand, and characterize the interactions which users desire of their information systems. Eventually designers shift from such understandings to system design representations which structure, support, and enhance interaction [BCST95]. I begin by characterizing information-seeking behavior.

Belkin, Marchetti, and Cool [BMC93] describe an information-seeking strategy (ISS) as a behavior a user engages in while interacting with a system. They have contributed a binary, four-dimensional ISS space (see Table 3.1) containing 16 (2^4) strategies [BMC93].

Each dimension can be considered as a factor of information seeking and describes a dichotomy. The ISS space factors are method of interaction (scan or search), goal of interaction (learn or select), mode of retrieval (recognize or specify), and resource (information or meta-information).

For instance, ISS15 is indicative of a highly specified search [BMC93]. A user is searching through an information base with the goal of selecting relevant items which match specification aspect input. ISS2, its complement, represents a prototypical example of a fuzzy and loose strategy. Here a user scans meta-information such as an index to learn to recognize where topics are situated. Depending on specific strategy instances, the information-seeking strategies (ISSs) in this space may overlap. More importantly, users typically shift between ISSs in the course of an information-seeking session, called an *episode* in [BMC93].

The following example illustrates such a shift. Consider a student who interacts with a university library information system to check out a reserved book for a course. If the student does not know the title of the book, he may interact with a directory indexed by course number to learn the title of the book (ISS12). After the student knows the title, he can use a search tool to find the book in the title-alphabetized reserve pages (ISS15).

Capturing and modeling such shifts is a way to support compelling experiences in information systems. The classification the space provides can be used to describe movement from one ISS to another. Design techniques to support combination through seamless movement from ISS to ISS are faithful to my vision of personalization through mixture of information-seeking activities as advocated throughout this chapter.

The most notable aspect of this work is that Belkin *et al.* [BMC93] view an ISS as an interaction with an information system. In other words, an interaction with an IR system is a dialog between a user and a system. Others projects divorce the two and view each ISS as a query or functional requirement of a system. Therefore, such systems do not take advantage of the interaction inherent in use. Rather than supporting interaction, such systems constrain, tolerate [BMC93], or react to it. This distinction goes to the heart of the difference between a one-way, reactive, interaction and a two-way, cooperative, dialog.

Most designers make provisions for personalization in systems from the onset rather than supporting it through interaction. This trend is most salient in templates, but also is seen in operators designed to implement personalization. Due to these reasons, Belkin [Bel97] feels that intelligent, agent-based approaches which circumvent the need for personal interaction with information resources are unlikely to be embraced by users.

These arguments have significant implications for the design of a system. Belkin *et al.* [BMC93] prefer the design of a system to explicitly support such interaction, both at the individual ISS and inter-ISS level. The work of Belkin *et al.* is thus visionary in making these novel observations and contributions.

Details of the transition from high level ISSs and interaction models to concrete implementation details need to be pinned down. Through the construction of a prototypical interface to an IR system, Belkin *et al.* [BMC93] explored this transition. The resulting system, called BRAQUE (BRowsing And QUEry formulation), is a two-level hypertext model of IR system DBs [MVPB93]. The system supports and validates the feasibility of the implementation of interaction as described here. In addition, and commensurate with systems presented above, BRAQUE blends query formulation and reformulation with browsing.

3.4.3 Structures of Interaction: Scripts, Cases, and Goal Trees

There are formalisms applicable to modeling interaction. The goals, operators, methods and selection rules (GOMS) model of human-computer interaction, introduced by Card, Moran, and Newell [CMN80a, CMN80b, CMN83] in the early 1980s, is accepted as a mature formalism. Since then, three variations of the original GOMS formulation have been developed—the keystroke-level model (KLM), natural GOMS language (NGOMSL), and cognitive-perceptual-motor GOMS (CPM-GOMS)—and are surveyed in [JK96]. Since GOMS is more of a formalism for modeling *cognition* than interaction, I do not discuss its details here.

Several models which describe the structure of the dialog between a user and an interactive information system have been developed. Three predominate classes of dialog models are transition networks, grammars, and events. They are comparatively surveyed in [Gre86]. Other dialog models, such as language, hybrid, and structural models, are showcased in [PQ96]. Dialog models form the underlying notations used in user interface management systems [Gre86].

Belkin *et al.* [BMC93] use a formal model called COR (Conversational Roles Model) which involves transition networks to represent dialog structures for information seeking. The model defines types of dialog structures between two actors: the information provider and the information seeker. These structures capture turn taking, jumping out of dialogs, termination, and error recovery. COR models high-level dialog structures while omitting details at the domain, task, and strategic levels. Therefore, a prescriptive interaction model in addition to the descriptive COR dialog model is needed. Cases and scripts fill this void.

A dialog is a specific instance of communication between a user and an information system. Dialogs may consist of many moves within a single ISS. For example, while employing an ISS, one user may decide to terminate interaction prematurely, while another may see the information-seeking goal to fruition. In either case, neither user has deviated from the particular ISS. The following is an example of a dialog related to ISS12 of the course textbook example above.

Dialog

1 **System:** May I have the course number please?
2 **User:** Yes. CS4604.
3 **System:** The title of the reserved book is *A First Course in Database Systems.*
4 **User:** Thanks.

Intra-strategy shifts however make dialogs a poor model of interaction for system design.

An interaction script, which is better suited, is a pattern in a two-party interaction or dialog. Belkin *et al.* [BCST95] describe a script as a plan for dialog between a user and an information system. Scripts are prototypes which model a class of concrete dialogs. Therefore, an actual dialog is a specific instance of a script. A script is prototypical in that it implements an ISS. Scripts structure user interaction for the design of a system similar to how an interpreter structures the interaction of a program. The level of expressivity in

1	**System:**	Here's what we can do (offers choice).
2	**User:**	Let's do this (chooses one).
3	**System:**	OK, here's how we'll do it. (presents plan and means for accomplishing script).
4	**User:**	a. OK. → 5 b. No, I don't like this. → 1

Figure 3.18: Preamble sequence for interaction scripts.

scripts is correct for design. Scripts are written in plain English and intended to be easily understood by the layman as opposed to COR models.

An alternate representation of interaction is a goal tree. A goal tree is arranged as a hierarchy of goals which organize the set of necessary moves in an ISS. Goal trees are represented in a PROLOG-style notation with goals corresponding to predicates. There are goal trees associated with each ISS. Furthermore, to model rich interaction, some goal trees may contain sub-goals (predicates) which represent jumps to other regions of the ISS space.

In a simple system design, scripts may be stored in a dialog manager. Upon entrance, a user and the system execute a preamble script to determine and retrieve the desired or appropriate script. Such introduction, which is not specific to any ISS from [BCST95], is given in Fig. 3.18 (from [BMC93]). Combinations of scripts also can be used to achieve more expressive dialogs.

In practice, knowledge of how the dimensions of the ISS space affect each other is invaluable to reduce the number of script combinations which a system must support. Knowledge of these dimensional relationships also makes the prediction of moves between the ISSs at decision points easier. Thus, these relationships help stir user interaction and form complex scripts. Xie [Xie02] addresses how interaction intentions relate to ISSs. Xie identifies patterns of interaction revealing the circumstances under which certain ISSs are employed.

Another approach to interaction shifts is to mine patterns of usage in systems to anticipate which subset of the remaining 15 possible ISSs users will most desire to follow. For instance, if the leaf node in a goal tree cannot be simplified, it can be expanded and replaced with the goal tree of an alternate ISS. This leads to the broader question of where scripts come from.

Belkin *et al.* outline two approaches for deriving scripts in [BCST95]. The first entails 'a general characterization of information-seeking goals and a related cognitive task analysis.' The second is driven by empirical observation of interaction patterns. This approach involves inducing patterns in system use akin to web log mining. Belkin *et al.* use case-based reasoning (CBR) for this purpose. The end-goal is to collect, analyze, and characterize cases in the ISS space.

In such an approach, the system and cases bootstrap each other. After collecting an initial set, the ISS space induces a partition on the gathered cases. Designers then attempt to select a prototypical case from each partition which leads to a script. Due to the iterative nature of CBR, it is acceptable to start the system with a prototype. MERIT [BCST95] is an interactive IR system which embodies these ideas.

3.4.4 Modeling Interaction on the Web

Modeling web interaction is becoming important as information systems migrate online. Such modeling has been done to combat the stateless HTTP protocol. A theoretical approach to dealing with this state maintenance problem is based on the concept of continuations [FWH01] from programming languages. In such an approach, a program is used as a novel model of interaction. The authors of [GFKF01, Que00] model interaction with a web application as a program in Scheme, a language which supports the user in explicitly capturing and manipulating continuations (via `call/cc`, call-with-current-continuation).

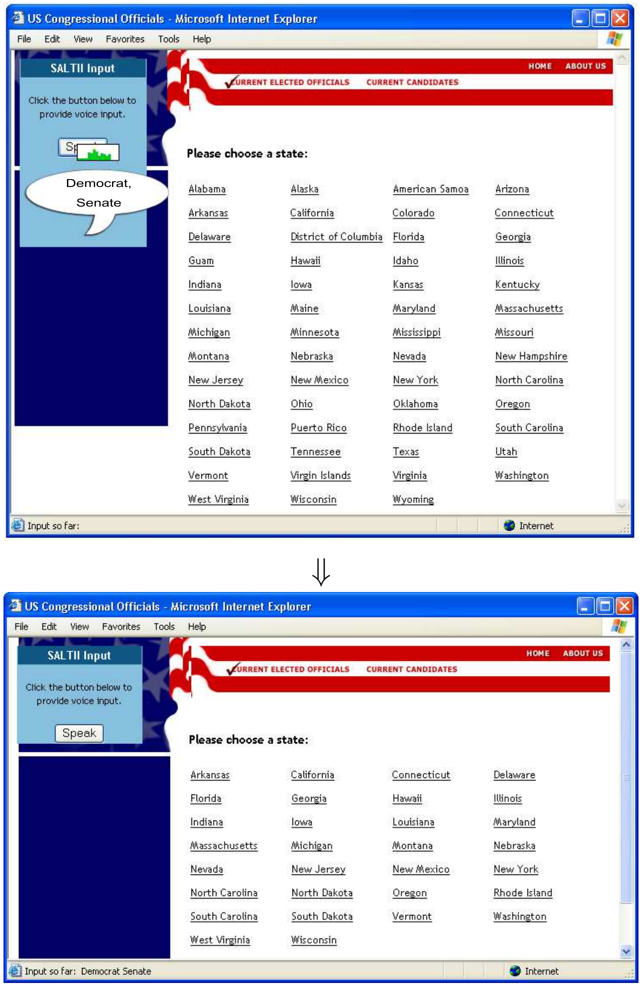
In order to maintain state, designers save continuations within these programs at certain points in the interaction. Stored continuations have several applications within the purview of state maintenance. For example, they can be used to maintain state when the user desires to explore alternate options by clicking the 'back button' or cloning a new browser window. In addition, if a user's interaction with a web resource is prematurely terminated due to a lost network connection, when the connection is restored, rather than starting the interaction from scratch, the user may resume a saved continuation.

3.4.5 PIPE: Personalization is Partial Evaluation

PIPE [Ram00] is a research project which employs representations of interaction similar to those discussed above for capturing and supporting information-seeking interactions. PIPE extends the script-based approach by *transforming* representations using program transformation techniques, to achieve personalized interaction. Similarly, it can be seen as extending the web continuation approach because rather than sequentially evaluating programs, PIPE non-sequentially evaluates programs.

Since I cover the details of the PIPE approach in the previous chapter and since representing and reasoning about interaction is the emphasis of this tier, rather than reinforcing the PIPE approach with additional, but similar, examples, such as those presented in the previous chapter, I appropriately illustrate additional advantages to modeling interaction programmatically here.

PIPE is cast as a modeling methodology for information personalization. However, it makes no commitments to a particular algorithm, format for information resources, type of information-seeking activities or, more fundamentally, the nature of personalization delivered. Instead, it emphasizes the modeling of an information space in a way where descriptions of information-seeking interactions can be represented as partial information. Such partial information is then exploited (in the model) by *partial evaluation*, a program specialization technique popular in the programming languages community [Jon96]. I model the information space as a program, partially evaluate it w.r.t. a partial assignment of its variables representing user input, and recreate the personalized information space from the specialized program to realize user-specified interactions. If interaction with an information system can be modeled programmatically and captured with partial information, then PIPE can personalize it.


US Congressional Officials - Microsoft Internet Explorer
File Edit View Favorites Tools Help
SALTII Input
Speak
Democrat, Senate
HOME
ABOUT US
CURRENT ELECTED OFFICIALS
CURRENT CANDIDATES
Internet
⇓

Click the button below to provide voice input.

Please choose a state:

Alabama Alaska American Samoa Arizona
Arkansas California Colorado Connecticut
Delaware District of Columbia Florida Georgia
Guam Hawaii Idaho Illinois
Indiana Iowa Kansas Kentucky
Louisiana Maine Maryland Massachusetts
Michigan Minnesota Mississippi Missouri
Montana Nebraska Nevada New Hampshire
New Jersey New Mexico New York North Carolina
North Dakota Ohio Oklahoma Oregon
Pennsylvania Puerto Rico Rhode Island South Carolina
South Dakota Tennessee Texas Utah
Vermont Virgin Islands Virginia Washington
West Virginia Wisconsin Wyoming

Input so far:

Click the button below to provide voice input.

Please choose a state:

Arkansas California Connecticut Delaware
Florida Georgia Hawaii Illinois
Indiana Iowa Louisiana Maryland
Massachusetts Michigan Montana Nebraska
Nevada New Jersey New Mexico New York
North Carolina North Dakota Oregon Rhode Island
South Carolina South Dakota Vermont Washington
West Virginia Wisconsin

Input so far: Democrat Senate

Figure 3.19: Communicating multiple terms (‘Democrat Senate’) in a single utterance.

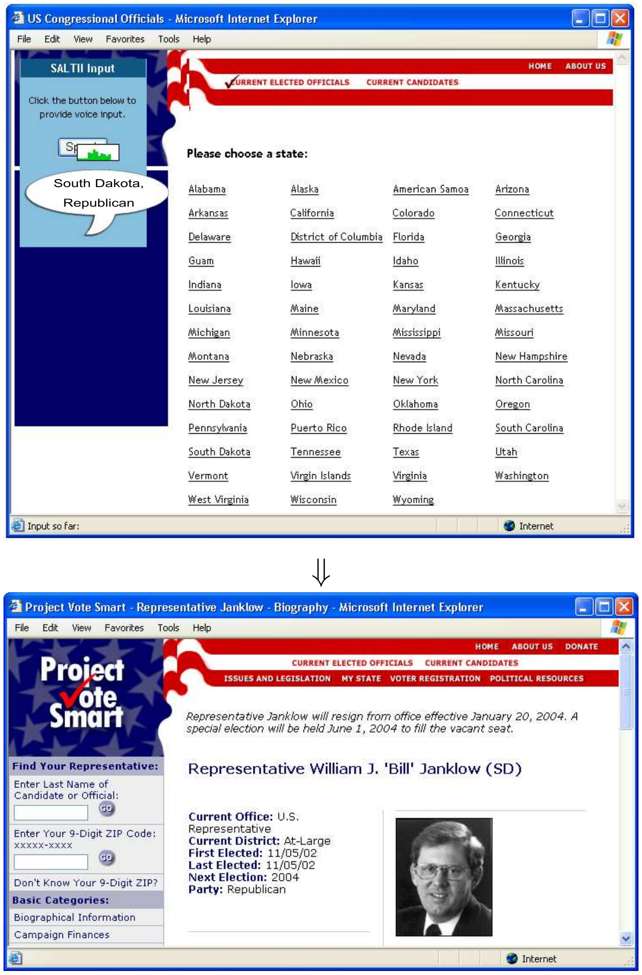
While in the previous chapter I used a toolbar interface to capture and communicate partial information, here, I illustrate the use of a voice interface called *SALTII* (SALT Interaction Interface; pronounced 'salty') to the same effect. I implemented SALTII using SALT 1.1 (Speech Application Language Tags) [SAL02] to support out-of-turn voice input. SALT is a standard which augments HTML with tags for speech input/output to create webpages which can talk and listen, rather than just passively display content. It employs SRGS (Speech Recognition Grammar Specification) to specify the underlying grammar. Such technologies, including X+V (XHTML plus Voice [ACL$^+$01]), are a playing a key role in the emergence of the speech-enabled web [Lai00]. SALTII enables multimodal interaction when used in conjunction with hyperlinks. SALTII is thus the analog of Extempore in a voice modality (see Fig. 3.19 for the Project Vote Smart (PVS) example revisited). However, while Extempore captures out-of-turn input as a bag of words (hence supporting phrases), currently the SALTII-enabled pages do not accommodate lengthy constructs (for ease of speech recognition). The SALTII and Extempore interaction interfaces are available for download from our project website at http://pipe.cs.vt.edu.

Modeling in PIPE

PIPE provides many modeling options, beyond the simple 'processing of partial inputs in nested conditionals' approach given in the previous chapter. For instance, I can exploit interesting dependencies underlying politicians' attributes in PVS. For instance, if the user says 'senior seat,' I can infer by functional dependency that he is referring to a member of the Senate and not the House. So, I can partially evaluate w.r.t. these additional variables. As another example, entering 'South Dakota' and 'Republican' in the current political landscape defines a unique member of Congress because South Dakota has only one Republican congressional official, so the interface does not need branch of Congress or seat/district information (see Fig. 3.20).

It is important to consider such facets to deliver a compelling personalized experience. The net effect of such considerations will be the initialization of multiple program variables based on the user's input, and the site created at every stage reflects an accurate summary of the remaining dialog options. These simple examples illustrate not only the importance, but also the power in forming an explicit representation of interaction as a basis for personalization.

In addition, a variety of other information spaces and corresponding information-seeking activities can be modeled in PIPE. Modeling options for representing information integration, abstracting within a webpage, interacting with recommender systems, modeling clickable maps, representing computed information, and capturing syntactic and semantic constraints pertaining to browsing hierarchies are described in [Ram00, RP01]. Opportunities to curtail the cost of partial evaluation for large sites also are described in [RP01]. I do not address such modeling aspects here except to say that the effectiveness of a PIPE implementation depends on the particular modeling choices made *within* the programmatic representation (akin to [Wil84]). I cannot overemphasize this aspect. I can deliver a higher degree of personalization in an example such as PVS by conducting a more sophisticated modeling of the underlying domain. For example, individual congressional official webpages at the leaves could be modeled by a deeper nesting of conditionals involving address, educa-



US Congressional Officials - Microsoft Internet Explorer
File Edit View Favorites Tools Help
SALTII Input
Sp
South Dakota, Republican
HOME ABOUT US
CURRENT ELECTED OFFICIALS CURRENT CANDIDATES
Alabama Alaska American Samoa Arizona
Arkansas California Colorado Connecticut
Delaware District of Columbia Florida Georgia
Guam Hawaii Idaho Illinois
Indiana Iowa Kansas Kentucky
Louisiana Maine Maryland Massachusetts
Michigan Minnesota Mississippi Missouri
Montana Nebraska Nevada New Hampshire
New Jersey New Mexico New York North Carolina
North Dakota Ohio Oklahoma Oregon
Pennsylvania Puerto Rico Rhode Island South Carolina
South Dakota Tennessee Texas Utah
Vermont Virgin Islands Virginia Washington
West Virginia Wisconsin Wyoming
Internet
⇓
Project Vote Smart - Representative Janklow - Biography - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Project Vote Smart
HOME ABOUT US DONATE
CURRENT ELECTED OFFICIALS CURRENT CANDIDATES
ISSUES AND LEGISLATION MY STATE VOTER REGISTRATION POLITICAL RESOURCES
Find Your Representative:
GO
GO
Don't Know Your 9-Digit ZIP?
Basic Categories:
Biographical Information
Campaign Finances
Internet

Click the button below to provide voice input.

Please choose a state:

Input so far:

Enter Last Name of Candidate or Official:

Enter Your 9-Digit ZIP Code: xxxxx-xxxx

Representative Janklow will resign from office effective January 20, 2004. A special election will be held June 1, 2004 to fill the vacant seat.

Representative William J. 'Bill' Janklow (SD)

Current Office: U.S. Representative
Current District: At-Large
First Elected: 11/05/02
Last Elected: 11/05/02
Next Election: 2004
Party: Republican

Figure 3.20: Automatic input expansion by functional dependency.

tion, and other attributes of the individual. In other words, a single page could be further modeled as a browsable hierarchy and 'attached' (functionally invoked) at various places in its programmatic rendition. Conversely, I can deliver a lesser degree of personalization in the PVS example by requiring categorical information along with user input. For instance, replacing `if (Democrat)` with `if (Party == Democrat)` implies that the specification of the type of input (namely that 'Democrat' refers to the 'name of a party') is required for the expression to be partially evaluated. Such a modeling would not help a consumer shopping for a Harry Potter book who does not know if 'Harry Potter' is the book's author or title. Personalization systems built with PIPE can thus be distinguished by what they model and the forms of personalization enabled by applying partial evaluation to such a modeling.

3.5 Making It Work: Systems Support and Enabling Technologies

I briefly mention some systems support technologies to bring personalization solutions into mainstream adoption and use.

3.5.1 Data Modeling

Researchers have identified data modeling as critical to the degree of personalization delivered [Chi97, RS97]. For personalization purposes, data modeling often involves databases techniques for the web [ABS00, FLM98]. I focus here on content modeling and information integration techniques, such as web crawling and wrapping.

Web Wrappers and Information Integration

The main motivation for wrappers is bridging the gap between the abundance of data on the web and applications which have no direct access to the web [ABS00, HGMC$^+$97]. WSQ [GW00] is an example of a system which can benefit from such modeling. The type of information extraction techniques employed are dependent on the type of personalization intended.

In template-based systems, a query typically drives the modeling process [AK97, KMA$^+$98]. Manually designing a web wrapper and subsequently maintaining it is a painstaking process due to dependence on the source format. Therefore, research has been conducted on automatically generating wrappers. Such programs exploit structural cues in data. Ashish and Knoblock take a regular expression, grammar-based, approach to wrapper generation [AK97]. An alternate approach [SA99] is to exploit intermediate mappings between system-defined formats and standard formats, such as XML and DOM. The project culminated in the World Wide Web Wrapper Factory (W4F) toolkit [SA99].

All of these projects are focused on answering queries and thus approach wrapper generation from a within-page modeling standpoint. Others take a broader approach and model site structure or mediate inter-site differences [KMA$^+$98]. Central to this approach is the flow of information within a site and across sites. In other words, information is integrated through data flow. The output of the first source is fed into the second source as input and so on. Such an approach can be contrasted to formalisms for information integration that use shared schemas and mediated queries [GMPQ$^+$97, GBMS99, KLSS95].

These approaches suffer from a pitfall endemic to all wrappers, whether automatically generated or not. Crawling or wrapping a third party website is error prone due to page irregularities, extensive use of stylish page formatting, and an abundance of semistructured data [ABS00]. While many wrapper and crawling packages are freely available on the web, such tools are difficult to use out-of-the-box and typically require a level of manual customization for a particular site. It is often useful to conduct a preliminary inspection of page design and site layout before implementing such systems. In addition, a variety of semantic issues exist for effective information integration which are currently handled with heuristics.

Several distinct solutions to this problem have emerged. One idea is on focus modeling to specific document structures. Rus and Subramanian concentrate on capturing and modeling tabular structures and thus employ document segmentation and structural detection algorithms [RS97]. XTRACT [GGR$^+$00], a system similar to [AK97], uses grammars and AI techniques to infer DTDs (Document Type Definitions) for XML data. The endurance of such approaches are tested by richer standards for document types such as XSchema (XML Schema)[Fal01]. It is widely believed that XSchema may render DTDs obsolete. An alternate approach to webpage modeling, which also employs AI techniques, is wrapper induction [KWD97]. Systems such as [GGR$^+$00, KWD97] scale well with regard to frequently changing sites due to the exploitation of machine learning techniques. Yet another solution uses program compaction techniques (e.g., greatest fixpoint semantics of monadic datalog programs) to mine schemas from semistructured data [NAM97, NAM98].

3.5.2 Requirements Gathering

Techniques discussed in this section address requirements gathering for personalization systems. This problem can be approached from two distinct angles. The first involves empirical and explicit requirements analysis techniques such as scenario-based design. An alternate approach involves weblog mining to implicitly capture requirements. Ultimately these techniques are directed toward closing the gap between the goals of a system designer and the task model of a user [MAB00].

Scenario-Based Methods

The techniques presented here are especially important with regard to representing and reasoning about interaction. Carroll and Rosson make an explicit science out of scenario-based design and claims analysis in [CR92] where they describe the 'task-artifact' methodology. The end-goal of this research, which lies at the intersection of human-computer interaction (HCI) and software engineering, is to develop an action science approach to HCI.

The first step in the methodology is to collect scenarios. Scenarios are narrative accounts of users performing tasks and can be generated empirically or analytically. Carroll and Rosson develop a classification of scenarios, or typology, which aids in analytical and empirical approaches to scenario collection.

The next step in the methodology is claims analysis. A claim is 'a specific psychological consequence of a system feature' [CR92]. While a scenario provides a narrative account, a claim provides a causal account. Claims analysis attempts to explain scenarios and must proceed in parallel with scenario generation. Scenarios and claims thus developed can be utilized by CBR as applied to script-directed information systems. For instance, they can be processed to yield cases and identify prototypical scripts.

This work also has important connections to requirements engineering in PIPE [RP01]. In particular, scenario-based design and claims analysis can be used to generate interaction sequences in a domain which lacks precise, explicit, and clear semantics. In existing systems, the task-artifact cycle can be used to characterize interaction sequences. This is particularly interesting in sites based on the metaphor of social network navigation. In such non-traditional information spaces, scenario-based design can be employed to either unroll unbounded interaction sequences to a manageable level or define personalization. The resulting scenarios (i.e., interaction sequences) would be invaluable to finding an appropriate programmatic design representation of interaction.

Rosson has researched the integration of task and object models [Ros99] in software design. To facilitate this goal, she proposes using object-oriented analysis and design of scenarios. Scenarios are helpful in identifying an initial set of software objects. Claims analysis of the scenarios identifies constraints and opportunities. This work has ties to PIPE as well. In PIPE, a user's personalized experience, analogous to the task model, closely resembles the system's programmatic model of interaction, analogous to the object model.

These connections between scenario-based design and PIPE are explored in [RRC01]. In addition, the authors discuss how explanation-based generalization (EBG) can be used to explain scenarios to provide a starting point for a personalization system. EBG is a machine learning technique which has strong ties to partial evaluation [Jon96]. Proof trees in EBG used in explaining scenarios resemble the goal trees used by Belkin *et al.* [BCST95].

An alternative approach to requirements gathering is metaphorical design [Mad94]. It is now well accepted that metaphors provide intuitive ways to think about interaction with information systems (e.g., the desktop). For instance, Wexelblat and Maes [WM99] explore the use of *footprints* as a navigation design metaphor.

Web Mining

The web has become fertile ground for what O'Leary calls 'AI Renaissance' [O'L97]. The use of collaborative filtering in recommender systems was one of the first attempts at conducting personalization. Collaborative filtering is difficult, since the the majority of web users are privacy conscious and dislike providing explicit feedback. When applying these techniques, care must be taken to ensure that privacy is not compromised.

Web log mining is an alternate approach to capturing user interest and has been referred to as 'observational personalization' [MAB00]. Web log mining is implicit, unobtrusive, and entails chartering the footprints left by visitors. One can analyze web logs to mine navigational patterns.

For instance, IndexFinder [PE00] mines patterns to guide a non-destructive and transformation approach to website adaptation. Non-destructive adaptations are those that add structure, pages to sites, or links to pages, but do not destroy structure or otherwise remove information from a site. IndexFinder identifies co-occurring page visits and recommends candidate index pages to the web master. Thus, IndexFinder is a semi-automatic approach.

Web navigation patterns are sought to evaluate website usability as well [Spi00]. The focus here is on avoiding costly and error prone formative usability evaluations. The miner is looking for sequences of frequently visited pages and routes connecting pages frequently accessed together. Two popular weblog mining software systems are MiDas and Web Utilization Miner (WUM) [Spi00]. Another project which has user modeling goals is discussed in [MCS00]. Here, weblog mining helps form associations which are used in a collaborative filtering style to aid a recommendation engine.

While web mining is data-driven and therefore heuristic at best, it is inexpensive and can be applied more frequently than its manual counterparts discussed above. The projects described here show that mining access logs is a feasible approach to gathering requirements for personalization. This approach however suffers from problems of coordination and ethics. Therefore, social and operational issues need to be addressed to make such techniques practical and more appealing.

3.5.3 Transformation Algorithms

XML has evolved from simple text markup for data interchange to a mature technology with a rich suite of associated tools. The eXtensible Stylesheet Language for Transformations (XSLT) [Cha99] describes transformations from XML to XML, including XHTML and VoiceXML, and various other formats including, plain text and HTML. An XSLT transformation is specified in a stylesheet containing pattern-action rules. These rules are recursively applied, starting from the root of a tree-structured XML document. Whenever a pattern match is encountered, the associated actions are executed. XSLT is essentially a vehicle to implement graph transformations [ET99]. Therefore, the transformation capabilities of XSLT can be used to implement partial evaluation, among other operations, and to create a robust and easily maintainable personalization application, based on the ideas illustrated above.

Specifically, if an XML document models interaction with a website, then an XSLT stylesheet representing partial information (i.e., a user request) can be matched against XML tag names to personalize interaction. Since XSLT can transform XML into multiple output formats, transforming the model of interaction into a browsable website (i.e., HTML) naturally follows. XSLT thus unifies the processes required to conduct personalization into a single, mature, robust, and well-accepted technology. Details regarding the use of these new and emerging W3C standards are presented in [CDA00]. I have programmed the underlying transformation engine of various PIPE applications, including the Project Vote Smart congressional application [PR03b], with XSLT.

3.5.4 Delivery Mechanisms and Intermediaries

Intermediaries, which are ‘programs or agents that meaningfully transform information as it flows for one computer to another,’ [MB00] are critical to the success of personalization applications on the web. Examples of intermediaries are portals, proxies, and transcoders. IBM’s WBI [MB00] provides a programming model for intermediaries akin to PIPE’s contribution of a programming model for personalized interaction.

3.6 Niche Domains

3.6.1 Adaptive Hypermedia

Over the past 15 years, hypermedia has been extended to support personalization capabilities (e.g., the adaptive web [BM02]). Adaptive hypermedia lies at the intersection of hypermedia and user modeling [Bru01]. Hypermedia services, such as educational and online-help systems, have been most impacted by personalization research.

Links in adaptive hypermedia systems are dynamic, leading to different destinations for different users. The techniques employed include direct guidance, adaptive link sorting, hiding, annotation, generation, and map adaptation. In addition to navigational adaptations, such applications modify the aesthetics of presentation to direct a user. I refer the interested reader to [Bru96] and its sequel [Bru01] for a comprehensive survey of methods and techniques of adaptive hypermedia systems and to [BBH99] for a succinct introduction. Examples of browsing-oriented adaptive hypermedia systems are 'Syskill & Webert' [PMB96] and WebWatcher [JFM97].

3.6.2 Mobile Environments

Mobile arenas, which host the fastest growing segment of web users, are plagued with low bandwidth networks, thin clients, and information appliances [Ber00]. Furthermore, ubiquity is enriched and propelled by wireless portals, avatars [AR02, LFW01], and information kiosks [MBG$^+$01]. As these devices become commonplace, transcoding the information they present will not only become a necessity, but also vital to their widespread use and success [BBE$^+$02, Pan01]. Therefore, the use of personalization technology here extends past aesthetics. It is rather a requirement. To introduce this application domain, I present the following two representative projects.

Proteus

Proteus [ADW01] is a mobile personalization system developed at the University of Washington. The goal of the system is to both transcode and personalize web content based on mobile devices. To achieve the first goal, the designers segment webpages into screens using a probabilistic model. To achieve the goal of personalization, the designers collect training data from desktop computer usage to build user models.

Proteus supports both destructive and constructive within-page adaptations and implements three transformation operators – elide-content, swap-siblings, and add-shortcut. Creating a new webpage or adding new links between existing pages is not supported. The system can be contrasted to other adaptive systems such as IndexFinder [PE00]. While IndexFinder provides only non-destructive adaptation targeted by topic to all site visitors, Proteus is destructive as well and provides customization per individual. In addition, Proteus's user models are richer than those that result from weblog mining, which are essentially limited to navigational usage patterns.

W^3IQ

W^3IQ [JPK98] aims to provide asynchronous mobile access to the web. The designers explore collaborative information retrieval techniques to minimize resource use and information overload. W^3IQ employs intermediaries, such as proxy filters and cache servers, to facilitate disconnected browsing. In addition, it supports three types of transaction-like operations, which save state and are thus tolerant to disconnection.

3.6.3 Voice Interfaces and Multimodal Interaction

Speech and dialog-based systems, which afford mixed-initiative interaction [HM97], provide ripe domains for personalization. Zadrozny *et al.* state that natural language is 'a compelling enabling technology for personalization' [ZBC$^+$00] and that mixed initiative dialog is a form of personalization. Voice applications (e.g., voice portals[1]) and associated tools (e.g., VoiceXML) have collectively spawned the voice web [SB02]. Furthermore, this domain demonstrates how researchers in qualitatively different areas can work unconsciously on the same problem. I survey such connections below.

Sisl (Several interfaces, single logic) [BCD$^+$00], a primarily speech-based system, aims to minimize dialog constraints to provide extensive flexibility to users. Thus, the motivations of Sisl are commensurate with those of PIPE. Furthermore, the authors of Sisl recognize the idea of engaging a system in a two-way dialog as a means to provide personalization.

Sisl however takes a broader approach to personalization and supports multiple interfaces, error recovery, reversion, partial input, and partial orderings on specification aspects in dialog. In contrast to PIPE, Sisl adopts an event-based approach. The designers model application logic by event handling (reactive) mechanisms. The specification aspects of PIPE are called events in Sisl.

Sisl makes a distinction between partial orderings and partial information. Partial information is incomplete in that all specification aspects required to complete a dialog or information-seeking activity are communicated incrementally. Partial orders, on the other hand, permit aspects to arrive in different orders. Furthermore, Sisl makes a distinction between *out-of-turn* aspects and *unsolicited* aspects. PIPE traditionally does not make this distinction, because its support mechanism, partial evaluation, handles both uniformly.

Both Sisl and PIPE rely on the assumption that a representation of default order execution exists (e.g., a script). This representation involves anticipation in both approaches. PIPE eagerly (partially) evaluates that representation with respect to specification aspects, which may arrive in any order, to implement partial orderings in dialogs. Sisl, on the other hand, lazily evaluates aspects. In other words, when Sisl receives an aspect out-of-turn, which violates its representation, it logs that aspect in a queue. The system retrieves that aspect when the default order of execution solicits it later. At that time, the event is enabled and added to the activated set. The designers refer to this process, which handles unsolicited events and thus minimizes anticipation, as lookahead.

A closer connection between PIPE and speech-based systems is made in [RCPQ02] where the form interpretation algorithm of VoiceXML [MBD$^+$01] is shown to be a partial evaluator in disguise.

---

[1] Examples are Tellme (tellme.com) and BeVocal (bevocal.com).

3.7 Discussion

I have presented an overview of personalization systems according to the interaction they afford. As personalization systems become prevalent, the need to engage the user in compelling interactions will become more important.

Several factors lead us to be optimistic about the future of personalization as an academic discipline. For instance, the widespread use of physical computing devices, location-aware systems, and embedded Internet appliances means that personalization will transcend current delivery mechanisms. Such domains pose interesting problems that will continue to challenge our assumptions about personalization. Users create context in physical situations that can be stored and retrieved for use in electronic access paradigms. Thinking about how information access works in such multimodal settings will lead to a theory of human-information interaction, as espoused in [TMK$^+$02].

Road Map

In the following chapter, I develop and formalize the program transformation approach to out-of-turn interaction. The chapter includes several interpretations for out-of-turn interaction, in addition to that used in the previous chapter. It also formally details and generalizes the properties of websites, such as the levelwise structure of the websites presented in Chapter 2, the mutually-exclusive nature of the hyperlinks on each of their webpages, and the dependencies between their facets.

Chapter 4

Formalizing Out-of-turn Interaction

> "The important thing in science is not so much to obtain new facts as to discover new ways of thinking about them."
>
> Sir William Lawrence Bragg, the youngest-ever recipient of the Nobel Prize.

In this chapter, I formalize (out-of-turn) interaction with websites. This chapter places everything we have seen so far in a rigorous mathematical context.

4.1 Research Theme

The central theme of this thesis is to pose interactive information-seeking as the application of a program transformation technique to a programmatic representation of interaction based on partial user input.

Representation × Transformation × Partial Input ⇒ Interaction Paradigm

The creativity in my research (see Fig. 4.1) arises from relating concepts in the web domain (e.g., sites, interactions) to notions in the program-theoretic domain (e.g., programs, program transformations). An additional opportunity for creativity in my work arises from varying the two terms on the l.h.s. for which I have control, the (Representation, Transformation) pair (later formalized as a *model*), to achieve the desired form of personalization. The predominate form of personalization discussed in this thesis is out-of-turn interaction.

web:	(website × interaction technique)	×	user input	⇒	personalized website
⋮	⋮		⋮		⋮
⋮	⋮		⋮		⋮
program-theoretic:	(representation × transformation)	×	partial input	⇒	interaction paradigm

Figure 4.1: The connection between the web and program-theoretic domains.

4.1.1 Objectives of this Chapter

The objectives of this chapter are to:

1. Develop terms and tools to build up to graph-theoretic interpretations of out-of-turn interaction with a general class of websites.
2. Illustrate how these interpretations can be supported on a programmatic landscape by a (programmatic representation, transformation technique) pair (called a model), often involving *program slicing*, a general class of program transformations.
3. Evaluate the soundness and completeness (as well as other properties) of a model w.r.t. its intended interaction paradigm (e.g., browsing or out-of-turn interaction).
4. Identify a partial order of classes of hierarchical hypermedia and explain its implications on out-of-turn interaction.
5. Develop support terms and tools to automatically identify particular instances of these classes.
6. Describe how several types of functional dependencies in websites can be mined and used to expand partial input from the user, to either further customize the experience or deliver the experience.
7. Illustrate program transformation techniques based on program slicing to mine functional dependencies.
8. Show that an alternate program transformation technique, which employs the functional dependencies above, can achieve the same effect as the originally developed interpretation of out-of-turn interaction. This highlights the duality in uses of program slicing for personalized interaction.
9. Develop specialized program transformation techniques for some specific classes of hierarchical hypermedia.

4.1.2 Methodology

When a user says something out-of-turn, I ask the question 'What can I reasonably prune out of the website? Answers to this question lead to interpretations for out-of-turn interaction. I have developed the following two interpretations, described using web vocabulary:

1. When a user says something out-of-turn,
   (a) first find leaf webpages reachable by a path involving a hyperlink labeled with the out-of-turn input; and
   (b) prune all paths through the site that *do not* lead to any of these leaf pages.
2. When a user says something out-of-turn prune all paths through the site which *do not* involve a hyperlink labeled with the out-of-turn input.

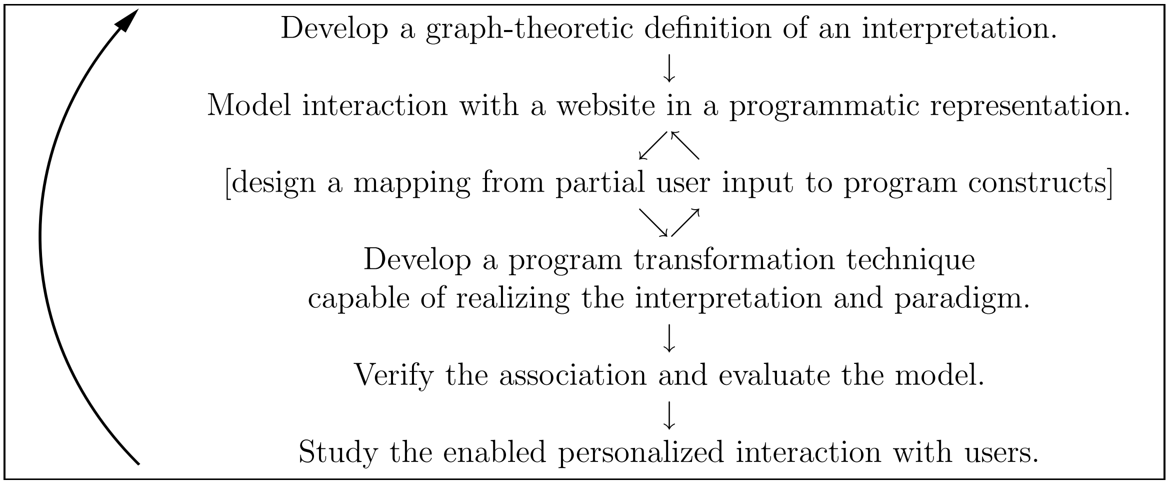

Develop a graph-theoretic definition of an interpretation.
Model interaction with a website in a programmatic representation.
[design a mapping from partial user input to program constructs]
Develop a program transformation technique capable of realizing the interpretation and paradigm.
Verify the association and evaluate the model.
Study the enabled personalized interaction with users.

Table 4.1: My research methodology.

Notice that both these interpretations make the in-vocabulary assumption. An interaction using ‘illegal’ input may either be undefined or cause an error. Interpretation 2 entails interpretation 1 because every path retained by interpretation 2 is retained under interpretation 1. In other words, interpretation 2 prunes all paths pruned under interpretation 1, but the converse does not hold. The next question I ask is ‘How can I support (model and realize) the interpretation of out-of-turn interaction in a program-theoretic domain?’ Answering this question involves:

1. Developing a graph-theoretic definition of the interpretation.
2. Modeling interaction with the underlying site in an explicit programmatic representation.
3. Identifying a set of program transformations and developing a technique involving a composition of them to relate to the interpretation.
4. Designing a mapping from partial user input to program constructs to capture requirements.

This suggests an iterative process (see Table 4.1) of developing a programmatic model of interaction with an instance of hierarchical hypermedia and identifying (compositions of) program transformations, capable of capturing and supporting the desired interactions from such a model. As importantly, I must design a mapping from user requests (partial input) to program constructs which the particular transformations are capable of accepting as input and exploiting.

I next evaluate the model by computing several metrics. I also experiment with the personalized interaction enabled by conducting studies with users. Studies often reveal insights into new interpretations, interaction paradigms, and interaction techniques, and thus, help close the loop. Studies with users are the subject of the following chapter.

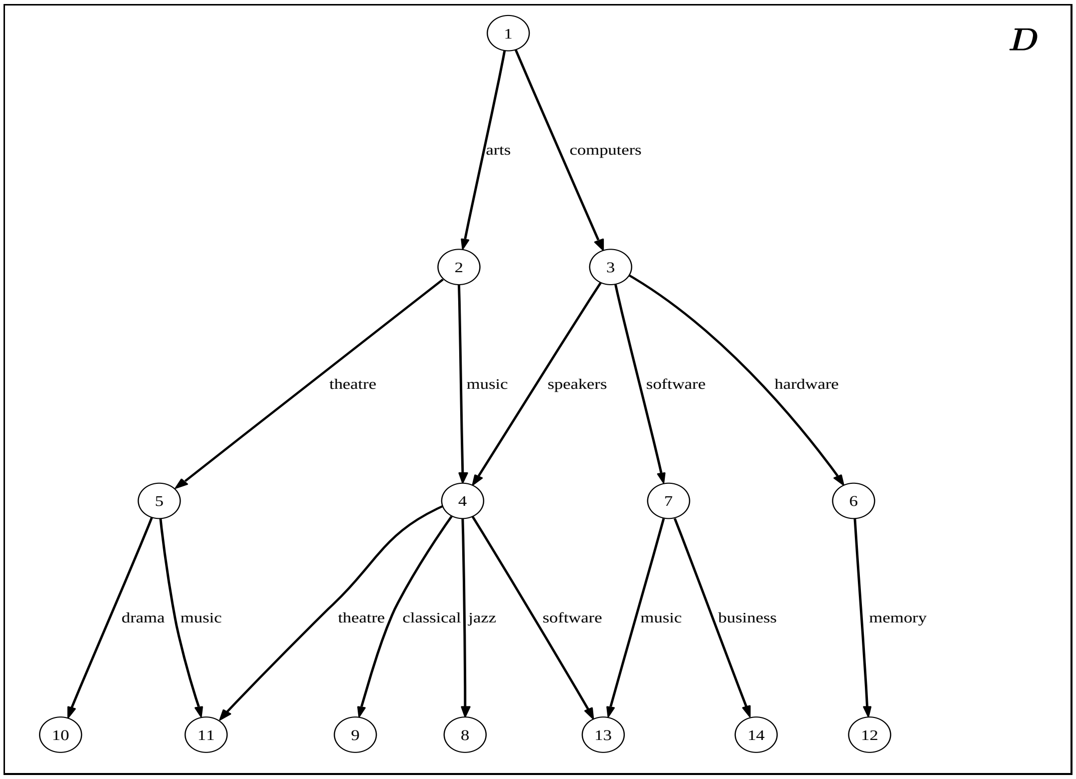


Figure 4.2: Example of a DAG model of a hypothetical hierarchical web directory with characteristics similar to those in Yahoo!.

4.2 Graph-theoretic View of Out-of-turn Interaction

I begin by developing syntactic notions from graph theory and progressively attach web interaction semantics to develop a theory of representing and reasoning about interaction with hierarchical hypermedia.

4.2.1 Syntactic and Semantic Notions

A *directed, acyclic, edge-labeled graph* (DAG) is a directed, connected graph with no cycles, where each edge in the graph is labeled and where a vertex never participates as a source in more than one edge with the same label. I use the variable D to represent a DAG. One vertex of D is distinguished as the root. In addition, a 'parenthood' relation places a hierarchy

Graph-theoretic construct	**Web analog**
Graph	Website
Vertex	Webpage
Edge	Hyperlink
Edge-label	Hyperlink label
Root	Homepage

Table 4.2: A list of graph-theoretic constructs and their web analogs.

partial order on the set of vertices of D. This representation of a hierarchical website is similar to the popular model of semistructured data: a rooted, edge-labeled graph [ABS00]. Fig. 4.2 illustrates a DAG model of a synthetic, hierarchical website possessing characteristics similar to human-compiled taxonomies of links to web resources, such as Yahoo! or the Google Directory at Google.com/dirhp. Table 4.2 is an abridged mapping from graph-theoretic notions to their web analogs.

Edges, which represent structural links [All95], help model paths through a website a user follows to access leaf vertices. Leaf vertices model leaf webpages which contain content, although this content is not explicitly modeled in D. I refer to a leaf content page as *terminal information* and the terms therein as *units of terminal information*. Assume non-leaf vertices do not contain content. Edge-labels, which I refer to as *structural information*, model hyperlink labels or, in other words, choices made by a navigator en route to a leaf. An edge-label, a unit of structural information, is therefore a *term of information-seeking* (or simply a *term*) which a user may bring to bear upon her information seeking. Structural information thus helps make distinctions among terminal information.

A set of units of structural information is *complete* when it determines a particular leaf webpage; otherwise it is *partial*. An *interaction set* of D is the complete set of the terms along a path from the root of D to a leaf vertex of D. An interaction set constitutes complete information; any proper subset of it is partial information. I say an interaction set of D *classifies* a leaf vertex of D. Notice that an interaction set does not capture any order of the terms, according to my definition.

I now provide some definitions that pertain to a user's interaction with a website. A term is *in-turn* information if it appears as a hyperlink label on the user's current webpage (i.e., currently solicited by the system). The label of each edge whose source is the root of D is thus in-turn information. On the other hand, a term is *out-of-turn* information if it models a hyperlink label nested somewhere deeper in the site (i.e., currently unsolicited from the system, but relevant to information seeking). Each term from D which is not in-turn information is out-of-turn information.

Several partial orders can be defined over an interaction set w.r.t. the time at which the user communicates the term to the system, called *arrival time*. When a user clicks on a hyperlink, she implicitly communicates the information modeled by the hyperlink label to the system. For instance, when a user clicks on the hyperlink labeled 'arts' followed by that labeled 'music,' she communicates the $\prec$arts, music$\succ$ terms, in that order. Similarly, when the user speaks out-of-turn he is communicating terms to the system. These partial orders can be summarized in the form of partially ordered sets or posets. Each linear extension of such a poset is a total order called an *interaction sequence*. A *browsing interaction sequence* of D is a total order on an interaction set of D w.r.t. the 'parenthood' relation of D. An *out-of-turn interaction sequence* of D is a total order on an interaction set of D w.r.t. the arrival time relation implied by out-of-turn interaction. Interestingly, both interpretations of out-of-turn interaction introduced above imply the same arrival time relation. This arrival time relation is a partial order containing only the reflexive tuples of terms from the interaction set. In other words, none of the terms from the interaction set are required to be ordered. The linear extensions of the posets associated with these partial orders are out-of-turn interaction sequences.

An *interaction paradigm* $\mathcal{P}$ for D can be given by the union of all linear extensions of posets defined over interaction sets of D. In other words, an interaction paradigm is a complete set of realizable interaction sequences from D w.r.t. an interaction technique. When an edge-label labels more than one edge in a path from the root of D to a leaf vertex of D, it is advantageous to think of an interaction sequence as a finite effective enumeration of an interaction set of D, where the order of the terms in the enumeration corresponds to the arrival time relation afforded by the interaction technique w.r.t. D.

The following is the browsing paradigm of D in Fig. 4.2.

{$\prec$arts, music, jazz$\succ$,
$\prec$arts, music, classical$\succ$,
$\prec$arts, music, theatre$\succ$,
. . . ,
$\prec$computers, hardware, memory$\succ$}

Likewise, the following describes *an* out-of-turn paradigm:

{$\prec$arts, music, jazz$\succ$,
$\prec$music, arts, jazz$\succ$,
[the remaining 4 permutations of {arts, music, jazz}],
$\prec$arts, music, classical$\succ$,
$\prec$music, arts, classical$\succ$,
[the remaining 4 permutations of {arts, music, classical}],
. . . ,
$\prec$computers, hardware, memory$\succ$,
$\prec$hardware, computers, memory$\succ$,
[the remaining 4 permutations of {computers, hardware, memory}]}

Notice that while there can be only one browsing paradigm, there are multiple out-of-turn paradigms. In addition, the browsing paradigm for D is typically a subset of an out-of-turn paradigm.

4.2.2 Support Terms and Tools

I use the symbol $\mathcal{D}$ to represent the universal set of DAGs, and the symbol $\mathcal{T}$ to represent the universal set of terms. Similarly, I use $\mathcal{L}$ to denote the universal set of leaf webpages. Before I can expand this discussion to functions over $\mathcal{D} \times \mathcal{T}$ to realize the sequences of a particular interaction paradigm, I must develop some support terms and tools. In the following, edge-labels are primarily addressed as terms.

Sequencize is a total function $SQ : \mathcal{D} \rightarrow P\ (\mathcal{I})$ which given D returns the complete set of browsing interaction sequences in D (and thus the browsing paradigm of D). I use the symbol $\mathcal{I}$ here to represent the universal set of interaction sequences. $P\ (\cdot)$ denotes the power set function.

Term extraction is a total function $TE : \mathcal{D} \rightarrow P\ (\mathcal{T})$ which given D returns a set of all terms in D, i.e., with duplicates removed. A *term-co-occurrence set* of D is a set T s.t. $T \subseteq TE\ (D)$. Notice that this definition covers any set of terms from D. The use of the 'co-occurrence' phrase will be motivated later. Let the *level* of an edge-label in D be the depth

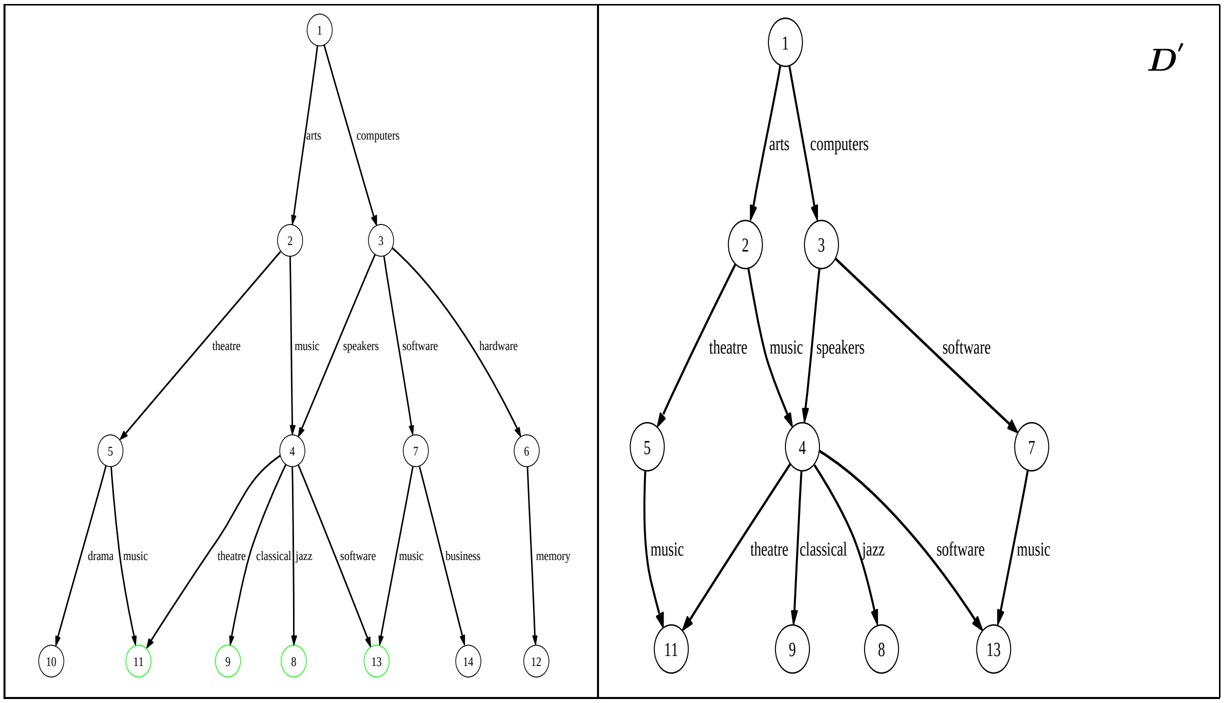


Figure 4.3: Illustration of forward-propagation (FP) and back-propagation (BP) on the DAG in Fig. 4.2. (left) Forward-propagation w.r.t. the term 'music': $FP\ (D,\ \text{music})$. (right) Back-propagation w.r.t. the leaf vertices highlighted green in left: $BP\ (D,\ FP\ (D,\ \text{music}))$.

of the source vertex of the edge it labels. If a given edge-label occurs multiple times in D, a level is associated with every occurrence. A *term-level set* of D is a term-co-occurrence set comprising *all* unique terms in D with the same level. *Term-level extraction* is a total function $TLE : (\mathcal{D} \times \mathcal{N}) \rightarrow P\ (TE\ (D))$ which given D and a level $l\ (\geqslant 1) \in \mathcal{N} = \{1, 2, \ldots, M\}$ returns the set of all unique terms in D with level l (i.e., a term-level set). I use the variable M to represent the the maximum depth of D. If D represents the DAG in Fig. 4.2, $TLE\ (D,\ 2) = \{\text{theatre, music, speakers, software, hardware}\}$. In any DAG, $TLE\ (D,\ 1)$ returns the set of terms available to supply via browsing.

Get sequences is a partial function $GS : (\mathcal{T} \times P\ (\mathcal{I})) \rightarrow P\ (\mathcal{I})_{\perp}$ which given a term t and a set of interaction sequences I_S returns the set of all interaction sequences in I_S each of which contain t as a member. The notation $P\ (\mathcal{I})_{\perp}$ denotes the partial nature of the function GS, i.e., the value of GS is undefined for some inputs. *Forward propagate* is a total function $FP : (\mathcal{D} \times T) \rightarrow \mathcal{L}$ which given D and a term $t \in T = TE\ (D)$ returns a set of leaf vertices L of D, where L contains each leaf vertex reachable from all paths of D containing an edge labeled by t. *Collect results* is a total function $CR : \mathcal{D} \rightarrow \mathcal{L}$ which given D returns a set of all the leaf vertices in D. Collect results w.r.t. the sub-DAG rooted at vertex 2 in Fig. 4.2 is the $\{8, 9, 10, 11, 13\}$ set of vertices. *Back propagate* is a total function $BP : (\mathcal{D} \times \mathcal{L}) \rightarrow \mathcal{D}$ which given D and a set of leaf vertices L returns a DAG D', where D' contains only interaction sequences which classify the leaf vertices in L. Fig. 4.3 illustrates forward-propagation (left) on the DAG D in Fig. 4.2 followed by back-propagation (right) yielding D'.

Notice that D also can be represented as a $|TE\ (D)| \times |CR\ (D)|$ term-document matrix, where the rows correspond to terms (structural information, or edge-labels) and the columns

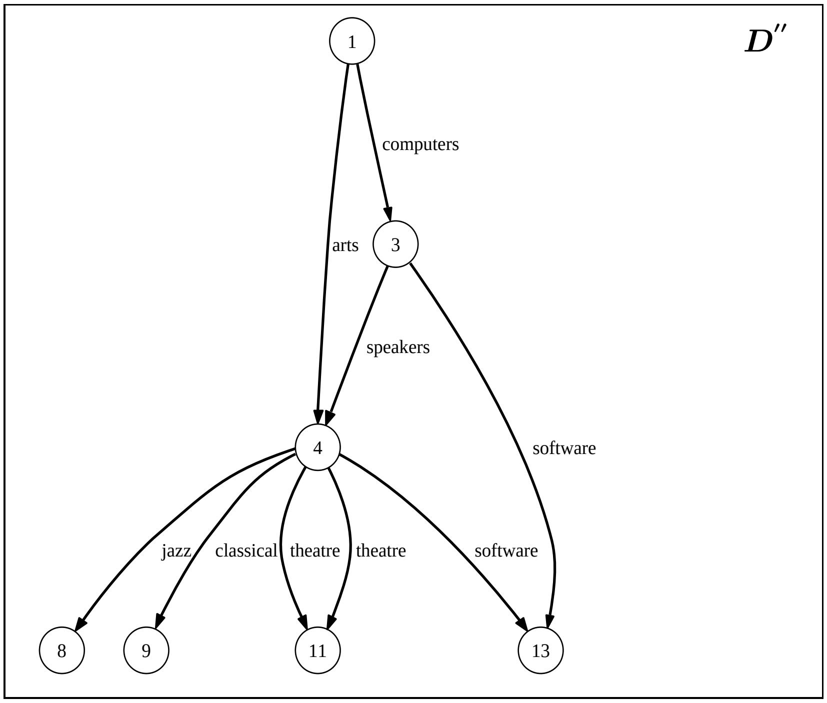


Figure 4.4: Results of interpretation 1 of out-of-turn interaction with the DAG D shown in Fig. 4.2 w.r.t. the term 'music': OOT_1 (D, music). Alternatively, one can think of this DAG as the result of shrink edges with the DAG D' in Fig. 4.3 (right), i.e., SE (D', music). Notice that this graph is no longer a DAG, according to my definition, since vertex 4 is the source of two edges with the same label and target. I defer addressing this issue until later when it will be better motivated.

correspond to webpages (terminal information, or leaf vertices). However observe that such a matrix is insufficient to reconstruct D.

4.2.3 Interpretations of Out-of-turn Interaction

Prior to providing graph-theoretic interpretations for out-of-turn interaction, I formalize browsing over DAG models of websites. *Browse* is a partial function $B : (\mathcal{D} \times \mathcal{T}) \rightarrow \mathcal{D}_\perp$, which given D and a term $t \in TLE\ (D,\ 1)$ returns the sub-DAG rooted at the target vertex of the edge in D labeled with edge-label t whose source vertex is the root of D. If the DAG in Fig. 4.2 is D, B (D, computers) returns the sub-DAG rooted at vertex 3, which would represent the result of a user clicking on the hyperlink labeled 'computers.'

I formally cast the above two interpretations for out-of-turn interaction in graph-theoretic terms as follows:

Interpretation 1 of out-of-turn interaction is a partial function $OOT_1 : (\mathcal{D} \times \mathcal{T}) \rightarrow \mathcal{D}_\perp$ which given D and a term $t \in TE\ (D)$ returns D'. It is defined as

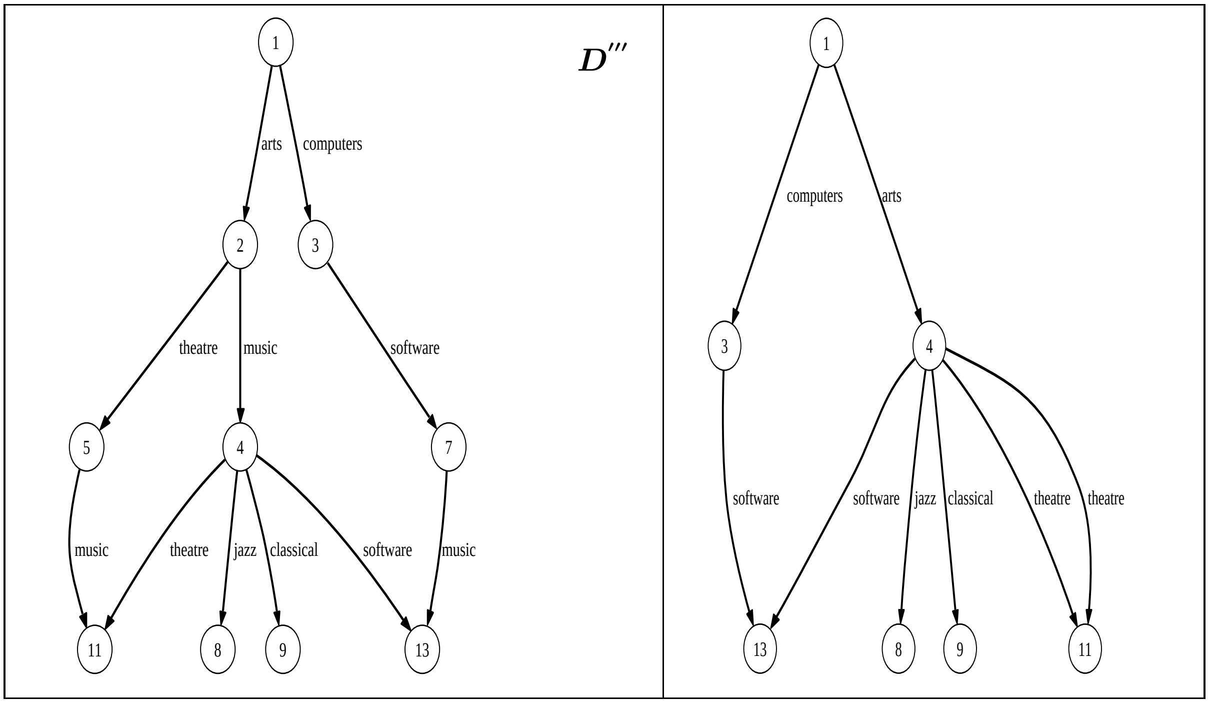


Figure 4.5: (left) Results of applying select paths to the DAG D' shown in Fig. 4.3 (right) w.r.t. the term 'music': SP (D', music). (right) Results of interpretation 2 of out-of-turn interaction with the DAG D shown in Fig. 4.2 w.r.t. the term 'music': OOT_2 (D, music). Alternatively, one can think of this DAG as the result of shrink edges with the DAG D''' (left): SE (D''', music).

$$OOT_1\ (D,\ t) = SE\ \overbrace{(\underbrace{BP\ (D,\ \overbrace{FP\ (D,\ t)}^{\text{Fig. 4.3 (left)}})}_{\text{Fig. 4.3 (right)}},\ t)}^{\text{Fig. 4.4}}, \tag{4.1}$$

where SE (*Shrink edges*) is a partial function $SE : (\mathcal{D} \times \mathcal{T}) \rightarrow \mathcal{D}_{\perp}$ which given D and a term $t \in TE\ (D)$ returns $D^{'}$, where any edge e in D labeled with t is removed in $D^{'}$, the source v_s of e is replaced with its target v_t in $D^{'}$, and v_t becomes the new target of any edge $e^{'}$ which v_s participates in as a target in $D^{'}$.

Interpretation 2 of out-of-turn interaction is a partial function $OOT_2 : (\mathcal{D} \times \mathcal{T}) \rightarrow \mathcal{D}_{\perp}$ which given D and a term $t \in TE\ (D)$ returns $D^{'}$. It is defined as

$$OOT_2\ (D,\ t) = \overbrace{SE\ (\underbrace{SP\ (BP\ (D,\ FP\ (D,\ t)),\ t)}_{\text{Fig. 4.5 (left)}})}^{\text{Fig. 4.5 (right)}}, \tag{4.2}$$

where SP (*Select paths*) is a partial function $SP : (\mathcal{D} \times T) \rightarrow \mathcal{D}_{\perp}$ which given D and a term $t \in TE\ (D)$ returns $D^{'}$, where $D^{'}$ contains only the interaction sequences from D involving t (i.e., $SQ\ (D^{'}) = GS\ (t,\ SQ\ (D))$). All interaction sequences pruned under OOT_1 also are pruned under OOT_2, but the converse does not hold. Formally, I state $SQ\ (OOT_2\ (D,\ t)) \subseteq SQ\ (OOT_1\ (D,\ t))$. Interpretation 2 retains shrunken versions of only interaction sequences involving the out-of-turn input, while interpretation 1 retains shrunken versions of all interaction sequences which classify the leaf vertices classified by sequences involving the out-of-turn input.

The results of FP are a set of leaf vertices classified by the interaction sequences involving the out-of-turn input. I use this set of leaves to back-propagate up to the root of the DAG via BP. To generalize my approach, I can replace FP with any function that returns a set of leaf vertices: SL (*Select leaves*). This can be any total function $SL : (\mathcal{D} \times \mathcal{T}) \rightarrow \mathcal{L}$ which given D and a term from D returns a set of leaf vertices of D. FP is an instance of SL. My use of a function such as SL allows us to combine my approach with standard techniques from information retrieval. Moreover, my inclusion of a function which returns a set of leaf vertices leads to the possibility of bringing units of terminal information (additional terms modeled in the leaf documents and not explicitly used in the classification), in replacement of, or in addition to, structural information, to bear upon the interaction. For instance, we might perform a query (e.g., 'Picasso') in a vector-space model over the set of leaf webpages (documents) using cosine similarity to arrive at a target set of leaves to back-propagate. However, I do not study this extension in this dissertation and present SL primarily as a technique that works with structural information.

Generalized interpretation 1 of out-of-turn interaction is a partial function $GOOT_1 : (\mathcal{D} \times \mathcal{T}) \rightarrow \mathcal{D}_{\perp}$ which given D and a term $t \in TE\ (D)$ returns $D^{'}$. It is defined as

$$GOOT_1\ (D,\ t) = SE\ (BP\ (D,\ SL\ (D,\ t)),\ t). \tag{4.3}$$

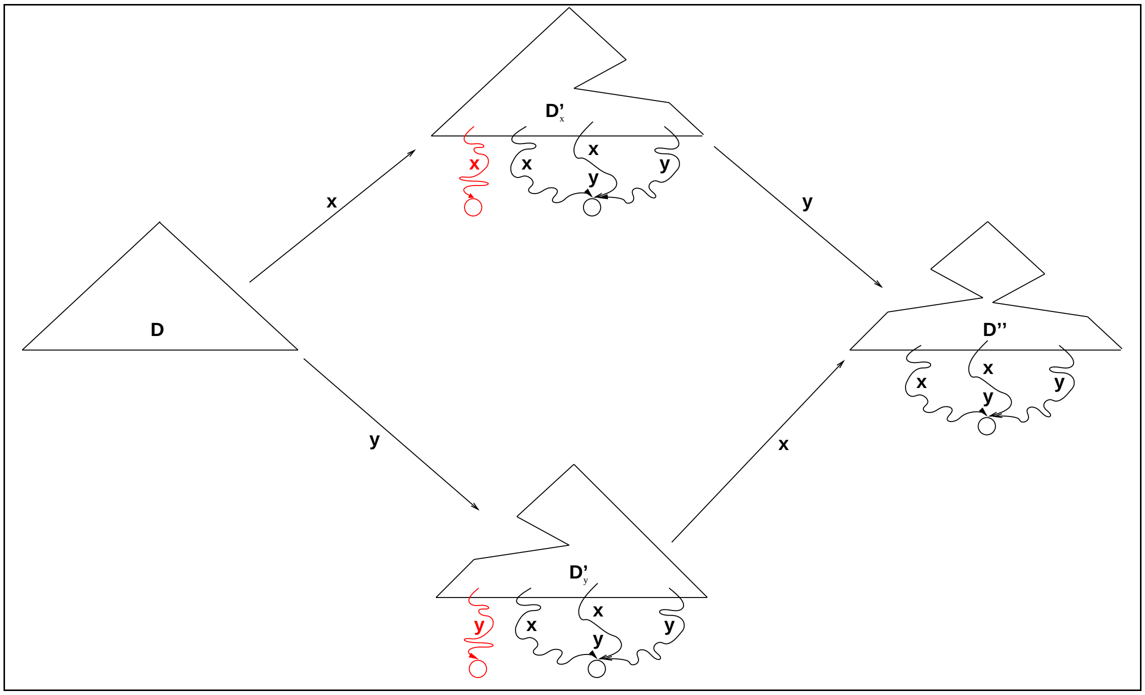


Figure 4.6: Schematic of proof for Lemma 1.

Generalized interpretation 2 of out-of-turn interaction is a partial function $GOOT_2 : (\mathcal{D} \times \mathcal{T}) \rightarrow \mathcal{D}_\perp$ which given D and a term $t \in TE\ (D)$ returns $D^{'}$. It is defined as

$$GOOT_2\ (D,\ t) = SE\ (SP\ (BP\ (D,\ SL\ (D,\ t)),\ t),\ t) \tag{4.4}$$

Notice that equations 4.3 and 4.4 are analogous to 4.1 and 4.2, respectively, in that FP in the later is replaced by SL. Note also that the out-of-turn paradigm studied in this thesis is supported by all four interpretations of out-of-turn interaction.

I define *Size of Out-of-turn Paradigm* as a total function $SoOP : \mathcal{D} \rightarrow \mathbb{N}$, which given D returns the size of its out-of-turn interaction paradigm. I can define $SoOP$ as follows:

$$|\mathcal{P}| \ = \ SoOP\ (D) \ = \sum_{I_i \ \in \ SQ\ (D)} |gIS\ (I_i)|!,$$

where gIS (*get Interaction Set*) is a total function $gIS : \mathcal{I} \rightarrow \mathcal{S}$ which given an interaction sequence I_i returns the interaction set over which it is defined. I use $\mathcal{S}$ to denote the universal set of sets. The ratio of the number of sequences in a DAG's out-of-turn paradigm to those in its browsing paradigm is given by the expression:

$$\frac{SoOP\ (D)}{|SQ\ (D)|}.$$

Lastly, notice also that SE can be made optional in each interpretation of out-of-turn interaction presented here. The absence of SE however would require the user to ultimately browse to reach leaf content pages.

Lemma 1 *Any interpretation of out-of-turn interaction is commutative, assuming both sides are defined. I.e.,*

$$OOT_1\ (OOT_1\ (D,\ x),\ y) = OOT_1\ (OOT_1\ (D,\ y),\ x),$$
$$OOT_2\ (OOT_2\ (D,\ x),\ y) = OOT_2\ (OOT_2\ (D,\ y),\ x),$$
$$GOOT_1\ (GOOT_1\ (D,\ x),\ y) = GOOT_1\ (GOOT_1\ (D,\ y),\ x), \text{ and}$$
$$GOOT_2\ (GOOT_2\ (D,\ x),\ y) = GOOT_2\ (GOOT_2\ (D,\ y),\ x),$$

where x and y represent terms.

> **Outline of Proof**: I illustrate the commutativity of the first interpretation of out-of-turn interaction. The proofs of the commutativity of the other interpretations follow similarly and are omitted. A diagrammatic outline of the proof of Lemma 1 is shown in Fig. 4.6. Spline curves in Fig. 1 are not meant to represent edges, but rather a series of edges. In addition, the labels decorating the curves are edge-labels, however, their order along the curve is not meant to reflect the order of the edges they label. For purposes of presentation simplification, my argument as well as Fig. 4.6 has deliberately ignored the shrinking of edges (SE) involved in the definition of OOT_1. This does not affect the proof because SE does not remove any paths completely. Rather it only shortens paths and is thus purely cosmetic.
>
> $OOT_1\ (D,\ x)$ removes all paths to leaves not reachable via a path containing an edge labeled x. In other words, it retains all paths to leaves reachable via a path containing an edge labeled x. There are four types of paths remaining:
>
> 1. Those containing an edge labeled x, but not an edge labeled y. These paths *do not* reach leaves reachable by a path with an edge labeled y.
> 2. Those containing an edge labeled x, but not an edge labeled y. These paths *do* reach leaves reachable by a path with an edge labeled y.
> 3. Those containing an edge labeled x and an edge labeled y.
> 4. Those containing an edge labeled y, but not an edge labeled x.
>
> Paths 1–4 are depicted below $D_x^{'}$ in Fig. 4.6 (top-center) in a left-to-right order. $OOT_1\ (D_x^{'},\ y)$ removes all paths to leaves not reachable via a path containing an edge labeled y. In other words, it retains all paths to leaves reachable via a path containing an edge labeled y. Therefore, it will remove only type 1 paths, depicted using red in Fig. 4.6 (top-center). It will retain all paths of type 2–4. The net effect of these two successive applications of OOT_1 is that all paths to leaves not reachable via a path containing an edge labeled x *and* a path containing an edge labeled y (they could be the same path) are pruned. In other words, these two successive applications of OOT_1 retains all paths to leaves reachable via a path containing an edge labeled x and a path containing an edge labeled y. Running this argument where all x's are replaced with y's and vice versa results in the same conclusion. ∴ Lemma 1 is true. ________________________ □

4.3 Program-theoretic View of Out-of-turn Interaction

Let us now see how we can view the above graph-theoretic interpretations of out-of-turn interaction through a programmatic lens. I relate out-of-turn interaction, and generalizations thereof, to *program slicing*, an alternate program transformation. I begin by illustrating how I model interaction with a hierarchical website in an explicit programmatic representation.

4.3.1 Modeling Interaction Programmatically

Researchers have predominately modeled web interaction programmatically to maintain state across and within sessions [GFKF01, Que00]. Here I model interaction programmatically to customize information access. Fig. 4.7 (left) illustrates a programmatic model of a user's browsing interactions with Fig. 4.2. Notice my use of procedures to model recurring subtrees induced by crosslinks. A crosslink (also called a 'symbolic link') is a hyperlink to a webpage which has existing incoming hyperlinks. For example, the edge from vertex 3 to 4 in Fig. 4.2 labeled 'speakers' is a crosslink. Crosslinks are necessary for a graph to be a `DAG`. In large taxonomies of links to web resources, such as Yahoo! and the Open Directory Project (ODP at dmoz.org), crosslink labels are augmented with '@' and transport a user from one sub-branch of the taxonomy (e.g., News) to another (e.g., Sports). Notice that my use of procedures breaks the symmetry with the underlying website, which was preserved in Chapter 2 when modeling trees. While modeling the same with `goto`s preserves symmetry, 'slicing with `goto` statements is fraught with strangeness' [Anda] and therefore I prefer to avoid it. The program slicing system I experimented with, CodeSurfer (described briefly below and further in Appendix C), takes the approach espoused in [KH02] for slicing programs with jumps and switches. Static slicing in the presence of `goto`s is described in [CF94].

The expressive constructs, most notably, conditionals, of most modern programming languages make programming languages an attractive vocabulary of discourse for modeling interaction with hierarchical hypermedia. While I can model interaction with (imperative, functional, or logic) programming languages, I use C here for purposes of presentation and familiarity. At the NATO Conference on Software Engineering Techniques held in Rome, Italy in 1969, A. Perlis said, 'A good programming language is a conceptual universe for thinking about programming' [RB69]. Similarly here, a good programming language is a conceptual universe for thinking about interacting with hierarchical hypermedia.

4.3.2 Program Slicing

I relate the four interpretations of out-of-turn interaction to the application of program slicing, a common program transformation employed in debuggers, to suitably selected programmatic representations of interaction, such as that shown Fig. 4.7 (left). Program slicing [BG96, HH01, Tip95], originally introduced by Weiser [Wei79, Wei82, Wei84], is a technique used to extract statements, which *may* affect or be affected by the values of variables of interest computed at some point of interest, from a program. A slice of a program is taken w.r.t. a (point of interest, variable of interest) pair, referred to as the *slicing criterion.* The point of interest may be specified with a line number from the program. The resulting

$\mathtt{P_D}$

```
if (arts)
  if (theatre)
    if (drama)
      page = 10;
    if (music)
      f1();
  if (music)
    f2();
if (computers)
  if (speakers)
    f2();
  if (software)
    if (music)
      f3();
    if (business)
      page = 14;
  if (hardware)
    if (memory)
      page = 12;

void f1()
  page = 11;
}

void f2() {
  if (theatre)
    f1();
  if (classical)
    page = 9;
  if (jazz)
    page = 8;
  if (software)
    f3();
}

void f3() {
  page = 13;
}
```

$\mathtt{P_{D'}} = [\![\mathtt{zoom}]\!][\mathtt{P_D},\ \mathtt{music}]$

```
if (arts)
  if (theatre)


    if (music)
      f1();
  if (music)
    f2();
if (computers)
  if (speakers)
    f2();
  if (software)
    if (music)
      f3();






void f1();
  page = 11;
}

void f2() {
  if (theatre)
    f1();
  if (classical)
    page = 9;
  if (jazz)
    page = 8;
  if (software)
    f3();
}

void f3() {
  page = 13;
}
```

$\mathtt{P_{D''}} = [\![\mathtt{mix}]\!][\mathtt{P_{D'}},\ \mathtt{music} = 1]$

```
if (arts)
  if (theatre)



      f1();

    f2();
if (computers)
  if (speakers)
    f2();
  if (software)

      f3();






void f1(); {
  page = 11;
}

void f2() {
  if (theatre)
    f1();
  if (classical)
    page = 9;
  if (jazz)
    page = 8;
  if (software)
    f3();
}

void f3() {
  page = 13;
}
```

Figure 4.7: Modeling interaction programmatically. (left) $\mathtt{P_D}$, programmatic representation of interaction with the website modeled by the DAG D in Fig. 4.2. (center) $\mathtt{P_{D'}}$, results of applying the zoom transformation to (left) w.r.t. '`music`.' This program is the representation of interaction with the website modeled by the DAG D' in Fig. 4.3 (right). (right) $\mathtt{P_{D''}}$, results of applying partial evaluation to (center) w.r.t. '$\mathtt{music} = 1$.' This program is the representation of interaction with the website modeled by the D'' in Fig. 4.4.

line	program	backward (6, vol)	forward (1, h)
(1)	read (r, h);	read (r, h);	read (r, h);
(2)	cArea = π*r^2;	cArea = π*r^2;	
(3)	sArea = 2*cArea+2*r*π*h;		sArea = 2*cArea+2*r*π*h;
(4)	vol = cArea*h;	vol = cArea*h;	vol = cArea*h;
(5)	print (sArea);		print (sArea);
(6)	print (vol);	print (vol);	print (vol);

Figure 4.8: Illustration of program slicing (simplified for purposes of presentation). (left) A program which takes the radius and height of a cylinder as input and computes and prints the cylinder's surface area and volume. (center) A static backward slice w.r.t. (6, `vol`). (right) A static forward slice w.r.t. (1, `h`) (variable key: `r` = radius; `h` = height; `cArea` = circle area; `sArea` = surface area; `vol` = volume).

slice consists of all program statements which may affect or be affected by the value of the variable at the specified point.

Fig. 4.8 illustrates program slicing. Slices such as that shown in Fig. 4.8 (center), which Weiser first articulated, are called 'executable backward static slices' [BG96, HRB90, Ven91] (referred to here as simply *backward slices*). They are *executable* because the slice is required to be an executable program. The slice is *backward* since this is the direction in which dependencies are followed to their sources in the program. Contrast this with a *forward slice* [HRB90] which consists of the program statements affected by the value of a particular variable at a particular statement (see Fig. 4.8, right). Backward slices contain data and control predecessors, while forward slices consist of data and control successors. Lastly, the slice is *static* because it is computed without consideration of the program's input. Dynamic slicing techniques are covered in Appendix B. For an introduction to program slicing, techniques for computing slices, and applications, I refer the interested reader to [BG96, HH01, Tip95].

I relate FP to the program intersection of two unions of forward slices. Researchers in the programming languages community consider a union of slices as a slice itself and trivial to compute. The slicing criteria 'can be easily extended to slicing w.r.t. a collection of locations and a collection of variables at each location by taking the union of the individual slices' [BG96]. The first union consists of the forward slices of the program w.r.t. the variable modeling the out-of-turn input at *every* occurrence of it in the program. The second union consists of the forward slices of the program w.r.t. the variable indexing the leaf vertices at *every* occurrence of it in the program. The intersection of these two unions results in several occurrences of the variable indexing the leaf vertices (e.g., `page` in Fig. 4.7). Each occurrence is at a point in the program which is affected by the variable corresponding to the out-of-turn input. In graph terms, this procedure results in the set of leaf vertices classified by all interaction sequences involving the out-of-turn input.

The set of program fragments, thus obtained from FP, can be thought of as slicing criteria input to BP. BP is then the union of backward slices, each w.r.t. every (point, variable) pair resulting from the initial forward slicing procedure. Intuitively, given valid input, a forward slice is performed w.r.t. the corresponding program variable to determine the terminal webpages that are reachable from that point. These webpages are collected and back-propagated via backward slicing, so that only those paths that reach these pages

	Syntax-preserving	**Semantic-preserving**
Partial evaluation	×	√
Program slicing	√	×

Table 4.3: Comparison of partial evaluation and program slicing along a syntax- vs. semantic-preserving dichotomy.

are retained. Notice that these two operations implicitly capture exclusions among program variables; e.g., when the user says 'Democrat' the slices will remove any program segments that involve Republicans. Such a combination of forward and backward slicing is quite similar (but not identical) to Sacco's zoom operator for interactively pruning information hierarchies [Sac00]. Thus, I call this entire program transformation technique *zoom*.

The idea of performing set-theoretic operations on forward and backward slices is closely related to the concepts of *program dicing* [BG96, LW87] and *program chopping* [JR94b]. Performing set-theoretic operations on one or more backward program slices yields a *program dice* [BG96, LW87]. Originally program dicing was limited to backward slices. Program chopping, on the other hand, which also is a generalization of slicing [JR94b], is an extension of dicing to forward slices. Forward slices increase the usefulness of dicing [BG96]. Chopping identifies the statements which transmit values from one statement to another. In other words, it shows all the ways which one set of program points affect another set of points. A *program chop* [JR94b, JR94a, RR95] therefore consists of all program points affected between one point (the chop source) and another (the chop target) [ART03]. It also is the subset of the intersection of a forward and backward slice [Andb]. In the absence of procedures, a chop 'can be viewed as a generalized kind of program dice' [BG96].

Table. 4.3 compares partial evaluation and program slicing from a programming languages perspective. It reinforces that while partial evaluation is semantic-preserving, it is not syntax-preserving. Conversely, while the variants of program slicing considered in this thesis are syntax-preserving, they are not semantic-preserving. However, there are variants of program slicing (e.g., *amorphous slicing*, also known as *semantic slicing*) which are the reverse: semantic-preserving, but not syntax-preserving (see Appendix B for more details).

Notation: Programs as Data Objects

To succinctly capture and describe my program transformation techniques (e.g., the above notion of zoom) in a programming languages context, I adopt a slightly modified notation for describing the semantic function of a programming language used in a popular textbook on partial evaluation [JGS93b]. A specification language, defined by a context-free grammar, for program transformations is introduced in [HC90]. The language is imperative; for purposes of presentation, here I prefer to use a declarative style. GrammaTech, Inc., the company which develops and produces the state of the art program slicer for ANSI C, CodeSurfer, is currently working on a textual representation, employing a Lisp-like syntax, for set-theoretic operations over program slices for a future release, e.g., (intersect (slice A) (slice B)) [Anda].

1. $[\![\mathtt{int}]\!][\mathtt{P_D}, \mathtt{x_1} = \mathtt{1}, \mathtt{x_2} = \mathtt{0}, \ldots, \mathtt{x_n} = \mathtt{1}]$ denotes 'partially, interpret the programmatic representation of DAG D ($\mathtt{P_D}$) w.r.t. the partial assignment of variables $\mathtt{x_1} = \mathtt{1}$, $\mathtt{x_2} = \mathtt{0}$,

(Interpretation of) Interaction Technique	**Program Transformation Technique**
Browsing	$\llbracket\mathtt{int}\rrbracket\ [\mathtt{P_D},\ \mathtt{input}\ =\ \mathtt{1}]$
Interpretation 1 of OOT Interaction	$\llbracket\mathtt{mix}\rrbracket\ [\llbracket\mathtt{zoom}\rrbracket\ [\mathtt{P_D},\ \mathtt{input}],\ \mathtt{input}\ =\ \mathtt{1}]$
Interpretation 2 of OOT Interaction	$\llbracket\mathtt{mix}\rrbracket\ [\llbracket\mathtt{sp}\rrbracket\ [\mathtt{P_D},\ \mathtt{input}],\ \mathtt{input}\ =\ \mathtt{1}]$
Gen. Interpretation 1 of OOT Interaction	$\llbracket\mathtt{mix}\rrbracket\ [\llbracket\mathtt{backward}\rrbracket\ [\mathtt{P_D},\ \llbracket\mathrm{SL}\rrbracket\ [\mathtt{P_D},\ \mathtt{input}]],\ \mathtt{input}\ =\ \mathtt{1}]$
Gen. Interpretation 2 of OOT Interaction	$\llbracket\mathtt{mix}\rrbracket\ [\llbracket\mathtt{sp}\rrbracket\ [\llbracket\mathtt{backward}\rrbracket\ [\mathtt{P_D},\ \llbracket\mathrm{SL}\rrbracket\ [\mathtt{P_D},\ \mathtt{input}]],\ \mathtt{input}],\ \mathtt{input}\ =\ \mathtt{1}]$

Table 4.4: Relating interaction techniques in DAG models of a websites to compositions of program transformations. Notice that $\llbracket\mathrm{SL}\rrbracket$ is a *meta*-program-transformation. It represents any program transformation which returns a set of program points containing the variable `page`.

…, $\mathtt{x_n} = \mathtt{1}$.' $\llbracket\mathtt{int}\rrbracket$ denotes the application of a stepwise interpreter. Although it is not traditionally viewed as such, I use interpretation here as a program transformer.

2. $\llbracket\mathtt{mix}\rrbracket[\mathtt{P_D}, \mathtt{x_1} = \mathtt{1}, \mathtt{x_2} = \mathtt{0}, \ldots, \mathtt{x_n} = \mathtt{1}]$ denotes 'partially evaluate (non-sequentially interpret) the programmatic representation of DAG D ($\mathtt{P_D}$) w.r.t. the partial assignment of variables $\mathtt{x_1} = \mathtt{1}$, $\mathtt{x_2} = \mathtt{0}$, …, $\mathtt{x_n} = \mathtt{1}$.' $\llbracket\mathtt{mix}\rrbracket$ is the conventional way to denote a partial evaluator [Jon96, Jon97, JGS93b]. It refers to 'mixed computation' since a partial evaluator performs a mixture of interpretation and code-generation [Jon96].

3. $\llbracket\mathtt{forward}\rrbracket[\mathtt{P_D}, \mathtt{x}]$ denotes 'union each forward slice of the the programmatic representation of DAG D ($\mathtt{P_D}$) w.r.t. the variable `x` at every program point containing `x`.'

4. $\llbracket\mathtt{backward}\rrbracket[\mathtt{P_D}, \mathtt{x_1}, \mathtt{x_2}, \ldots, \mathtt{x_n}]$ denotes 'union each backward slice of the the programmatic representation of DAG D ($\mathtt{P_D}$) w.r.t. the variable `x` at program points $\mathtt{1}, \mathtt{2}, \ldots, \mathtt{n}$.'

Using this notation, $\llbracket\mathrm{zoom}\rrbracket$ is formally defined as

$$\llbracket\mathtt{zoom}\rrbracket[\mathtt{P_D},\ \mathtt{input}] = \llbracket\mathtt{backward}\rrbracket[\mathtt{P_D},\ \llbracket\mathtt{forward}\rrbracket[\mathtt{P_D},\ \mathtt{input}] \cap \llbracket\mathtt{forward}\rrbracket[\mathtt{P_D},\ \mathtt{page}]],$$

where `input` is the program variable modeling the out-of-turn input and `page` is the variable indexing the leaf vertices.

I define $\llbracket\mathrm{sp}\rrbracket$, the programmatic analog to *Select paths* (SP), as

$$\llbracket\mathtt{sp}\rrbracket[\mathtt{P_D},\ \mathtt{x}] = \llbracket\mathtt{forward}\rrbracket[\mathtt{P_D},\ \mathtt{x}] \cup \llbracket\mathtt{backward}\rrbracket[\mathtt{P_D},\ \mathtt{x}]$$

I define $\llbracket\mathtt{te}\rrbracket[\mathtt{P_D}]$, the programmatic analog to *Term extraction* (TE), as the union of all *program data successors* of each structural variable at its declaration. A program data successor is a restriction to a forward slice in that rather than including transitive dependencies, it just contains the immediate dependency of a program point. Specifically, 'a program point's data successors are the points where the variables that were modified at that point are used' [ART03]. Since structural variables' sole presence in these programs arises in the context of an `if (...)` expression, we might just as easily think of conducting $\llbracket\mathtt{te}\rrbracket[\mathtt{P_D}]$ with a regular expression.

Armed with these formalisms, I can relate many interpretations of interaction techniques, including browsing, to classical program transformations (see Table 4.4). Program slicing is typically used for debugging, safety, and security. Only few have used program slicing for web applications [RT01b]. My use of slicing here helps marry it with information personalization.

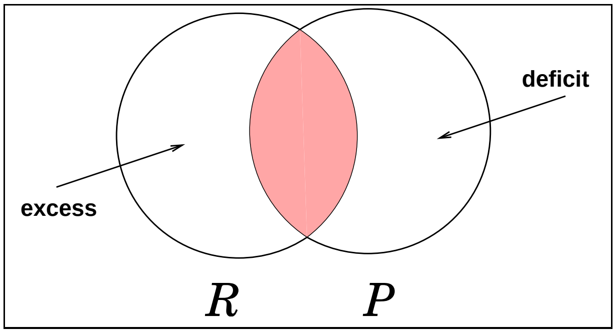


Figure 4.9: Venn diagram highlighting the intersection between the set of sequences R (left) staged by an incomplete, unsound model and its intended interaction paradigm $\mathcal{P}$ (right).

Recall that Fig. 4.4, the result of an out-of-turn interaction, violates my definition of a DAG. This implies that I must post-process the program resulting from the application of each program transformation technique to fuse program segments modeling edges with the same source and label. This can be done with grammar-based approaches to program transformation such as that espoused in [HC90].

4.4 Evaluation Criteria

A *model* $\mathcal{M} = (P_D,\ \mathcal{X})$ is a (programmatic representation, transformation technique) pair. I would like to evaluate a model by assessing its capability to realize a desired set of interaction sequences. To do so I measure how close the model comes to realizing its targeted interaction paradigm $\mathcal{P}$. Ideally, I would like to have

$$\mathcal{M} \rightsquigarrow I \leftrightarrow I \in \mathcal{P},$$

where $\rightsquigarrow$ denotes 'stages,' i.e., $\mathcal{M}$ stages interaction sequence I *iff* I is in the interaction paradigm $\mathcal{P}$. A model stages an interaction sequence if successive applications of its transformation technique to its programmatic representation, given user input, realize the interaction sequence. In this manner, the model *stages* the user's interaction.

4.4.1 Soundness of a Model: $\mathcal{M} \rightsquigarrow I \rightarrow I \in \mathcal{P}$

A model $\mathcal{M}$ is *sound* for an interaction paradigm $\mathcal{P}$ if each interaction sequence that $\mathcal{M}$ stages is in $\mathcal{P}$. In other words, if the model can stage an interaction sequence, then the sequence is in the paradigm.

4.4.2 Completeness of a Model: $I \in \mathcal{P} \rightarrow\ \mathcal{M} \rightsquigarrow I$

A model $\mathcal{M}$ is *complete* for an interaction paradigm $\mathcal{P}$ if $\mathcal{M}$ can stage each interaction sequence in $\mathcal{P}$. In other words, if an interaction sequence is in the paradigm, then the model can stage it.

In this view, I can identify both the *excess* and *deficit* of a model. A complete, but unsound, model has excess – interaction sequences not in its intended paradigm that it stages. On the other hand, a sound, but incomplete, model has deficit – sequences in its intended interaction paradigm that it fails to stage. An unsound and incomplete model exhibits both excess and deficit w.r.t. its targeted paradigm. A *sufficiency* metric for a model can then be formulated akin to the recall measure in information retrieval:

$$\mathit{sufficiency} = \frac{|R \, \cap \, \mathcal{P}|}{|\mathcal{P}|} \; = \; \frac{|R \, \cap \, \mathcal{P}|}{SoOP \; (D)},$$

where R represents the set of sequences staged by the model. Fig. 4.9 illustrates how the excess and deficit of model arises.

The program transformation techniques in Table 4.4 are each complete for the out-of-turn paradigm studied in this dissertation. They are unsound because the supplied terms may not all lie on a single path. Notice that a browsing paradigm is a proper subset of an out-of-turn interaction paradigm. This is a significant result as it means that I can support the union of these two interaction paradigms with a single program transformation technique. In other words, no anticipation of in-turn or out-of-turn input is necessary to discern a program transformation technique. Rather, since both in-turn and out-of-turn inputs are partial, and since my techniques exploit partial information, I achieve uniform processing of partial input. For interaction, this means that the user can interleave hyperlink clicks (browsing) and voice utterances (out-of-turn inputs) in any order she desires, to achieve a mixed-initiative mode of information seeking.

Lemma 2 *Any complete model for the interaction paradigm of the out-of-turn interaction techniques considered here is complete for a browsing paradigm. In other words, out-of-turn interaction subsumes browsing.*

> **Outline of Proof**: Since there is only one parenthood relation in any DAG D, given the definition of a browsing interaction sequence from above, there is only one browsing interaction sequence per interaction set from D. Since a browsing interaction sequence is a total ordering by definition, its corresponding poset has only one linear extension (the total ordering itself). Therefore, the browsing paradigm of D contains as many, and no more, linear extensions as are interaction sets (or paths) from D (i.e., to reiterate, $|\mathcal{P}| = |SQ \; (D)|$).
>
> The arrival time relation implied by out-of-turn interaction is a partial ordering containing only the reflexive tuples of terms from any interaction set. In other words, none of the terms from the interaction set are ordered. Since none of the terms are ordered, the number of linear extensions (out-of-turn interaction sequences) of the posets associated with these partial orders equals the number of all complete permutations of the interaction set. Since all complete permutations of each interaction set are included in the out-of-turn interaction paradigm and since one of the complete permutations of each interaction set corresponds to the parenthood relation of D, each browsing interaction sequence from D is added at some point to the out-of-turn interaction paradigm of D. $\therefore$ by construction of the out-of-turn interaction paradigm of D, Lemma 2 is true. ____________ □

The completeness of each program transformation technique in Table 4.4 holds under the assumption that no path from the root of the website to each leaf contains more than one hyperlink with the same label. Intuitively, this is because communicating (in-turn or out-of-turn) partial information initiates a program transformation technique which simplifies the site w.r.t. *all* hyperlinks labeled with that partial input, some of which may lie on the same path. This assumption is captured by my definition of interaction set and (browsing) interaction sequence. As a result, my definition of browsing is slightly different than its traditional interpretation.

Beyond soundness and completeness, I developed a measure which estimates the *compression* achieved in a model – the ratio of interaction sequences realizable via interpretation to the total number of sequences realizable via transformation:

$$\mathit{compression} = \frac{|R - E|}{|R|},$$

where E represents the set of interaction sequences stageable from P_D with an interpreter ($[\![\mathtt{int}]\!]$). Intuitively, the compression ratio quantifies the percentage of sequences which I get 'for free' by using the program transformation technique.

An effective model is one which maximizes both sufficiency and compression. Notice however that these measures are bipolar and foster a tradeoff which resembles the precision-recall tradeoff in IR. To maximize sufficiency, I might choose to explicitly model each interaction sequence in the representation, affecting the compression ratio negatively.

An alternative way to characterize models is to study the level at which they 'factor' the desired space of interaction sequences. In [RP01] we have defined three classes of factored representations: under-factored, well-factored, and over-factored. These classes also can be used to characterize the developed models.

There are several alternate applicable evaluation criteria for this research. Evaluating personalization applications for traditional user-satisfaction and task completion metrics is the prevalent practice, and is the topic of Chapter 6. In addition to measuring satisfaction, studies with users can improve our understanding of an extant interaction paradigm or yield new paradigms to model. Personalization applications also can be evaluated w.r.t. IR metrics. For example, Sacco [Sac00] studies how the application of zoom as well as bucket size (i.e., number of documents classified under each terminal concept) affect the maximum resolution of a taxonomy. Maximum resolution, which is a measure of retrieval effectiveness, is the average minimum number of documents the user has to manually inspect.

4.5 Graph-theoretic Classes of Hierarchical Hypermedia

In this section, I identify classes of hierarchical hypermedia for insight into the possibility of specialized program transformation techniques for each class. Relating specialized techniques to these classes will afford an alternate method, other than the all-accommodating general program transformation technique, for supporting personalized interaction. I begin by defining a few terms.

The *maximum depth* of D is the level of an edge-label in D which is greater than or equal to the level of all other edge-labels in D. A *cluster* c is a term-level set s.t. no edge label in

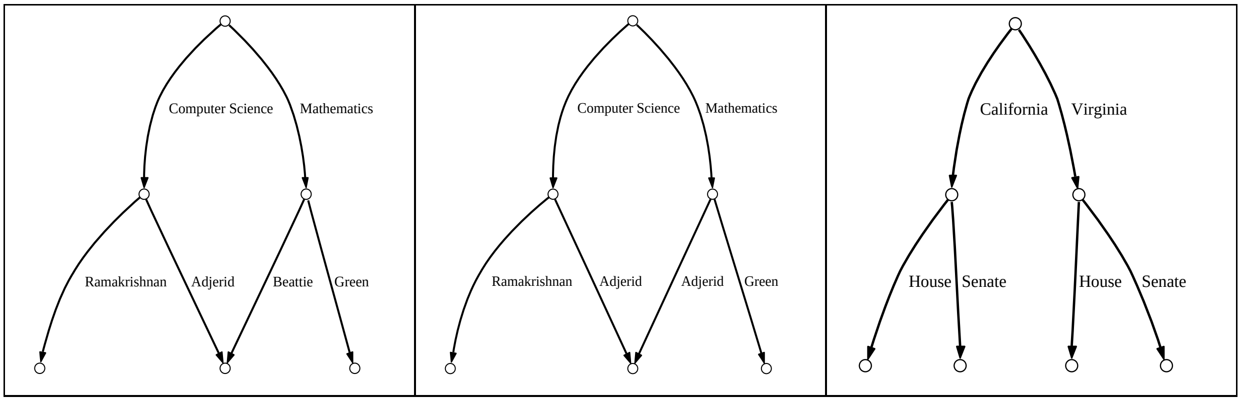


Figure 4.10: Simple levelwise DAGs. (left) not-mutually-exclusive. (center) weak-mutually-exclusive. (right) strong-mutually-exclusive.

it labels an edge in a different term-level set. If the maximum depth of D equals the number of clusters in D, then D is *levelwise*. Intuitively, a levelwise DAG is one where each level of the DAG corresponds to a facet of information assessment in the website it models, or in other words each term $t_i \in TE\ (D)$ resides at exactly one level.

Each DAG shown in Fig. 4.10 is levelwise. The DAG in Fig. 4.10 (left) models a simplified path through the online Virginia Tech timetable of courses. Its clusters are {Computer Science, Mathematics} and {Ramakrishnan, Adjerid, Beattie, Green} implying that the two levels shown correspond to 'Department' and 'Instructor' facets, respectively. Notice that crosslinks, such as those illustrated in Fig. 4.10 (left and center) by the edge labeled 'Adjerid' emanating from the target of the edge labeled 'Computer Science,' are necessary to model the presence of cross-listed courses. In the case of Fig. 4.10 (center), the target of the two edges labeled 'Adjerid' models a course cross-listed in both the Computer Science and Mathematics departments. The DAG in Fig. 4.10 (right), which models the Congressional portion of the Project Vote Smart (PVS at votesmart.org) website, also is levelwise, albeit without crosslinks. Its clusters are {California, Virginia} and {House, Senate} implying that the two levels shown correspond to 'State' and 'Branch of Congress' facets, respectively. In contrast to those shown in Fig. 4.10, notice that the DAG in Fig. 4.2 is not-levelwise.

I now introduce the concept of mutual-exclusivity in graph models of websites. If no leaf vertex of D lies at the end of two paths from D which each involve a distinct edge-label from a term-co-occurrence set T, then I say that T is *mutually-exclusive*. While there are several mutually-exclusive term-co-occurrence sets (e.g., {Computer Science, Green}) in the DAGs shown in Fig. 4.10 (left and center), none are clusters. On the other hand, the term-level sets, {California, Virginia} and {House, Senate}, of the DAG shown in Fig. 4.10 (right), are mutually-exclusive and clusters. If D is levelwise and has at least one mutually-exclusive cluster, then D is *weak-mutually-exclusive*. If D is levelwise and no leaf vertex of D lies at the end of two distinct paths from D, where each path contains a distinct term from the same cluster, then D is *strong-mutually-exclusive*. Note that the mutually-exclusivity of DAGs subsumes the levelwise property, by definition. Notice further that replacing only one edge-label (i.e., 'Beattie' to 'Adjerid') in the DAG shown in Fig. 4.10 (left) makes it weak-mutually-exclusive (see Fig. 4.10, center): its cluster at level-two is mutually-exclusive, but that at level-one is not, due to the crosslink. This DAG models a course in the online Virginia Tech timetable which presumably has two sections, each taught by a different instructor, from different departments. If there is a unique simple path from the root of D to each vertex in D, then D is an *edge-labeled, rooted tree* (hereafter referred to as a *tree*). Notice that the DAG in Fig. 4.10 (right) is a tree (more on this later). Observe also that I developed the above classes of hierarchical hypermedia without reference to any semantics. These notions are purely syntactic. For instance, the levelwise property does not take into account *polysemy* of terms.

Lemma 3 *A* DAG *D is levelwise and each term-level set of D is a cluster and mutually-exclusive iff D is strong-mutually-exclusive.*

Outline of Proof:

$\longrightarrow$: If a DAG D is levelwise and each term-level set of D is a cluster and mutually-exclusive, then D is strong-mutually-exclusive.

Assume D is levelwise and each term-level set c_l of D is a cluster and mutually-exclusive. This means that no leaf vertex of D lies at the end of two distinct paths from D, where each path contains a distinct term from c_l. Assume further that D is not strong-mutually-exclusive. This means that some leaf vertex of D lies at the end of two distinct paths from D, where each path contains a distinct term from the same cluster, call it c_x. However, this implies that c_x is not-mutually-exclusive. This is a contradiction.

$\longleftarrow$: If a DAG D is strong-mutually-exclusive, then D is levelwise and each term-level set of D is a cluster and mutually-exclusive.

Assume D is strong-mutually-exclusive. By the definition of a strong-mutually-exclusive DAG, D is levelwise. By the definition of levelwise, each term-level set in D is a cluster. Since D is strong-mutually-exclusive, no leaf vertex of D lies at the end of two distinct paths from D, where each path contains a distinct term from the same cluster. This means that each term-level set in D is a cluster and mutually-exclusive.

$\therefore$ Lemma 3 is true. ______________________________ □

Lemma 4 *A strong-mutually-exclusive* DAG *which is not a tree does not exist, given my definition of a* DAG*.*

Outline of Proof: The presence of crosslinks shatters strong-mutually-exclusivity. The only way that a non-tree DAG could be strong-mutually-exclusive is if the edges of each level above every crosslink had the same label (see Fig. 4.11). However, this violates my definition of a DAG and thus violates the definition of a strong-mutually-exclusive DAG. For instance, try to convert the tree of Fig. 4.10 (right) into a general DAG, while retaining the strong-mutual-exclusive property. $\therefore$ Lemma 4 is true. ______________________________ □

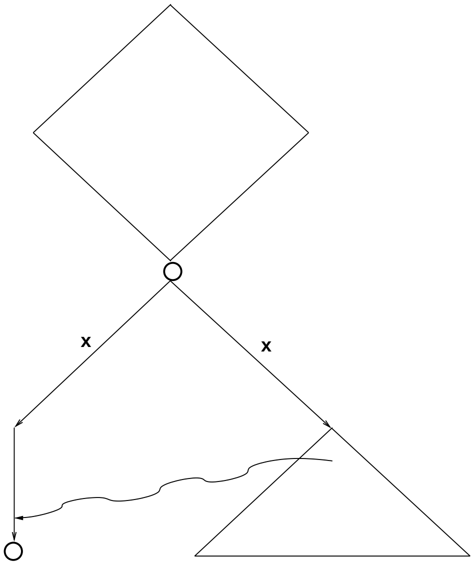


Figure 4.11: An 'almost' strong-mutually-exclusive DAG. The crosslink is unlabeled here to reinforce that its label is irrelevant.

Lemma 5 *Interpretation 1 of out-of-turn interaction over a tree D is functionally equivalent to interpretation 2 of out-of-turn interaction over D.*

> **Outline of Proof**: Assume D is a tree. Let $D^{'} = BP\ (D,\ FP\ (D,\ t))$. Since
>
> $$OOT_1\ (D,\ t) = SE\ (BP\ (D,\ FP\ (D,\ t)),\ t) \text{ and}$$
> $$OOT_2\ (D,\ t) = SE\ (SP\ ((BP\ (D,\ FP\ (D,\ t)),\ t)),$$
>
> this proof reduces to proving that $SP\ (D^{'}) = D^{'}$.
>
> Let L_t be the set of leaves returned from $FP\ (D,\ t)$. Since D is a tree, there is one only path from the root of D to each of its leaves and therefore each leaf in L_t can be reached via only one path from the root of D. Since I attained L_t by forward propagating paths containing an edge labeled t, every path from the root of the tree returned by $BP\ (L_t)$ to each leaf in L_t will contain an edge labeled t. Thus, $SP\ (D^{'})$ will return all paths from the root of $D^{'}$ to each of the leaves in L_t (i.e., it will return $D^{'}$). I have just shown that $SP\ (D^{'}) = D^{'}$. $\therefore$ Lemma 5 is true. ______ $\square$

Lemma 6 *Generalized interpretation 1 of out-of-turn interaction over a tree D is functionally equivalent to generalized interpretation 2 of out-of-turn interaction over D.*

> **Outline of Proof**: This proof follows analogously from Lemma 5. I can construct its outline by replacing all occurrences of FP in the proof of Lemma 5 with SL, since FP is an instance of SL. $\therefore$ Lemma 6 is true ______. $\square$

Lemma 7 *The interpretations of out-of-turn interaction considered in this thesis preserve the levelwise property in* DAG*s.*

Outline of Proof: The interpretations of out-of-turn interaction provided simply remove entire paths from a DAG and shrink the remaining paths w.r.t. the out-of-turn input. Assume a DAG D is levelwise. Since D is levelwise, each of its term-level sets are clusters by the definition of a levelwise DAG. Lemma 5 illustrates that the DAG resulting from each of

$$\begin{gathered} BP\ (D,\ FP\ (D,\ t)), \\ SP\ (BP\ (D,\ FP\ (D,\ t)),\ t), \\ BP\ (D,\ SL\ (D,\ t)), \text{ or} \\ SP\ (BP\ (D,\ SL\ (D,\ t)),\ t) \end{gathered}$$

contains only paths from the root of D to each leaf of the resulting DAG $D^{'}$ which contain an edge labeled t. Since I have shrunk no edges, $D^{'}$ has the same maximum depth of D. This means that each term-level set from $D^{'}$ is a subset of the term-level set in D at the corresponding level. Thus, $D^{'}$ is levelwise. This means that each of its term-level sets are clusters. Since a cluster cannot contain two terms which label two distinct edges on the same path, no path from the root of $D^{'}$ to each of its leaves can contain two distinct edges with the same label. Since $D^{'}$ is levelwise, t can exist in one cluster and thus one term-level set. This means that the application of SE to $D^{'}$ w.r.t. t removes exactly one edge from each path from the root of $D^{'}$ to each of its leaves. Thus, the application of SE to $D^{'}$ w.r.t. t reduces the maximum depth of $D^{'}$ by at most one. $\therefore$ Lemma 7 is true. ______ □

Corollary 1 *The interpretations of out-of-turn interaction considered in this thesis preserve the levelwise property in trees.*

Outline of Proof: Follows directly from Lemma 7. ______ □

Lemma 8 *If a* DAG *D is a levelwise tree, then D is a strong-mutually-exclusive tree.*

Outline of Proof: Assume D is a levelwise tree. This means that there is a unique simple path from the root of D to each vertex in D, and thus each leaf in D. Thus, no leaf vertex of D lies at the end of two distinct paths from D, where each path contains a distinct term from the same cluster. I have just shown that D meets both conditions for being strong-mutually-exclusive. $\therefore$ Lemma 8 is true. ______ □

Corollary 2 *The interpretations of out-of-turn interaction considered in this thesis preserve the strong-mutually-exclusive property in trees.*

Outline of Proof: Follows directly from Lemmas 7 and 8. ______ □

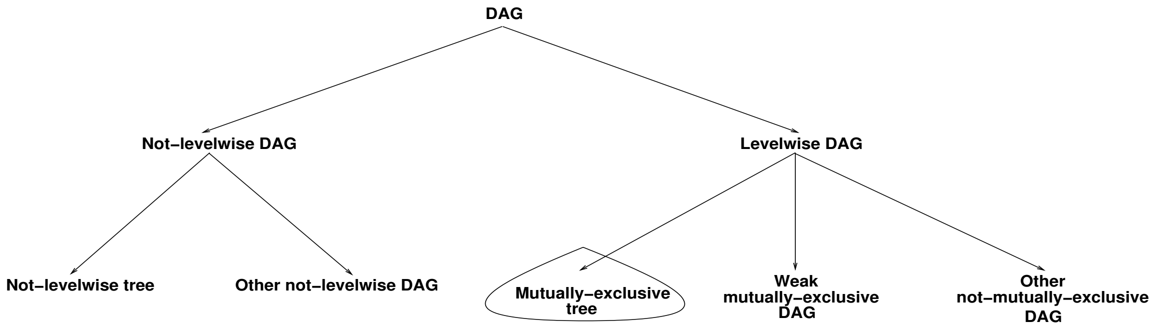


Figure 4.12: Partial order of classes of hierarchical hypermedia.

DAG	**not-ME**	**ME**
not-levelwise	$\surd$	$\perp$
levelwise	$\surd$	$\surd$

Tree	**not-ME**	**ME**
not-levelwise	$\surd$	$\perp$
levelwise	$\perp$	$\surd$

Table 4.5: Alternate illustration of classes of hierarchical hypermedia making the five considered classes (leaves of the diagram in Fig. 4.12) salient. Symbols $\surd$ and $\perp$ denote 'defined' and 'undefined,' respectively.

Lemmas 4 and 8 make the type of mutually-exclusivity unequivocal in reference to non-tree DAGs and trees. Thus, unless stated otherwise, I refrain from qualifying mutually-exclusivity as 'weak' or 'strong' in such contexts.

A Hasse diagram for a partial order of these classes of hierarchical hypermedia is shown in Fig. 4.12. The leaf vertices in this Hasse diagram are the five classes considered in this chapter. However, I do not use the distinction between not-levelwise trees and other not-levelwise DAGs and simply refer to them using their parent class: not-levelwise DAG. The mutually-exclusive tree class (circled in Fig. 4.12) is my primary focus later in this thesis, especially when I study user interactions with websites (Chapter 6). Note that the other not-mutually-exclusive DAG class does not contain any trees, by Lemma 8, as shown in Fig. 4.12. An alternate view of these five classes is given in Table 4.5.

4.6 Additional Support Terms and Tools

Before I can broaden my discussion of these classes of hierarchical hypermedia, which includes methods for automatically identifying them as well as reasoning about interaction therein, I must develop some support terms and tools.

4.6.1 Path-term-co-occurrence

Intuitively, a *path-term-co-occurrence set* is a term-co-occurrence set which only contains all of the terms from D which co-occur with a particular term along paths through D. *Build path-term-co-occurrence set* is total function $BPTC : (\mathcal{D} \times T) \rightarrow P\ (T)$ which given D and a term $t \in T = TE\ (D)$ returns a path-term-co-occurrence set. It is defined as

$$BPTC\ (D,\ t) = \bigcup_{I_{t_i}\ \in\ GS\ (t,\ SQ\ (D))} gIS\ (I_{t_i}) - \{t\},$$

where I_{t_i} is an interaction sequence from D containing term t_i. The set {Adjerid, Green} is the path-term-co-occurrence set in the DAG D illustrated in Fig. 4.10 (center) w.r.t. the term 'Mathematics.'

4.6.2 Leaf-term-co-occurrence

Given a set of leaves from D reachable via paths containing a particular term t (i.e., $L_t = FP\ (D,\ t)$), a *leaf-term-co-occurrence set* is a term-co-occurrence set which only contains all of the terms from D, excluding t, which lie along all of the paths from D which reach all of the leaves in L_t. *Build leaf-term-co-occurrence set* is total function $BLTC : (\mathcal{D} \times T) \rightarrow P\ (T)$ which given D and term $t \in T = TE\ (D)$ returns a leaf-term-co-occurrence set. It is defined as

$$BLTC\ (D,\ t) = TE\ (BP\ (FP\ (D,\ t))) - \{t\}.$$

The set {Computer Science, Adjerid, Green} is the leaf-term-co-occurrence set in the DAG D illustrated in Fig. 4.10 (center) w.r.t. the term 'Mathematics.'

Lemma 9 *If a* DAG *D is a tree, then any path-term-co-occurrence set w.r.t. a term t in D is a leaf-term-co-occurrence set w.r.t. t in D. In other words, if a* DAG *D is a tree, then $BPTC\ (D,\ t) = BLTC\ (D,\ t)$.*

> **Outline of Proof**: Assume a DAG D is a tree. Thus, there is one simple path from the root of D to each of its leaves, by the definition of a tree. Suppose that T is a path-term-co-occurrence set w.r.t. term t in D. By the definition of a path-term-co-occurrence set, the terms in T are the complete set of terms which co-occur with t on each path from the root of D to a leaf involving t. Since there is only one such path in D per leaf because D is a tree, the terms in T constitute a complete set of terms from D, excluding t, which lie along all of the paths from D which reach all of the leaves reachable by each path involving t from the root of D to a leaf. This means that T is a leaf-term-co-occurrence set w.r.t. t in D by the definition of a leaf-term-co-occurrence set in a DAG. $\therefore$ Lemma 9 is true. □

Corollary 3 *If a* DAG *D is a tree, then the complete set of path-term-co-occurrence sets in D equals the complete set of leaf-term-co-occurrence sets in D. In other words, if a* DAG *D is a tree, then*

$$\bigcup_{t_i\ \in\ TE\ (D)} \{BPTC\ (D,\ t_i)\} = \bigcup_{t_i\ \in\ TE\ (D)} \{BLTC\ (D,\ t_i)\}.$$

> **Outline of Proof**: Follows directly from Lemma 9. □

Now I can use these tools to build up other tools which will help in automatically identifying classes of hypermedia.

Is disjoint term-co-occurrence set is a total function $iDTCs : (\mathcal{D} \times \mathcal{T}_C) \rightarrow \mathcal{B}$ which given D and a term-co-occurrence set T from D returns *true iff* no two edge-labels, $e_{l_1} \in T$; and $e_{l_2} \in T$, label two edges, e_1 and e_2, respectively, of D which lie on the same path, and $false$ otherwise. I use the variable $\mathcal{B}$ to denote the set $\{true,\ false\}$ and $\mathcal{T}_C$ to represent the universal set of term-co-occurrence sets. This function is defined as

$$iDTCs\ (D,\ T) = \bigwedge_{t_i\ \in\ T} \neg(\bigvee_{t_j\ \in\ BPTC\ (D,\ t_i)} M\ (t_j,\ T - \{t_i\})),$$

where M is a set-membership function. *Is set of disjoint term-co-occurrence sets* is a total function $iSoDTCs : (\mathcal{D} \times \mathcal{T}_T) \rightarrow \mathcal{B}$ which given D and a set of term-co-occurrence sets S returns *true* if $\forall T_i \in S,\ iDSoT\ (D,\ T_i) = true$, and $false$ otherwise. I use the variable $\mathcal{T}_T$ to denote the universal set of sets of term-co-occurrence sets. This function is defined as

$$iSoDTCs\ (D,\ S) = \bigwedge_{T_i\ \in\ S} iDTCs\ (D,\ T_i).$$

Test mutual-exclusivity of term-co-occurrence set is a total function $METC : (\mathcal{D} \times \mathcal{T}_C) \rightarrow \mathcal{B}$ which given D and a term-co-occurrence set T from D returns *true iff* no leaf vertex of D is reachable via two distinct paths from D, which each contain a distinct edge, e_1 and e_2 (labeled by edge-labels e_{l_1} and e_{l_2}, respectively) from T (i.e., $e_{l_1} \in T$ and $e_{l_2} \in T$), and $false$ otherwise. This function is defined as

$$METC\ (D,\ T) = \bigwedge_{t_i\ \in\ T} \neg(\bigvee_{t_j\ \in\ BLTC\ (D,\ t_i)} M\ (t_j,\ T - \{t_i\})),$$

Test mutual-exclusivity of set of term-co-occurrence sets is a total function $METCs : (\mathcal{D} \times \mathcal{T}_T) \rightarrow \mathcal{B}$ which given D and a set of term-co-occurrence sets S from D returns *true iff* $\forall T_i \in S, METC\ (D,\ T_i) = true$, and $false$ otherwise. It is defined as

$$METCs\ (D,\ S) = \bigwedge_{T_i\ \in\ S} METC\ (D,\ T_i).$$

Now we can see procedurally why the set

{{Computer Science, Mathematics}, {Adjerid, Green}}

of clusters is not a complete set of mutually-exclusive clusters and why the set {{California, Virginia}, {House, Senate}} is a complete set of mutually-exclusive clusters in the DAGs shown in Fig. 4.10 (left and right, respectively).

4.7 Automatically Identifying the Classes

I defined the terms, notation, and functions from the previous sections in part to simplify the automatic identification of instances of the developed classes of hierarchical hypermedia.

Is levelwise DAG is a total function $iL : \mathcal{D} \rightarrow \mathcal{B}$, which given D returns *true iff* D is levelwise and $false$ otherwise. It is defined as

$$iL\ (D) = iSoDTCs\ (D,\ LOT\ (D)),$$

where LOT (*term-level-order traversal*) is a total function $LOT : \mathcal{D} \rightarrow \mathcal{S}$ which given D returns a set $\{s_1, s_2, \ldots, s_M\}$ of sets where each set T_l $(= TLE\ (D,\ l))$ is the term-level set at level l. Now we can see procedurally why the sets

{{Computer Science, Mathematics},
{Ramakrishnan, Adjerid, Beattie, Green}},
{{Computer Science, Mathematics},
{Ramakrishnan, Adjerid, Green}}, and
{{California, Virginia}, {House, Senate}}

are complete sets of the clusters in the DAGs shown in Fig. 4.10 (left, center, and right, respectively). This helps us to see procedurally why all DAGs given in Fig. 4.10 are levelwise.

Is weak-mutually-exclusive DAG is a total function $iWMED : \mathcal{D} \rightarrow \mathcal{B}$, which given D returns *true iff* D is weak-mutually-exclusive and $false$ otherwise (i.e., *iff* D is not-mutually-exclusive). It is defined as

$$iWMED\ (D) = iL\ (D)\ \wedge\ (\bigvee_{T_l\ \in\ LOT\ (D)} METC\ (D,\ T_l)).$$

Now we can see procedurally why the DAG shown in Fig. 4.10 (center) is weak-mutually-exclusive, namely, because it is levelwise and its term-level set at level-two, {Ramakrishnan, Adjerid, Green}, is a mutually-exclusive cluster.

Is mutually-exclusive DAG is a total function $iMED : \mathcal{D} \rightarrow \mathcal{B}$, which given D returns *true iff* D is strong-mutually-exclusive and $false$ otherwise (i.e., *iff* D is not-strong-mutually-exclusive). Note that $iMED$ will never return true for a non-tree DAG. Following Lemma 3, $iMED$ is defined as

$$iMED\ (D) = iL\ (D)\ \wedge\ METCs\ (D,\ LOT\ (D)).$$

Identifying a mutually-exclusive tree D involves placing terms in a set if they do not lie on the same path, but have the same level. If the maximum depth of the tree equals the number of resulting sets (clusters), then D is mutually-exclusive. *Is mutually-exclusive tree* is a total function $iMET : \mathcal{D} \rightarrow \mathcal{B}$, which given a DAG D returns *true iff* D is a mutually-exclusive tree and $false$ otherwise. It is defined as

$$iMET\ (D) = iT\ (D)\ \wedge\ iMED\ (D) = iT\ (D)\ \wedge\ iL\ (D)\ \text{(by Lemma 8)},$$

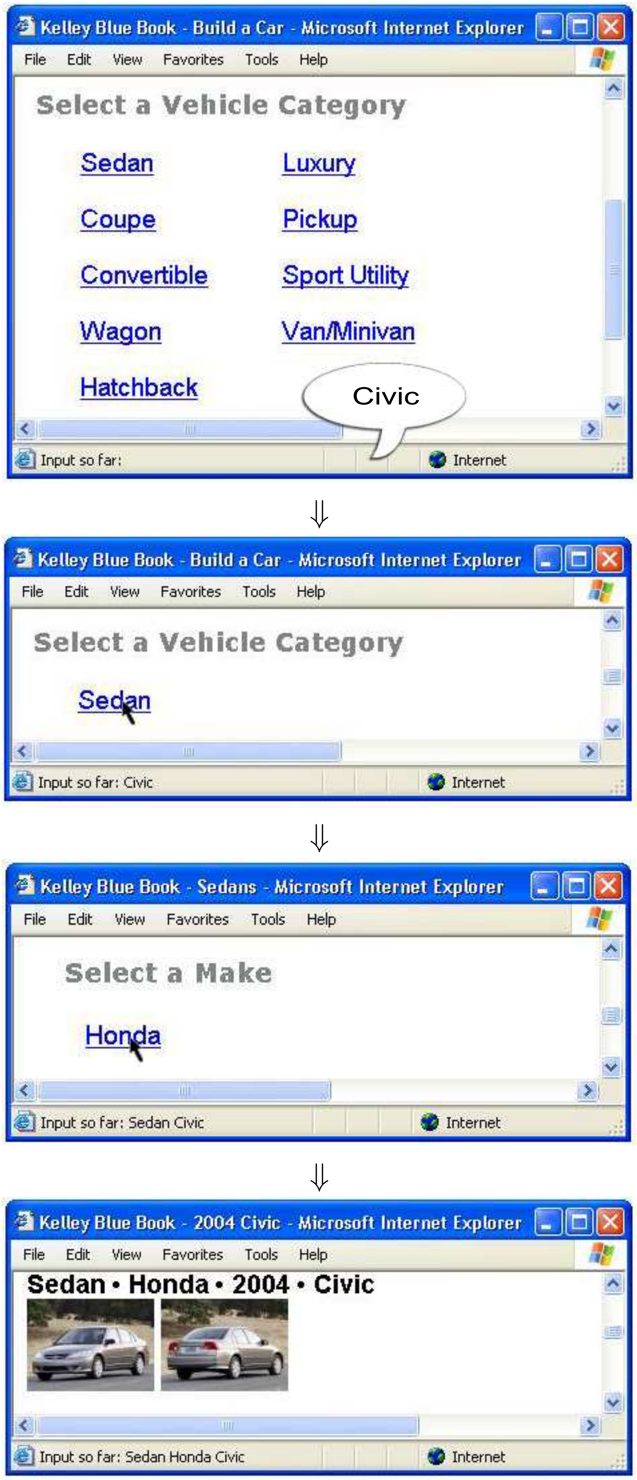
where iT (*is tree*) is a total function $iT : \mathcal{D} \rightarrow \mathcal{B}$, which given D returns *true iff* D is a tree and $false$ otherwise. Now we can see procedurally why the DAG D in Fig. 4.10 (right) is a mutually-exclusive tree. By Lemma 3, since it is levelwise and its complete set {{California, Virginia}, {House, Senate}} of clusters is a mutually-exclusive set of clusters, D is a mutually-exclusive tree. Note that the application of any interpretation of out-of-turn interaction to a DAG may result in a mutually-exclusive tree or a mutually-exclusive DAG. This will mean that to take advantage of program transformation techniques specialized for these classes, which I develop below, I will need to apply these identification procedures after *each* out-of-turn interaction.

4.8 Mining Functional Dependencies for Input Expansion

Before re-entering the program-theoretic domain, I must develop a few more graph-theoretic terms and tools which will help provide intuition for the relations to some of the program transformation techniques to follow. I illustrate how mining functional dependencies (FDs) in websites helps expand input for out-of-turn interaction.

Identifying structural relationships in data-intensive websites, in areas such as e-commerce (e.g., Amazon.com), digital libraries (e.g., CITIDEL at citidel.org), and scientific computing (e.g., GAMS at gams.nist.gov) is becoming an increasingly popular precursor to customizing information access [ABS00]. Deploying out-of-turn interaction involves addressing some practical considerations, including the identification of such relationships. Specifically, when the targeted website contains dependencies between and across the levels of its DAG model, an out-of-turn interaction can result in the system soliciting information from the user which can be inferred from the previous input. For instance, consider a user's interaction with the Kelly Blue Book (KBB at kbb.com), an automobile website which progressively solicits for automobile attributes, in an order pre-determined by the site designer, and forces users to communicate those attributes in this manner to access a vehicle webpage. KBB is modeled by a mutually-exclusive tree.

Fig. 4.13 illustrates an out-of-turn interaction episode with KBB to motivate FDs on the web. In the first window, when the site solicits for vehicle category, the user responds with 'Civic' out-of-turn, by speaking to the browser. This causes all but one vehicle category to be pruned out. Notice that when the site reclaims the initiative, it asks the user to make a selection for vehicle category (because that attribute is still unspecified), even though there is only one option left (Sedan). Next the user clicks the 'Sedan' (second window) and 'Honda' (third window) hyperlinks in-turn to arrive at the content page for a Honda Civic (fourth window). Asking the user to make a choice where it is obvious, or forcing her to click through a long and painstaking series of webpages, each containing only a single hyperlink (windows two and three), may confuse or frustrate her. When the user says 'Civic' out-of-turn I can infer 'Sedan Honda' by functional dependency, e.g., 'Civic → {Sedan, Honda},' transparently expand the user's input to 'Civic Sedan Honda,' and communicate these terms of information-seeking to the system in one stroke. I call FDs of this type *positive-path FDs*. The 'Civic → Honda' FD asserts that all of the paths through the site involving 'Civic' also involve 'Honda.' Clearly, the reverse does not hold as Honda makes several automobile models. Detecting such dependencies between the site's levels (in this case, vehicle-type, -maker, and -model) and leveraging them for input expansion delivers a compelling experience to the end-user. When such dependencies are employed, it is important to provide real-time feedback to users so that the information contained on the r.h.s. of the FD is not lost. This can be done by collecting the r.h.s. of each FD triggered and augmenting



Kelley Blue Book - Build a Car - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Select a Vehicle Category
Sedan
Luxury
Coupe
Pickup
Convertible
Sport Utility
Wagon
Van/Minivan
Hatchback
Civic
Input so far:
Internet
⇓
Kelley Blue Book - Build a Car - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Select a Vehicle Category
Sedan
Input so far: Civic
Internet
⇓
Kelley Blue Book - Sedans - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Select a Make
Honda
Input so far: Sedan Civic
Internet
⇓
Kelley Blue Book - 2004 Civic - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Sedan • Honda • 2004 • Civic
Input so far: Sedan Honda Civic
Internet

Figure 4.13: A cumbersome out-of-turn interaction experience with the Kelly Blue Book.

the leaf webpage with this information. Alternatively, I could enhance the 'Input so far:' label in the browser's status bar at each step in the interaction to incrementally include the r.h.s. of any fired FDs. Notice that firing such FDs affects the stageable interaction sequences and thus the realizable (browsing and out-of-turn) interaction paradigms. Thus, I may have to update my definition of an interaction sequence so that it can be defined over a subset of an interaction set.

Using FDs to expand user input ultimately creates invisible *shortcuts* through the website for the user. For example, saying 'Washington, DC' out-of-turn at the top level of PVS, expands to 'Washington, DC House Democrat District-at-large' and directly reaches the webpage of an individual congressional official. In levelwise sites, only a cursory understanding of the underlying domain is necessary to manually identify a majority of the FDs fitting the re-occurring '[model] → [make]' FD schema (e.g., 'Civic → Honda') in KBB. FDs are less salient in sites that are not-levelwise, such as large taxonomies of links to web resources (e.g., Yahoo! and ODP). Thus, techniques from association rule mining, especially those designed for the web [EV03, MCS00, MAB00, SCDT00], become important and applicable in both classes.

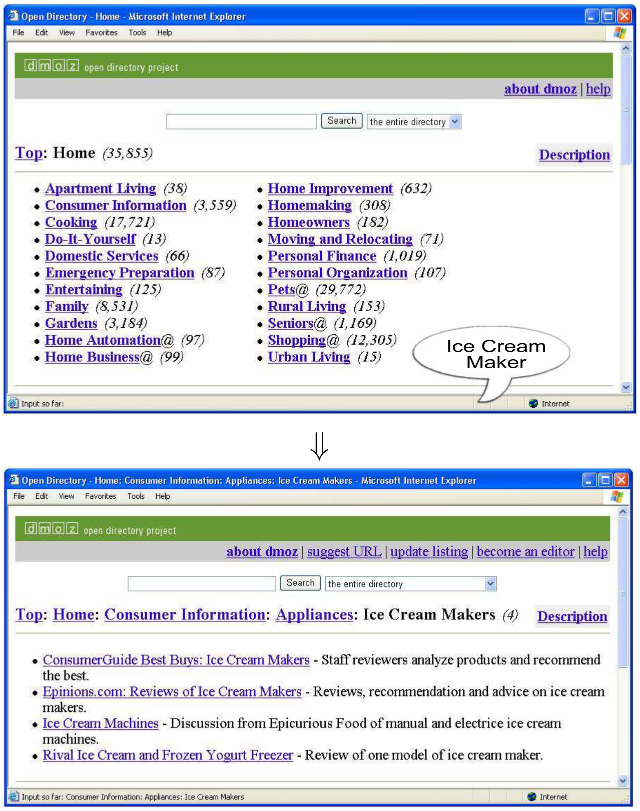
4.8.1 Tools for Mining Positive FDs

Positive-path FDs

Discovering positive-path FDs, such as those described above, entails identifying pairs of hyperlink labels where *all* of the paths through the site involving one also involve the other. Then when a user supplies out-of-turn input, I consult the set of FDs to answer the question: 'What terms in the complete set of terms, $TE\ (D)$, lie along each path involving a hyperlink labeled with the out-of-turn input?' An instance of such an FD in PVS is 'Senior seat → Senate.' Mining such FDs takes $O((|TE\ (D)| \times |TE\ (D)|) - |TE\ (D)|)$ time, discounting FDs such as 'x → x.' It is important to note that the number of positive-path FDs in a site changes after every (in-turn or out-of-turn) interaction because the number of paths remaining through the site is reduced as a result of each interaction. Thus, a new set of FDs must be mined at each step. However, since positive-path FDs can only exist among terms which co-occur on a path, I need not examine $TE\ (D)$ terms per term. Rather I can make use of a path-term-co-occurrence set and examine only the terms which co-occur on a path with the particular term (corresponding to the out-of-turn input). This approach implies computing a path-term-co-occurrence set after the user supplies an out-of-turn input and using it to discover any positive-path FDs relevant to the user's interaction. *Mine positive-path-FD set* is total function $mFDs_{pp} : (\mathcal{D} \times \mathcal{T}) \rightarrow P\ (T)$ which given D and a term $t_i \in T = TE\ (D)$ returns a term-co-occurrence set. It is defined as

$$mFDs_{pp}\ (D,\ t_i) = \bigcup_{t_j\ \in\ BPTC\ (D,\ t_i)} \{t_j\} \quad \text{s.t. } GS\ (t_i,\ SQ\ (D)) \subseteq GS\ (t_j,\ SQ\ (D)).$$

This function finds FDs of the form $t_i \rightarrow t_j$. A *positive-path-FD set* of D is any term-co-occurrence set returned from $mFDs_{pp}$. The complete set of positive-path FDs which hold in the DAG shown in Fig. 4.10 (left) is


Open Directory - Home - Microsoft Internet Explorer
File Edit View Favorites Tools Help
dmoz open directory project
about dmoz | help
Search
the entire directory
Description
Ice Cream Maker
Input so far:
Internet
⇓
Open Directory - Home: Consumer Information: Appliances: Ice Cream Makers - Microsoft Internet Explorer
File Edit View Favorites Tools Help
dmoz open directory project
about dmoz | suggest URL | update listing | become an editor | help
Search
the entire directory
Description
Input so far: Consumer Information: Appliances: Ice Cream Makers
Internet

Top: Home *(35,855)*

- Apartment Living *(38)*
- Consumer Information *(3,559)*
- Cooking *(17,721)*
- Do-It-Yourself *(13)*
- Domestic Services *(66)*
- Emergency Preparation *(87)*
- Entertaining *(125)*
- Family *(8,531)*
- Gardens *(3,184)*
- Home Automation@ *(97)*
- Home Business@ *(99)*
- Home Improvement *(632)*
- Homemaking *(308)*
- Homeowners *(182)*
- Moving and Relocating *(71)*
- Personal Finance *(1,019)*
- Personal Organization *(107)*
- Pets@ *(29,772)*
- Rural Living *(153)*
- Seniors@ *(1,169)*
- Shopping@ *(12,305)*
- Urban Living *(15)*

Top: Home: Consumer Information: Appliances: Ice Cream Makers *(4)*

- ConsumerGuide Best Buys: Ice Cream Makers - Staff reviewers analyze products and recommend the best.
- Epinions.com: Reviews of Ice Cream Makers - Reviews, recommendation and advice on ice cream makers.
- Ice Cream Machines - Discussion from Epicurious Food of manual and electrice ice cream machines.
- Rival Ice Cream and Frozen Yogurt Freezer - Review of one model of ice cream maker.

Figure 4.14: Using FDs to expand out-of-turn user input. The resulting effect relieves the user from manually clicking through a series of hyperlinks which all lead to the same webpage.

$$\begin{aligned}\{&\text{Ramakrishnan} \rightarrow \text{Computer Science},\\ &\text{Adjerid} \rightarrow \text{Computer Science},\\ &\text{Beattie} \rightarrow \text{Mathematics},\\ &\text{Green} \rightarrow \text{Mathematics}\}.\end{aligned}$$

Notice that no positive-path FDs hold in the DAG shown in Fig. 4.10 (right).

Using Positive-path FDs for Input Expansion

I use the mined FDs to drive input expansion for out-of-turn interaction. Fig. 4.14 illustrates an out-of-turn interaction with the 'Home' sub-tree of the ODP hierarchy. In the top window, the user communicates 'Ice Cream Maker' out-of-turn by speaking to the browser, producing the page in the bottom window. This interaction automatically triggers two FDs: 'Ice Cream Makers → {Appliances, Consumer Information}' and has the same effect as a shortcut to the page, shown in the bottom window, nested deeper in the taxonomy. There are 452,690 browsing interaction sequences through the ODP taxonomy. Table 4.6[1] provides various frequencies of relevant items, including FDs, in selected sub-trees of ODP.

---

[1] FDs were mined from the ODP structure RDF (Resource Description Framework) dump (publicly available at http://rdf.dmoz.org) generated on February 9, 2004. Frequencies were tallied without consideration of crosslinks. In other words, I analyzed a tree model of ODP.

Sub-tree	**seq**	**t**	**(t, t)**	**cand. (t, t)**	**FDs**	**%**
Adult	6,014	2,129	4,530,512	34,163	8,711	25%
Business	7,560	4,082	16,658,642	57,290	10,241	18%
Computers	6,217	3,618	13,086,306	48,776	11,811	24%
Games	7,898	6,335	40,125,890	73,029	40,903	56%
Health	5,155	3,152	9,931,952	38,794	11,227	29%
Home	1,761	1,565	2,447,660	15,289	4,756	31%
Kids and Teens	2,339	2,270	5,150,630	25,012	12,450	50%
News	415	278	77,006	2,694	752	28%
Recreation	8,368	3,512	12,330,632	53,380	14,083	26%
Reference	8,076	6,159	37,927,122	89,913	33,819	38%
Science	8,600	7,441	55,361,040	98,689	45,303	46%
Shopping	3,984	2,987	8,919,182	32,725	12,382	38%
Society	21,105	11,076	122,666,700	171,664	61,001	36%
Sports	14,609	7,171	51,416,070	108,899	33,511	31%

Table 4.6: Descriptive statistics of positive-path FDs mined in selected top-level categories of ODP. The column labeled ‘seq’ contains the browsing interaction sequence frequencies. The column labeled ‘t’ provides the number of unique terms in each category. The column labeled ‘(t, t)’ contains the number of (term, term) pairs in the sub-tree, discounting those involving only one term (i.e., (t × t) − t). The column labeled ‘cand. (t, t)’ provides the frequencies of candidate FDs (term, term) pairs (i.e., the sum of the cardinality of every path-term-co-occurrence set in the category). The column labeled ‘FDs’ provides the number of positive-path FDs in the sub-tree. Lastly, the column labeled ‘%’ contains the percentage of FDs among the candidate FDs (i.e., FDs ÷ cand. (t, t)).

Positive-leaf FDs

Positive-leaf FDs are more general than positive-path FDs in that all positive-path FDs also are positive-leaf FDs. Discovering positive-leaf FDs entails identifying pairs of hyperlink labels where *all* of the leaves classified under paths involving one label also are classified under paths involving the other. Then when a user supplies out-of-turn input, I consult the set of FDs to answer the question: 'What terms in the complete set of terms, $TE\ (D)$, lie along paths which classify all the leaves reachable by paths involving a hyperlink labeled with the out-of-turn input?' As with positive-path FDs, mining these FDs takes $O((|TE\ (D)| \times |TE\ (D)|) - |TE\ (D)|)$ time, discounting FDs such as 'x → x.' Again, the number of positive-leaf FDs in a site changes after every interaction because the number of leaves remaining in the site is reduced as a result of each interaction. Thus, a new set of positive leaf FDs must be mined at each step. Akin to the previous discussion, since positive-leaf FDs can only exist among terms which co-occur w.r.t. leaves, I need not examine $TE\ (D)$ terms per term. Rather I can make use of a leaf-term-co-occurrence set and examine only the terms which co-occur with a particular term (corresponding to the out-of-turn input) w.r.t. leaves. This approach implies computing a leaf-term-co-occurrence set after the user supplies an out-of-turn input and using it to discover any positive-leaf FDs relevant to the user's interaction. *Mine positive-leaf-FD set* is total function $mFDs_{pl} : (\mathcal{D} \times \mathcal{T}) \rightarrow P\ (T)$ which given D and a term $t_i \in T = TE\ (D)$ returns a term-co-occurrence set. It is defined as

$$mFDs_{pl}\ (D,\ t_i) = \bigcup_{t_j\ \in\ BLTC\ (D,\ t_i)} \{t_j\} \quad s.t.\ FP\ (D,\ t_i) \subseteq FP\ (D,\ t_j).$$

This function mines FDs of the form $t_i \rightarrow t_j$. A *positive-leaf-FD set* of D is any term-co-occurrence set returned from $mFDs_{pl}$. The complete set of positive-leaf FDs which hold in the DAG D shown in Fig. 4.10 (left) is

{Ramakrishnan → Computer Science,
Adjerid → Computer Science,
Adjerid → Mathematics,
Adjerid → Beattie,
Beattie → Computer Science,
Beattie → Mathematics,
Beattie → Adjerid,
Green → Mathematics}.

Notice that no positive-leaf FDs hold in the DAG shown in Fig. 4.10 (right).

Lemma 10 *Any positive-path FD in a* DAG *D is a positive-leaf FD in D.*

> **Outline of Proof**: Assume 'x → y' is a positive-path FD in a DAG D. This means that all of the paths from the root of D to a leaf which involve x also involve y. This means that all of the leaves reachable via a path through D involving x are reachable via a path through D involving y. Thus, all of the leaves classified by paths involving x also are classified under paths involving y, thereby satisfying the definition of a positive-leaf FD. ∴ Lemma 10 is true. _ □

Corollary 4 *Any positive-path FD set w.r.t. a term* t_i *in a* DAG D *is a subset of the positive-leaf FD set w.r.t.* t_i *in* D*. In other words,* $mFDs_{pp}\ (D,\ t_i) \subseteq mFDs_{pl}\ (D,\ t_i)$*.*

> **Outline of Proof**: Follows directly from Lemma 10. ______________ □

I leave it to the reader to convince thyself that the converse of Lemma 10 (e.g., 'Any positive-leaf FD is a positive-path FD') is not true. However, we have:

Lemma 11 *If a* DAG D *is a tree, then any positive-leaf FD in* D *is a positive-path FD in* D*.*

> **Outline of Proof**: Let D be a tree. Thus, there is one simple path from the root of D to each of its leaves, by the definition of a tree. Assume 'x → y' is a positive-leaf FD in D. This means that all of the leaves reachable via a path through D involving x are reachable via a path through D involving y. Thus, all of the leaves classified by paths involving x also are classified under paths involving y. Since there is only one path from the root to each leaf in D, this means that y must co-occur with x on each path through D involving x, thereby satisfying the definition of a positive-path FD. ∴ Lemma 11 is true. ______ □

Corollary 5 *If a* DAG D *is a tree, than any positive-path FD set w.r.t. a term* t_i *in* D *equals the positive-leaf FD set w.r.t.* t_i *in* D*. In other words, if* D *is a tree, then* $mFDs_{pp}\ (D,\ t_i) = mFDs_{pl}\ (D,\ t_i)$*.*

> **Outline of Proof**: Follows directly from Lemmas 10 and 11. __________ □

Corollary 6 *If a* DAG D *is a tree, then the complete set of positive-path FDs in* D *equals the complete set of positive-leaf FDs in* D*.*

> **Outline of Proof**: Follows directly from Corollary 5. ______________ □

Notice that Lemma 11 uses *if*, not *iff*; I leave it to the reader to convince thyself that the converse of Lemma 11 (e.g., 'If any positive-leaf FD in a DAG D is a positive-path FD in D, then DAG D is a tree') is not true.

Identifying and employing instances of positive (-path and -leaf) FDs are helpful for usability purposes, but not necessary for realizing out-of-turn interaction. What is non-intuitive, however, is that a related concept, *negative* (-path and -leaf) FDs, turns out to be helpful in developing an alternate program transformation technique for out-of-turn interaction as well as specializations of it for the mutually-exclusive classes[2] of hierarchical hypermedia. Specifically, partially evaluating a programmatic representation of a website w.r.t. the variables modeling the terms on the r.h.s. of a *negative-path or -leaf* FD set assigned zero prunes the site.

---

[2] When I use the phrase 'mutually-exclusive classes,' I am referring to a 'mutually-exclusive tree' and a 'weak mutually-exclusive DAG,' not a general connotation to two (or more) classes that are mutually exclusive with each other.

4.8.2 Tools for Mining Negative FDs

Negative-path FDs

Identifying negative-path FDs entails identifying pairs of hyperlink labels where *none* of the paths through the site involving one label involve the other. Identifying negative-path FDs involves answering the following question: 'What terms in the complete set of terms, $TE\ (D)$, *never* lie along any of the paths involving a hyperlink labeled with a particular term?' An example of such an FD in PVS is 'Senior seat → ¬ House.' Intuitively, this means that none of the paths through the site involving the term 'Senior seat' involve the term 'House.' *Mine negative-path-FD set* is total function $mFDs_{np} : (\mathcal{D} \times \mathcal{T}) \rightarrow P\ (T)$ which given D and a term $t_i \in T = TE\ (D)$ returns a term-co-occurrence set. It is defined as

$$mFDs_{np}\ (D,\ t_i) = TE\ (D) - \{t_i\} - BPTC\ (D,\ t_i).$$

A *negative-path-FD set* of D is any term-co-occurrence set returned from $mFDs_{np}$. The complete set of negative-path FDs which hold in the DAG shown in Fig. 4.10 (left) is

{Computer Science → ¬ Mathematics,
Computer Science → ¬ Beattie,
Computer Science → ¬ Green,
Mathematics → ¬ Computer Science,
Mathematics → ¬ Ramakrishnan,
Mathematics → ¬ Adjerid,
Ramakrishnan → ¬ Mathematics,
Ramakrishnan → ¬ Adjerid,
Ramakrishnan → ¬ Beattie,
Ramakrishnan → ¬ Green,
Adjerid → ¬ Mathematics,
Adjerid → ¬ Ramakrishnan,
Adjerid → ¬ Beattie,
Adjerid → ¬ Green,
Beattie → ¬ Computer Science,
Beattie → ¬ Ramakrishnan,
Beattie → ¬ Adjerid,
Beattie → ¬ Green,
Green → ¬ Computer Science,
Green → ¬ Ramakrishnan,
Green → ¬ Adjerid
Green → ¬ Beattie}.

Likewise, the complete set of negative-path FDs which hold in the DAG shown in Fig. 4.10 (right) is

{California → ¬ Virginia,
Virginia → ¬ California,
House → ¬ Senate,
Senate → ¬ House}.

Negative-leaf FDs

Identifying negative-leaf FDs entails identifying pairs of hyperlink labels where *none* of the leaves classified under paths involving one label are classified under paths involving the other. Identifying negative-leaf FDs involves answering the following question: 'What terms in the complete set of terms, $TE\ (D)$, do not lie along paths from the root of D to its leaves which classify any of the leaves reachable by paths involving a hyperlink labeled with a particular term?' An example of such an FD in PVS is 'Senior seat → ¬ House.' Unlike the discussion of negative-path FDs, this type of FD must be interpreted as 'none of the leaves classified by paths involving the term 'House' are classified by the paths involving the term 'Senior seat.' *Mine negative-leaf-FD set* is total function $mFDs_{nl} : (\mathcal{D} \times \mathcal{T}) \rightarrow P\ (T)$ which given D and a term $t_i \in T = TE\ (D)$ returns a term-co-occurrence set. It is defined as

$$mFDs_{nl}\ (D,\ t_i) = TE\ (D) - \{t_i\} - BLTC\ (D,\ t_i).$$

A *negative-leaf-FD set* of a DAG D is any term-co-occurrence set returned from $mFDs_{nl}$. The complete set of negative-leaf FDs which hold in the DAG shown in Fig. 4.10 (left) is

{Computer Science → ¬ Green,
Mathematics → ¬ Ramakrishnan,
Ramakrishnan → ¬ Mathematics,
Ramakrishnan → ¬ Adjerid,
Ramakrishnan → ¬ Beattie,
Ramakrishnan → ¬ Green,
Adjerid → ¬ Ramakrishnan,
Adjerid → ¬ Green,
Beattie → ¬ Ramakrishnan,
Beattie → ¬ Green,
Green → ¬ Computer Science,
Green → ¬ Ramakrishnan,
Green → ¬ Adjerid
Green → ¬ Beattie}.

The complete set of negative-leaf FDs which hold in the DAG D shown in Fig. 4.10 (right) is (the complete set of negative-path FDs which hold in the D)

{California → ¬ Virginia,
Virginia → ¬ California,
House → ¬ Senate,
Senate → ¬ House}.

Note that the ¬ is always on the r.h.s. of a negative FD. Notice further that negative (-path and -leaf) and positive (-path and -leaf) FDs are not complements of each other.

Lemma 12 *Any negative FD $x \rightarrow \neg\, y$ considered here also implies $y \rightarrow \neg\, x$.*

Outline of Proof:

Negative-path FD: Any negative-path FD $x \rightarrow \neg\, y$ considered here also implies $y \rightarrow \neg\, x$.

Assume 'x → ¬ y' is a negative-path FD in a DAG D. By the definition of a negative-path FD, *none* of the paths involving x from the root of D to a leaf involve y. This relationship is symmetric, and thus, *none* of the paths involving y from the root of D to a leaf involve x. Thus, by the definition of a negative-path FD, 'y → ¬ x' is a negative-path FD in D.

Negative-leaf FD: Any negative-leaf FD $x \rightarrow \neg\, y$ considered here also implies $y \rightarrow \neg\, x$.

Assume 'x → ¬ y' is a negative-leaf FD in a DAG D. By the definition of a negative-leaf FD, *none* of the leaves classified under paths involving x in D are classified under paths involving y. This relationship is symmetric, and thus, *none* of the leaves classified under paths involving y from the root of D to a leaf are classified under paths involving x. Thus, by the definition of a negative-leaf FD, 'y → ¬ x' is a negative-leaf FD in D.

∴ Lemma 12 is true. ______________________________ □

Lemma 13 *Any negative-leaf FD in a* DAG D *is a negative-path FD in* D.

Outline of Proof: Assume 'x → ¬ y' is a negative-leaf FD in a DAG D. This means that y lies along none of the paths which classify the leaves classified under paths involving x. If y does not lie along a path which classifies any of the leaves classified under paths involving x, then y can never co-occur on a path involving x from the root of D to a leaf, because if it did, then it would lie along a path which classifies one or more of the leaves classified under paths involving x, thereby satisfying the definition of a negative-path FD. ∴ Lemma 13 is true. . □

Notice that the claim in Lemma 13 is the contrapositive of the claim in Lemma 10.

Corollary 7 *Any negative-leaf-FD set w.r.t. a term* t_i *in* D *is a subset of the negative-path-FD set w.r.t.* t_i *in* D. *In other words,* $mFDs_{nl}\ (D,\ t_i) \subseteq mFDs_{np}\ (D,\ t_i)$.

Outline of Proof: Follows directly from Lemma 13. ______________ □

I leave it to the reader to convince thyself that the converse of Lemma 13 (e.g., 'Any negative-path FD is a negative-leaf FD') is not true.

Lemma 14 *If a* DAG D *is a tree, then any negative-path FD in* D *is a negative-leaf FD in* D.

Outline of Proof: Let a $\mathcal{D}$ D be a tree. Thus, there is one simple path from the root of D to each of its leaves, by the definition of a tree. Assume 'x → ¬ y' is a negative-path FD in D. This means that y co-occurs with x on none of the paths involving x from the root of D to a leaf. Since there is only one path from the root of D to each leaf, none of the leaves classified by paths involving x are classified under paths involving y, because otherwise y would have to co-occur with x on one path, and I have assumed that it does not by supposing that 'x → y' holds. This means that none of paths involving y in D classify any of the leaves classified under paths involving x in D, thereby satisfying the definition of a negative-leaf FD. ∴ Lemma 14 is true. ________________ □

Corollary 8 *If a* DAG D *is a tree, then any negative-path-FD set w.r.t. a term* t_i *in* D *equals the negative-leaf-FD set w.r.t.* t_i *in* D*. In other words, if* D *is a tree, then* $mFDs_{np}\ (D,\ t_i) = mFDs_{nl}\ (D,\ t_i)$*.*

Outline of Proof: Follows directly from Lemmas 13 and 14. ________ □

Corollary 9 *If a* DAG D *is a tree, then the complete set of negative-path FDs in* D *equals the complete set of negative-leaf FDs in* D*.*

Interestingly, the classes of FDs I introduced above not only can be exploited by program transformation techniques, but also can be mined via them.

4.9 Mining FDs by Program Transformation

Positive-path FDs

The following program transformation technique can mine a positive-path-FD set w.r.t. a particular term.

$$[\![\texttt{ppfd}]\!][\texttt{P}_\texttt{D},\ \texttt{input}] = \bigcup_{\texttt{t}_\texttt{i}\ \in\ [\![\texttt{bptc}]\!][\texttt{P}_\texttt{D},\ \texttt{input}]} \{\texttt{t}_\texttt{i}\},$$

$$\text{if } ([\![\texttt{zoom}]\!][\texttt{P}_\texttt{D},\ \texttt{input}]\ \cap\ [\![\texttt{zoom}]\!][\texttt{P}_\texttt{D},\ \texttt{t}_\texttt{i}]) = [\![\texttt{zoom}]\!][\texttt{P}_\texttt{D},\ \texttt{input}],$$

where $[\![\texttt{bptc}]\!] = [\![\texttt{te}]\!][[\![\texttt{sp}]\!][\texttt{P}_\texttt{D},\ \texttt{input}]] - \texttt{input}$ is the program transformation technique I associate with *BPTC*. The reader will notice that the above expression closely reflects the textual definition of a positive-path FD. Intuitively, this program transformation technique mines positive-path FDs of the form 'input → $\mathrm{t_i}$.'

Positive-leaf FDs

I can mine a positive-leaf-FD set analogously.

$$[\![\texttt{plfd}]\!][\texttt{P}_\texttt{D},\ \texttt{input}] = \bigcup_{\texttt{t}_\texttt{i}\ \in\ [\![\texttt{bltc}]\!][\texttt{P}_\texttt{D},\ \texttt{input}]} \{\texttt{t}_\texttt{i}\},$$

$$\text{if } ([\![\texttt{zoom}]\!][\texttt{P}_\texttt{D},\ \texttt{input}]\ \cap\ [\![\texttt{zoom}]\!][\texttt{P}_\texttt{D},\ \texttt{t}_\texttt{i}]) = [\![\texttt{zoom}]\!][\texttt{P}_\texttt{D},\ \texttt{input}],$$

where $[\![\texttt{bltc}]\!] = [\![\texttt{te}]\!][[\![\texttt{zoom}]\!][\texttt{P}_\texttt{D},\ \texttt{input}]] - \texttt{input}$ is the program transformation technique I associate with *BLTC*. Similarly, this program transformation technique mines positive-leaf FDs of the form 'input → $\mathrm{t_i}$.' Since identifying negative FDs does not involve containment, program transformation techniques for doing so are more simplistic than those for positive FDs.

Negative-path FDs

Mining a negative-path-FD set entails using $[\![\mathtt{te}]\!]$ to extract all of the conditional variables from the programmatic complement of $[\![\mathtt{sp}]\!][\mathtt{P_D},\ \mathtt{input}]$.

$$[\![\mathtt{npfd}]\!][\mathtt{P_D},\ \mathtt{input}] = [\![\mathtt{te}]\!][\mathtt{P_D} - [\![\mathtt{sp}]\!][\mathtt{P_D},\ \mathtt{input}]],$$

where the '$-$' (minus sign) means complement.

Negative-leaf FDs

Similarly, discovering a negative-leaf-FD set entails using $[\![\mathtt{te}]\!]$ to extract all of the conditional variables from the complement of $[\![\mathtt{zoom}]\!][\mathtt{P_D},\ \mathtt{input}]$.

$$[\![\mathtt{nlfd}]\!][\mathtt{P_D},\ \mathtt{input}] = [\![\mathtt{te}]\!][\mathtt{P_D} - [\![\mathtt{zoom}]\!][\mathtt{P_D},\ \mathtt{input}]]$$

4.10 A Duality in Uses of Program Slicing

I have used partial evaluation and program slicing as pruning operators. There is a relationship and tradeoff between these two program transformations in the context of my work. Specifically, one can think of program slicing as a transformation for

1. directly pruning the website, *or*
2. extracting information about *what* to prune from the website and using this information with partial evaluation to conduct the same site pruning as in (1).

Table 4.4 (second row) illustrates how program slicing can play role (1) above to realize interpretation 1 of out-of-turn interaction. In the previous section, I show that program slicing can be used to mine FDs for my purposes, including negative path- and leaf-FDs. Notice that the members of the negative-leaf-FD set w.r.t. some relevant term t label only the edges to be pruned from a graph model of a website when the user communicates term t out-of-turn. Therefore, partially evaluating w.r.t. each variable modeling each term in the negative-leaf-FD set of a particular term set to zero is sufficient to prune the undesired paths from a website to realize interpretation 1 of out-of-turn interaction.

The following expression captures these two methods of realizing interpretation 1 of out-of-turn interaction. It also captures the dual role played by program slicing (role (1) above is played by $[\![\mathtt{zoom}]\!]$ on the l.h.s. and role (2) above is played by $[\![\mathtt{nlfd}]\!]$ on the r.h.s.).

$$[\![\mathtt{mix}]\!][[\![\mathtt{zoom}]\!][\mathtt{P_D},\ \mathtt{input}], \mathtt{input} = \mathtt{1}] \equiv [\![\mathtt{mix}]\!][\mathtt{P_D},\ [[\![\mathtt{nlfd}]\!]\ [\mathtt{P_D},\ \mathtt{input}]] = \mathtt{0},\ \mathtt{input} = \mathtt{1}],$$

where the notation

$$[[\![\mathtt{nlfd}]\!]\ [\mathtt{P_D},\ \mathtt{input}]] = \mathtt{0},$$

used within the expression, denotes 'the assignment of zero to each variable modeling each member of the negative-leaf-FD set of the term modeled by the variable `input`.'

The reader may have noticed that I use the word 'sufficient' above, rather than 'necessary.' This is because all terms labeling edges along the tails of paths to be pruned, when the user communicates term *t* out-of-turn will be members of *t*'s negative-leaf-FD set. However, to remove that tail, I need not partially evaluate w.r.t. all variables modeling the terms along that tail set to zero. I need only partially evaluate w.r.t. the variable modeling the term with the minimum depth set to zero. Such a partial evaluation will remove the remainder of the path by default. Intuitively, this means that I need only consider the members of the negative-leaf-FD which have the minimum depth in each path through the DAG. Considering this optimization compromises the clarity of the equivalence which I have just outlined as well as the duality in uses of program slicing. Therefore, I do not incorporate it into my presentation and expression.

Also notice that when dealing with trees, I can replace $[\![\texttt{nlfd}]\!]$ with $[\![\texttt{npfd}]\!]$ in the above expression by Corollary 8.

Now let us consider how I might develop an analogous equivalence to the program transformation technique given in Table 4.4 (third row) for interpretation 2 of out-of-turn interaction. Recall to support interpretation 2 I replace $[\![\texttt{zoom}]\!]$ with $[\![\texttt{sp}]\!]$ on the l.h.s. of the above equivalence expression. Thus, it natural to consider replacing $[\![\texttt{nlfd}]\!]$ with $[\![\texttt{npfd}]\!]$ on the r.h.s. of the above expression to form an analogous equivalence expression for interpretation 2 of out-of-turn interaction. However, upon careful examination we see that the r.h.s. is not equal to the l.h.s., and thus does not realize interpretation 2 of out-of-turn interaction:

$$[\![\texttt{mix}]\!][[\![\texttt{sp}]\!][\texttt{P}_\texttt{D},\ \texttt{input}], \texttt{input} = \texttt{1}] \not\cong [\![\texttt{mix}]\!][\texttt{P}_\texttt{D},\ [[\![\texttt{npfd}]\!]\ [\texttt{P}_\texttt{D},\ \texttt{input}]] = \texttt{0},\ \texttt{input} = \texttt{1}]$$

This is because if $\texttt{P}_\texttt{D}$ models a non-tree DAG, then my use of partial evaluation, as described above, to remove undesired paths may remove leaves which lie at the end of desired paths! For example, consider partially evaluating a programmatic representation of the website illustrated in Fig. 4.10 (left) w.r.t. the negative-path-FD set of 'Adjerid.' You will notice that this transformation will remove the source (leaf) of the edge labeled 'Adjerid' which is unfaithful to interpretation two. However, the equivalence expression for interpretation 1 can serve as a surrogate equivalence expression for interpretation 2 of out-of-turn interaction with trees (by Corollary 5), where, $[\![\texttt{zoom}]\!]$ and $[\![\texttt{nlfd}]\!]$ can be replaced by $[\![\texttt{sp}]\!]$ and $[\![\texttt{npfd}]\!]$, respectively.

I am unable to develop equivalence expressions for the generalizations of these interpretations of out-of-turn interaction here due to their template-oriented nature (i.e., the presence of $[\![\texttt{SL}]\!]$).

Studying this duality reveals that there might be simpler methods for discerning what branches must be removed, in specialized DAG classes. The following program transformation technique realizes out-of-turn interaction, but is specifically tailored toward taking advantage of the mutually-exclusive property in DAGs and trees.

$$[\![\texttt{dead-code}]\!][[\![\texttt{mix}]\!][\texttt{P}_\texttt{D},\ [[\![\texttt{tle}]\!][\texttt{P}_\texttt{D},\ \texttt{input}] - \texttt{input}] = \texttt{0},\ \texttt{input} = \texttt{1}]]$$

The $[\![\texttt{dead–end}]\!]$ transformation represents a dead-code detection and elimination transformation [CXY01, WZ91]. This approach illustrates that partial evaluation is a specialization of program slicing w.r.t. programmatic representations of levelwise, mutually-exclusive DAG models of websites. Here $[\![\texttt{tle}]\!]$ supports TLE programmatically. This transformation technique involves extracting all the structural variables at a particular level of nesting. The transformation technique $[\![\texttt{tle}]\!]$ is intended to be *polymorphic* in that its input can be either given extensionally as a level number or intensionally as a term that occurs at the desired level. Notice that the latter usage is unambiguous only in the case of mutually-exclusive classes (which are the only classes it is used for here).

Other researchers have echoed similar connections between partial evaluation and slicing; the two techniques have been shown to yield similar results in some situations and different results in others [RT96].

4.11 Discussion

I gained insight into the graph-theoretic formalisms developed here by implementing them in ML (*Meta*-Language) using SML/NJ. I explored the feasibility of the website – program associations I developed by conducting verification experiments on programmatic representations of interaction with PVS and VTTT using program transformation software systems. The 'set calculator' of CodeSurfer [ART03], a program slicing system, was especially helpful for performing set-theoretic operations over slices. In Appendix C, I discuss the details of the program transformation systems I used. Before concluding this chapter, I consider the possibility of multiple terms per utterance, an important ingredient for personalized interaction.

4.11.1 Enriching the Out-of-turn Paradigm: Multiple Terms per Utterance

I refer to a set of terms as an *utterance*. While the previous chapter illustrated communicating more than one partial (out-of-turn) input in one stroke (e.g., 'Democrat, Senate'), for ease of presentation, I developed the formalism here around examples which only involve the communication of one term per utterance. Formally, multiple terms per utterance simply means that two or more terms can have the same arrival time. While permitting multiple terms per utterance alters the interaction I afford the user, it does not affect the formalism developed here, largely due to Lemma 1 (commutativity). Thus, to accommodate multiple terms per utterance, I can re-define interpretations of out-of-turn as follows:

$$OOT_1\ (D,\ u) = OOT_1\ (\cdots OOT_1\ (OOT_1\ (D,\ t_1),\ t_2) \cdots,\ t_n), \tag{4.5}$$

where u denotes an utterance having n terms and each OOT_1 on the r.h.s. refers to Equation 4.1. Notice that if $OOT_1\ (D,\ u)$ returns a DAG containing one vertex v and no edges, then the utterance u is complete information and v is terminal information.

Consider how I now might update my derivation of a closed formula for the size of an out-of-turn interaction paradigm $\mathcal{P}$. In discrete mathematics [KS99], $s\ (m)$ is the set of all partitions of a set of size m into non-empty subsets, where m is a positive integer. In addition, $s\ (m,\ n)$ is the set of all partitions of a set of size m into exactly n non-empty subsets, where n is a positive integer and $n \leqslant m$. The *Bell number* of a set of size m is $B\ (m) = |s(m)|$. The *Stirling number* of a set of size m is $S\ (m,\ n) = |s(m,n)|$. It follows that $B\ (m) = \sum_{n=1}^{m} S\ (m,\ n)$. Thus, $|\mathcal{P}|$ corresponds to the sum, over all interaction sequences in $SQ\ (D)$, of the sum of the number all permutations of each element of $s\ (|i|)$, i.e., the sum of the sum of the number of all the permutations of each term partition of each interaction set. Therefore, I can re-define $SoOP$ as

$$|\mathcal{P}| \ = \ SoOP\ (D) \ = \sum_{I_i\ \in\ SQ\ (D)} \ \sum_{n=1}^{|gIS\ (I_i)|} n! \ \times \ S\ (|gIS\ (I_i)|,\ n).$$

Notice that this formula accounts for valid utterances containing more than two terms and makes no assumption on the consistency of the dialog length across all sequences. Once again, notice that processing user input can be likened to automatic input expansion.

Keep in mind that the interpretations for out-of-turn interaction presented here are not exhaustive, especially since I defined an open-ended generalized interpretation of out-of-turn interaction. For instance, I might expand the scope of addressable out-of-turn information by modeling the terms on the leaf webpages (i.e., terminal information) and making them available as out-of-turn input for the user to supply. In such a case, I might implement SL using a text-based search engine. Nevertheless, the use of a general program transformation, such as program slicing, suggests that alternate interpretations also can be accommodated. In addition, other interaction techniques and paradigms can be similarly supported.

When a given (programmatic representation, transformation technique) pair is deemed inadequate to achieve the desired form of personalization, I have two choices. One is to keep the representation fixed and investigate alternative program transformation techniques; alternatively, I can investigate representations w.r.t. a fixed suite of transformation techniques. This ability to vary the elements of this tuple not only constitutes the crux of the scope for creativity in this work, but also suggests the extensibility of the modeling methodology presented here.

Chapter 5

Supplementary Interactions, Generated Interfaces, and Software Framework

> “The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.”
>
> M. Weiser, in [Wei91]

Thus far in this dissertation, I have presented a formal model for representing and reasoning about personalized interaction. To target this model for real-time use in websites, I illustrate how the formalisms developed in the previous chapter must be augmented for supporting supplementary interactions, and how to automatically generate interaction interfaces. I conclude by presenting a software framework which brings all the relevant technologies together.

5.1 Supporting Supplementary Interactions

The success of a personalization system relies on those finer touches which deliver a compelling experience to the end-user. Studies of out-of-turn interaction have revealed that users desire supplemental interactions to enhance the personalized experience while engaged in a progressive out-of-turn dialog. In this section, I showcase a suite of such interactions and relate them to program transformation techniques. Interleaving out-of-turn interaction with these various interactions provides a comprehensive personalized experience.

5.1.1 *Meta*-enquery: What May I Say?

To employ a dialog-based system effectively, users need a mechanism to stay abreast of its underlying vocabulary at each point in the interaction. Keeping users attuned to the communicable information is an issue all information systems must address. Yankelovich echoes this issue as ‘how do users know what to say?’ [Yan96]. While each hyperlink label is always available, a new or causal user of a site may be unfamiliar with or unaware of subsequent solicitations for input and, hence, the terms of information seeking the designer has modeled which are available to supply. This is a classic problem in IR research, has been identified and described by many [BC99, CT87, Mar97, Sac00, Wil84], and is endemic to all IR systems. Users typically have a 'limited knowledge of a given database' [Wil84] and thus experience 'difficulty expressing their information need' [BC99].

The *Naive What May I Say?* interaction permits the user to determine what partial information remains unspecified in the user-system dialog. It can be trivially supported by TE. Supporting this interaction programmatically entails extracting all unique structural variables, which correspond to the partial information available to commit the system, from the representation. Analogously, I can achieve this interaction programmatically with $[\![\texttt{te}]\!]$.

In a voluminous space such a set of terms may be large and overwhelm the user. Thus, I could provide a similar interaction, called *What May I Say?*, which entails clustering terms by level (facet), using TLE, to help orient users. However, this approach is applicable only to the levelwise instances. When used in this manner, this interaction also updates the user on how their previous interactions have affected the remaining choices, and thus provides context. Supporting this interaction programmatically entails extracting all unique structural variables at a desired level of nesting from the representation. Thus, I can achieve this interaction programmatically with $[\![\texttt{tle}]\!]$.

5.1.2 Restructure Classification

Notice that while the out-of-turn interaction paradigm subsumes the browsing paradigm, interaction sequences *derivable* by out-of-turn interaction are not *describable* by browsing. The Restructure Classification interaction, which is only applicable to levelwise sites, enables the creation of a personalized browsing hierarchy, which can be further navigated via browsing or out-of-turn interaction. For example, a website organized along a author–journal–title motif could be restructured into a journal–author–title organization, supporting interactive aggregation scenarios.

Approaches to this interaction are to permit a user to restructure an entire enumerative classification *a priori* or incrementally. The semistructured data community [ABS00, FLM98] has advocated restructuring websites via declarative queries [FFLS97, FFK$^+$98]. Such an approach restructures an entire organization in one-stroke. A similar approach is taken with User-Defined Hierarchies [WB99] discussed in Chapter 3. An alternate approach is to permit the user to define the ensuing interaction incrementally while browsing. This interaction style has been echoed metaphorically as 'magically [laying] down track to suggest useful directions to go based on where [one has] been so far and what [one is] trying to do' [Hea00]. These ideas have been studied in the Flamenco Search System Project [HEE$^+$02]. Flamenco explores faceted classification in various catalogs and websites. The adaptive hypermedia community [BC99, Bru01] is a significant proponent of such an incremental approach to classification specification. In adaptive hypermedia, links are dynamic [Pok01] and lead to different destinations for different users.

Counterintuitively, my nested-conditional representation of interaction need not be restructured at all to accommodate this interaction! A method to extract structural variables corresponding to a particular level of conditional nesting is sufficient. These variables can be used to create hyperlinks on a generated webpage, each initiating an interaction which appears to be browsing, but is out-of-turn interaction w.r.t. the explicit representation in reality. This, yet again, speaks to the importance of investing in representation, and the flexibility in my representations.

Thus, support for this interaction requires level-order edge-label extraction, TLE. Supporting this programmatically entails the same code extraction, $[\![\texttt{tle}]\!]$, used for input expansion in mutually-exclusive classes of hierarchical hypermedia. Notice that the edge-label extraction for this interaction is w.r.t. one particular level, and is thus a specialization of the edge-label extraction required for (Naive) What May I Say?, which is w.r.t. all levels. In other words, the terms returned from TLE are a subset of those returned from TE.

5.1.3 Collect Results

This is a dialog termination interaction. It allows the user to request that the classification be flattened and re-presented as a flat list of hyperlinks to relevant content pages. For instance, while engaged in a dialog a user may wish to curb further interaction at a particular point and retrieve a flat list of the reachable webpages from that point. At such a point, the user may not care to pursue further distinctions or dependencies. Effectively, the user has prematurely declared that the dialog is over. This interaction involves the forward-propagation used to support out-of-turn interaction and is defined by CR which was introduced in the previous chapter. I support CR with the $[\![\texttt{cr}]\!] = [\![\texttt{forward}]\!][\texttt{P}_\texttt{D}, \texttt{page}]$ program transformation technique, a specialization of $[\![\texttt{zoom}]\!]$ which only extracts terminal variables from the representation. Notice that Collect Results suggests the need to invert the factoring of terminal information that I built into the programmatic representation with nesting. Lastly, observe that the function which supports Collect Results also can be used to generate *information previews*—hyperlink annotations, typically parenthesized and containing a frequency count of the documents reachable from the hyperlink it decorates—which have been shown important in situations where the user is confronted with a decision regarding where to go next [PSDB99].

5.1.4 Inverse Personalization

Inverse personalization is so named, not because its goals run contrary to personalization, but because in a given modeling it conducts a mapping from terminal (e.g., leaf pages) to structural information (e.g., hyperlink labels). This interaction helps support situations where the user knows what information she wants, but is interested in determining how to retrieve that information. It permits users to conduct 'what if' analyses. For instance, using Inverse Personalization with an online apartment recommender, a user can issue a request such as 'Under what conditions will Rockville apartments be the only choice?' Such a user is interested in the terms along the interaction sequences leading to that terminal information (e.g., 'if you want a swimming pool, covered parking, and free Internet access'). Such personalized interactions are supported in my model by back-propagating from the leaf webpages returned by SL, described in the previous chapter, and optionally extracting each term in each path from the root as a whole, using TE, or per interaction sequence, using $TE \circ SP$.

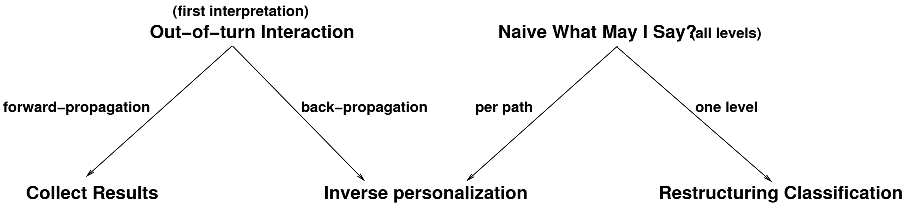


Figure 5.1: A taxonomy of supplementary interactions. Directed arrows represent specialization relations.

	read-only	closed	independent	program transformation technique
NWhat?	√	√	×	∪ {⟦`te`⟧ [P_D, `x`]}, for every structural variable `x`
What?	√	√	×	∪ {⟦`tle`⟧ [P_D, `x`]}, for every level `x`
RC	√	√	√	⟦`tle`⟧ [P_D, `x`], for one level `x`
CR	×	×	√	⟦`forward`⟧ [P_D, `page`]
P^{-1}	√	√	√	⟦`te`⟧ [⟦`sp`⟧ [⟦`backward`⟧ [⟦`SL`⟧ [P_D, `input`]]]]

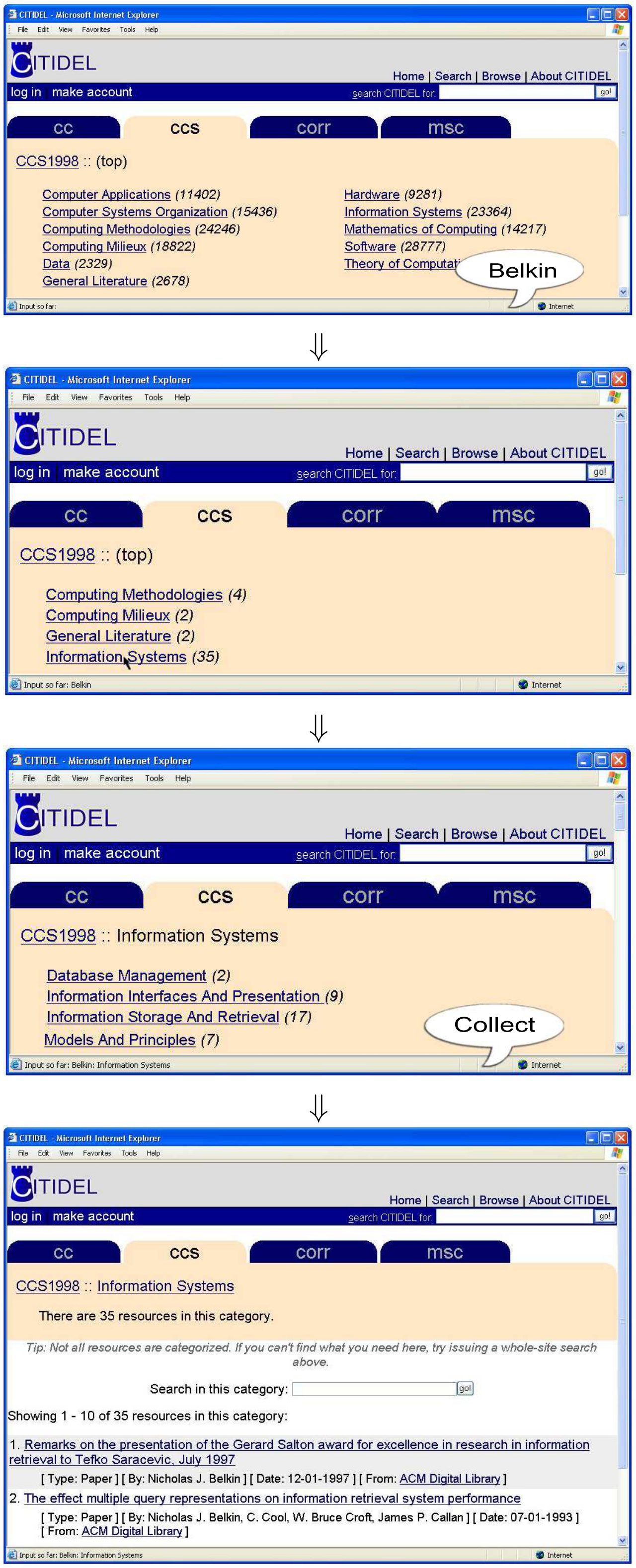
Table 5.1: Program transformation techniques for and observations on the supplemental interactions. (**key**: NWhat? = Naive What May I Say?; What? = What May I Say?; RC = Restructuring Classification; CR = Collect Results; P^{-1} = Inverse Personalization).

Observe that Inverse Personalization is similar to generalized interpretation 1 of out-of-turn interaction, as evidenced in its use of SL. Notice however, that, akin to Collect Results, this interaction is a specialized form of out-of-turn interaction. Honing in on leaves (i.e., forward-propagation) is unnecessary; back-propagation is sufficient, and therefore defines this interaction. In my conditional representation, terminal information is indexed by a terminal variable (e.g., `page`) which is not user-modifiable (as is structural information; e.g., `Democrat`). Therefore, I need a transformation capable of exploiting terminal variables, such a backward slicing. Collecting the terms along the resulting paths requires an extraction, ⟦`te`⟧ similar to that required for What May I Say?.

Fig. 5.1 illustrates the relationships between the interactions presented in this chapter.

5.1.5 Example

Fig. 5.2 illustrates a personalized interaction with the CITIDEL (Computing and Information Technology Interactive Digital Educational Library; citidel.org) digital library using SALTII [PMR+04]. Here the user is interested in papers by ‘Belkin’ but is unsure what categories Belkin has published in. Thus, the user is initially unable to respond to the solicitation of literature category and instead says ‘Belkin’ out-of-turn (window i). This causes many leaf pages to be removed (notice reduced frequency purviews annotating each hyperlink label) and some categories, such as ‘Hardware,’ to be completely pruned out, since Belkin’s papers are not indexed under such categories (window ii). The user then responds to the initiative by following the ‘Information Systems’ hyperlink (window ii). Next, the user decides to terminate the dialog by using Collect Results (window iii) to request a flat list of Belkin’s papers (window iv).


CITIDEL - Microsoft Internet Explorer
File Edit View Favorites Tools Help
CITIDEL
Home | Search | Browse | About CITIDEL
log in make account
search CITIDEL for:
go!
cc
ccs
corr
msc
Belkin
⇓
Collect
Internet

CCS1998 :: (top)

Computer Applications (11402)
Computer Systems Organization (15436)
Computing Methodologies (24246)
Computing Milieux (18822)
Data (2329)
General Literature (2678)
Hardware (9281)
Information Systems (23364)
Mathematics of Computing (14217)
Software (28777)
Theory of Computati

Input so far:

CCS1998 :: (top)

Computing Methodologies (4)
Computing Milieux (2)
General Literature (2)
Information Systems (35)

Input so far: Belkin

CCS1998 :: Information Systems

Database Management (2)
Information Interfaces And Presentation (9)
Information Storage And Retrieval (17)
Models And Principles (7)

Input so far: Belkin: Information Systems

CCS1998 :: Information Systems

There are 35 resources in this category.

Tip: Not all resources are categorized. If you can't find what you need here, try issuing a whole-site search above.

Search in this category:

Showing 1 - 10 of 35 resources in this category:

1. Remarks on the presentation of the Gerard Salton award for excellence in research in information retrieval to Tefko Saracevic, July 1997
[Type: Paper] [By: Nicholas J. Belkin] [Date: 12-01-1997] [From: ACM Digital Library]

2. The effect multiple query representations on information retrieval system performance
[Type: Paper] [By: Nicholas J. Belkin, C. Cool, W. Bruce Croft, James P. Callan] [Date: 07-01-1993]
[From: ACM Digital Library]

Input so far: Belkin: Information Systems

Figure 5.2: A personalized dialog with CITIDEL involving interactions supplemental to out-of-turn interaction.

5.1.6 Program Transformations for Supplemental Interactions

Table 5.1 contains program transformation techniques for the interactions showcased in this chapter. Notice that the What May I Say? and Restructuring Classification interactions require slight variations of the same transformation technique. Since Collect Results and Inverse Personalization are each specializations of out-of-turn interaction, they relate to forms of forward and backward slicing, respectively (and code extraction), and thus, when put together, come close to realizing out-of-turn interaction.

Table 5.1 also summarizes my observations on these interactions. The top axis of the matrix lists attributes of the interactions listed on the left. A read-only interaction is one which merely manipulates the representation rather than modifying it. An interaction is closed if it accepts a representation and returns a representation, and thus does not prevent further interaction. Independent interactions are those which not only complement out-of-turn interaction, but are also defined, applicable, and useful in its absence.

5.2 Automatically Generating Personalized Interaction Interfaces

The reader will have noticed that while some of the supplemental interactions are always applicable, others are only applicable in certain classes of hierarchical hypermedia (e.g., levelwise classes). Beyond such underlying class constraints, website designers may desire to control which supplemental interactions they afford users. Marchionini states that ‘it is naive to believe that any single interface can serve the needs of all users for all tasks’ [Mar97]. For these reasons, a monolithic interaction interface, with provisions for all interactions, is unrealistic. Therefore, I developed a system to automatically generate customized interaction interfaces tailored to specific websites, especially those which support personalized interaction.

5.2.1 Out-of-turn Toolbar Markup Language and Translator

My approach to automatic interface generation produces an Extempore-like toolbar or a SALTII-like voice interface [PRF04]. I built the generator using Java; a graphical interface (not shown here) configures an interaction interface by allowing designers to choose the interactions they desire to support. In this manner, it *personalizes* interfaces for personalized interaction. For website designers who are more computer-literate, I defined a small markup language, called OTML (Out-of-turn Toolbar Markup Language) using XSchema (see Appendix A) for describing an interaction toolbar specification capturing which interactions the resulting interface should support as well as its look and feel. My generator employs the transformation capabilities of XSLT to translate an OTML specification[1] into a XUL toolbar. Use of OTML supports a finer level of customization than the generator, especially for the look and feel of the interface. For instance, designers can customize tooltips (popup text describing a control’s utility) for each widget in the generated toolbar.

---

[1] The UI version of the generator simply generates an OTML specification from the parameters provided and makes a call to an XSLT processor in a manner transparent to the user.

The generator currently supports the following personalized interactions:

- Out-of-turn interaction (both interpretations presented in this thesis)
- Generalized out-of-turn interaction (both interpretations presented in this thesis)
- *Meta*-enquery: (Naive) What May I Say?
- Restructure Classification
- Collect Results
- Inverse Personalization

To the best of my knowledge, few tools exist to aid personalization system designers. I expect such semi-automated interface construction to become more attractive with the advent of tools for librarians to build their own digital libraries [Zhu02]. Using semi-automated toolbar construction in conjunction with such tools will result in digital libraries which take initial steps toward integrating personalization and specifically, out-of-turn interaction, into user experiences. Lastly, I anticipate approaches to generating interfaces for websites to become more popular with an increased interest in end-user programming [DE95].

5.3 Putting It All Together: Building a Robust Transformation Engine

I built a purely functional (side-effect free), robust transformation engine, based on the model developed in this dissertation, and deployed it as a web service. It handles in-turn and out-of-turn inputs, passed from any interaction interface, in a uniform manner. The engine implements a form of forward, followed by backward slicing to support out-of-turn interaction with a variety of websites. I built the engine with XSLT whose support for pattern-oriented programming is particularly advantageous here since both forward and backward slicing can be captured with ancestor or descendant axis types in location paths. While XML documents obey a tree-structured model, I used `ids` and `refids` to factor crosslinks, and hence enhanced the modeling capabilities of the representations introduced in Chapter 2 by effectively modeling DAGs as well. This engine also supports several of the supplemental interactions presented in this chapter.

The interaction interfaces I discuss in this dissertation, this transformation engine, and an interaction manager [Wil04] constitute a customizable software framework for creating multimodal web personalization systems which support mixed-initiative interaction (see Fig. 5.3)[NWPR04]. The interaction manager coordinates processes between the interfaces and engine. More importantly, since the interaction manager supports session control and caching, each instantiation of this framework enhances my initial prototypical implementation, described in Chapter 2, from supporting (1 user × 1 interaction) experiences to (n user × m interaction) experiences, in a manner that takes advantage of the statefullness of my approach. Our research group has instantiated this framework for several case studies,

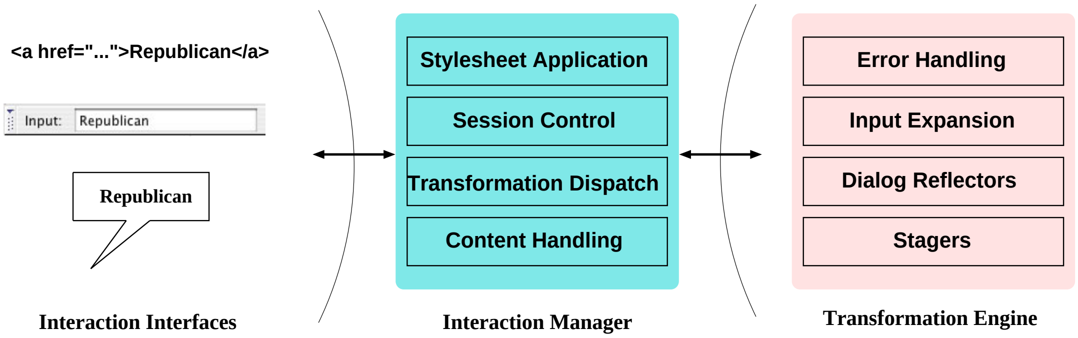


Figure 5.3: Multimodal, mixed-initiative web personalization framework architecture. Notice the central role played by an interaction manager in mediating communication between the interaction interfaces and the transformation engine.

including those presented in this dissertation, which afford interactions initiated by these interfaces, coordinated by this interaction manager, and staged by this transformation engine. Websites for which we instantiated our framework include:

- GAMS (Guide to Available Mathematical Software) at gams.nist.gov.
- Project Vote Smart at vote-smart.org.
- Pigments through the Ages at webexhibits.org/pigments/.
- CITIDEL (Computing and Information Technology Interactive Digital Educational Library) at citidel.org.
- Open Directory Project at dmoz.org.
- Online Virginia Tech Timetables of Classes accessible through vt.edu.

These sites are a rich assortment of the classes of hierarchical hypermedia introduced in the previous chapter. Some are available to demo at our project website, http://pipe.cs.vt.edu.

Chapter 6

Exploring Out-of-turn Interactions with Websites

> "A goal of HCI research is to develop input-output devices and mechanisms that map more naturally onto human channels."
>
> Gary Marchionini, in *Information Seeking in Electronic Environments* [Mar97]

In this chapter I present the first study to explore the use of out-of-turn interaction in websites. While out-of-turn interaction can be studied in many settings (as seen in Chapter 4), this chapter only discusses its use in conjunction with browsing websites modeled by levelwise, mutually-exclusive trees. The targeted website for this study was the Congressional portion of the Project Vote Smart (PVS) website discussed in Chapters 2–4. The main component of the study entailed asking participants to perform eight specific information-finding tasks using the Extempore and SALTII interfaces. Recall that Extempore works with Mozilla and SALTII with Internet Explorer[1]. Rationale was gathered through think-aloud and retrospective protocols.

The goal of the experiment was to study usage patterns for out-of-turn interaction, not to evaluate the interfaces used to realize it, or to compare out-of-turn interaction with other interaction techniques.

6.1 Methods

6.1.1 Participants

We collected data from 24 participants in the analysis; all were students with an average age of 21, and a majority were undergraduates in computer science. Some of the participants were recruited from a HCI course, and were compensated with extra-credit from the instructor. Since a component of this experiment involved voice recognition software, we primarily recruited native speakers of English. Average participant computer and web familiarity and

---

[1] At the time the study was conducted, a SALT plugin for Mozilla did not exist (and likewise with XUL for IE). One has since been implemented. Due to these technological constraints, I do not implement both interaction interfaces in the same browser.

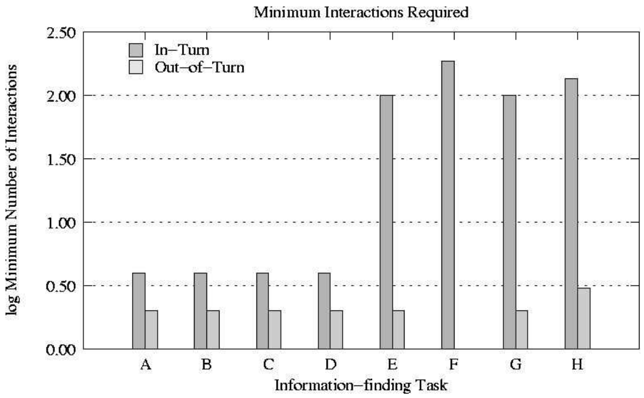


Figure 6.1: Minimum number of interactions ($\log_{10}$ scale) required to successfully satisfy each information-finding task using in-turn (dark) and out-of-turn (light) interaction. Note that Task F can be completed with just one out-of-turn interaction, so its entry in the graph shows zero.

use was 4.75 or greater on a 5-point Likert scale. Average participant familiarity with voice recognition software was 1.46, and mean familiarity with the structure of the US Congress was 2.83; no user had visited the PVS website prior to the experiment.

6.1.2 Tasks

The eight tasks were carefully formulated to generate a diverse set of interaction choices:

A. Find the webpage of the Junior Senator from New York.

B. Find the webpage of the Democratic Representative from District 17 of Florida.

C. Find the webpage of the Republican Junior Senator from Oregon.

D. Find the webpage of the Democratic member of the House in Rhode Island serving district 2.

E. Find the states which have at least one Democratic Senator.

F. Find the states which have twenty or more congressional districts.

G. Find the states which have at least one Republican member of the House.

H. Find the political party of the Senior Senator representing the only state which has congresspeople from the Independent party.

I refer to tasks A, B, C, and D as *non-oriented* tasks, in that they can be performed as easily by employing solely in-turn interaction (i.e., in this case, hyperlinks), solely out-of-turn interaction, or using a mixture of both. Out-of-turn interaction does not appear to be worthwhile with respect to these tasks because the effort required to perform them with out-of-turn interaction is commensurate with that of in-turn interaction. Tasks E, F, G, and H are *out-of-turn-oriented*, because they are difficult to perform with only in-turn interaction.

Formally, I say an information-seeking task is out-of-turn-oriented if the minimum number of browsing interactions required to successfully complete it exceeds the maximum depth of the targeted website; otherwise it is non-oriented.

The maximum depth of the PVS site is four and Figure 6.1 illustrates the minimum number of interactions required per task. In calculating this minimum number, I assumed that the user can supply at most one aspect at each step (in-turn or out-of-turn), and discounted back button clicks (happens when employing only in-turn interaction for an out-of-turn-oriented task). Notice also that some tasks, namely the non-oriented ones, cannot be performed by purely a sequence of out-of-turn interactions; a terminal in-turn input is often necessary and these are discounted as well. For instance, try solving task A using purely out-of-turn inputs.

6.1.3 Design

The study was designed as a within-subject experiment. Task was the independent variable and the interaction observed (in-turn vs. out-of-turn) was the dependent variable. Participants were given both the Extempore toolbar and the SALTII voice interface; and performed four tasks with each (two non-oriented and two out-of-turn-oriented). I designed the experiment with the provision for interfaces in two different modalities, to more naturally assess the use of out-of-turn interaction independent of a particular interface for it. Each participant performed the eight tasks in an order pre-determined by a latin square to control for unmeasured factors. In addition, the specific interface to be used (toolbar or voice) for a (task, participant) pair was determined *a priori* by complete counterbalancing within each task category. Thus, for each task, half of the twenty-four participants were given the toolbar interface and half the voice interface. The participants were free to utilize any strategy to complete the information-finding tasks, given the available interfaces; they were given unlimited time to complete each task.

6.1.4 Modeling Choices

A vocabulary for the PVS site was created by collecting all link labels, synonyms (e.g., 'Representatives' for 'House'), and alternate forms of common utterances (e.g., 'Senate', 'Senator', 'Senators'). Both Extempore and SALTII supported this vocabulary, with the toolbar supporting abbreviations (e.g., CA for California), in addition. To keep users abreast of partial information supplied thus far (either by browsing or via out-of-turn interaction), I continually updated an 'Input so far:' label in the browser status bar (see Fig. 3.19, bottom). I also included a provision for the user to inquire about what partial information is left unspecified at any step. Access to this feature is provided through a 'What May I Say?' button (labeled with a '?' in Fig. 2.2) or utterance. In the course of an interaction, when the supplied partial input uniquely determines a politician, I performed input expansion by positive-path FDs (ref. Chapter 4).

6.1.5 Equipment, Training, and Procedures

Equipment

Participants performed the tasks on an Extempore/SALTII-enabled browser in a Pentium III workstation, connected to a 17” monitor set at 2560×1024 resolution in 34-bit true color, running Windows 2000. While each participant performed the information-finding tasks, we captured their interaction with the system using the Camtasia screen and audio capture software. We used the resulting capture to aid participant recollection during the retrospective verbal protocol as well as in subsequent analysis (e.g., think-aloud). The Audacity audio recording application was used during the retrospective portion of the experiment to capture participant explanations. Data from the pre-questionnaire (demographics, computer familiarity) and post-questionnaires (rationale) was recorded on paper. Finally, at the end of the entire experiment we transcribed and collated the data gathered from all sources to construct a complete record of each participant session, including interaction sequences followed per task. Each participant session lasted approximately 90 minutes.

Training

Prior to revealing the information-seeking tasks, we gave users specific training on (i) the PVS website, including levels of classification, and interacting with it via hyperlinks; (ii) interacting with PVS using out-of-turn interaction (both toolbar and voice interfaces); and (iii) interleaving hyperlink clicks with commissions via Extempore/SALTII. Users were provided a card summarizing the vocabulary that the out-of-turn interfaces can understand, as well as explanations of political terms and their functional dependencies. This card was available for their use during the entire session, not just training. We did not use terms such as ‘in-turn’ or ‘out-of-turn’ during training or elsewhere in the study. This is to prevent biasing of participants toward any intended benefits of out-of-turn interaction, and also to help them conceptualize its functionality on their own. In other words, we simply trained users on how to employ the available interfaces for information seeking. After some self-directed exploration, users were given a short test consisting of four practice tasks (two with toolbar and two with voice).

Procedures

After the users completed the training tasks, we administered the actual test involving tasks A–H above, and employed both concurrent (think-aloud) and retrospective protocols to elucidate rationale. A structured interview, including a post-questionnaire, was conducted to gather additional feedback. The entire experiment generated (24×8 =) 192 (participant, task) interaction sequences.

These sequences were then analyzed for frequencies of usage of in-turn vs. out-of-turn interaction. For purposes of this study, I defined an in-turn interaction as a hyperlink click or the communication of in-turn partial information to the website via Extempore/SALTII. Notice that just saying ‘Connecticut’ will not qualify as an out-of-turn interaction, if the same choice was currently available as a hyperlink. Similarly, we defined an out-of-turn interaction to be the submission of one aspect of unsolicited partial information to the site.

Supplying more than one aspect of partial information to the site out-of-turn (e.g., saying 'Democratic Senators') corresponds to multiple out-of-turn interactions.

Notice that a user may supply in-turn and out-of-turn information to the website simultaneously via Extempore/SALTII. For instance, at the outset the user might say 'House, Florida, District 17, Democrat,' all in one utterance. Observe that a permutation of this utterance exists—'Florida, House, Democrat, District 17'—that, if conducted incrementally, could imply a purely in-turn interaction. Such an interaction is thus viewed as having four in-turn inputs. On the other hand, consider a user who says 'New York, Democrat' at the outset. There is no permutation with respect to the PVS site that permits viewing this utterance as comprising of purely in-turn input, and hence, it is classified as one in-turn input ('New York'), followed by an out-of-turn input ('Democrat'). This policy of counting does not favor (and actually deprecates) out-of-turn interaction.

Some users, after completing a given task via out-of-turn interaction, verified part of their results via in-turn interactions. This was confirmed through their retrospective feedback, and such in-turn interactions were discounted in the analysis.

6.2 Results

Of the 192 recorded interaction sequences, 177 of them involved the successful completion of the task by the participant. I analyze these 177 sequences first, followed by the remaining 15 sequences (which were all generated in response to out-of-turn-oriented tasks).

6.2.1 General Usage Patterns

Results indicate a high frequency of usage for out-of-turn interaction. 94.4% of the 177 sequences contained at least one out-of-turn interaction. In addition, every participant used out-of-turn interaction for at least 70% of the tasks, with 16 people using it in all tasks. Conversely, every task was performed with out-of-turn interaction by at least 80% of the participants, with 4 tasks enjoying out-of-turn interaction by all participants. These results are encouraging because Extempore/SALTII usage is optional and not prompted by any indicator on a webpage. Participants successfully completed the given tasks irrespective of the presented interface (voice or toolbar).

6.2.2 Classifying Interaction Sequences

The 177 interaction sequences were classified into five categories denoted by: (i) I, (ii) O, (iii) IO, (iv) OI, and (v) M. The I and O categories denote sequences comprised of purely in-turn, or out-of-turn inputs, respectively. In IO sequences all in-turn inputs precede out-of-turn inputs (analogously, for OI). M sequences ('mixed') are those which do not fall in the above categories. For instance, the interaction shown in Fig. 2.10 would be classified under I, and that in Fig. 2.11 is in OI. I posit that this classification provides insight into users' information-seeking strategies, and can be related to the nature of the information-finding task.

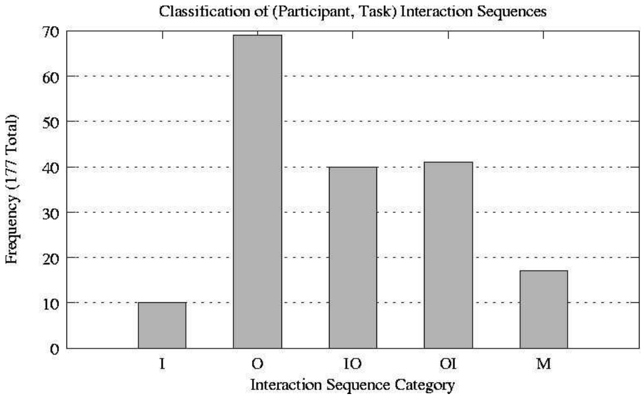


Figure 6.2: Classification of 177 (participant, task) interaction sequences.

	I	**{O,IO,OI,M}**	**total**
non-oriented	10	86	96
out-of-turn-oriented	0	81	81
total	10	167	177

Table 6.1: Breakdown of 177 interaction sequences in various categories. The total number of interaction sequences for out-of-turn-oriented tasks is 15 less than that for non-oriented tasks; these were the sequences where the participant did not complete the task successfully.

Figure 6.2 shows the distribution of the 177 sequences into the five classes, and Table 6.1 depicts a breakdown by both task orientation and classes. Notice that O, OI, IO, and mixed classes have been grouped in Table 6.1 to distinguish them from pure browsing interactions (I).

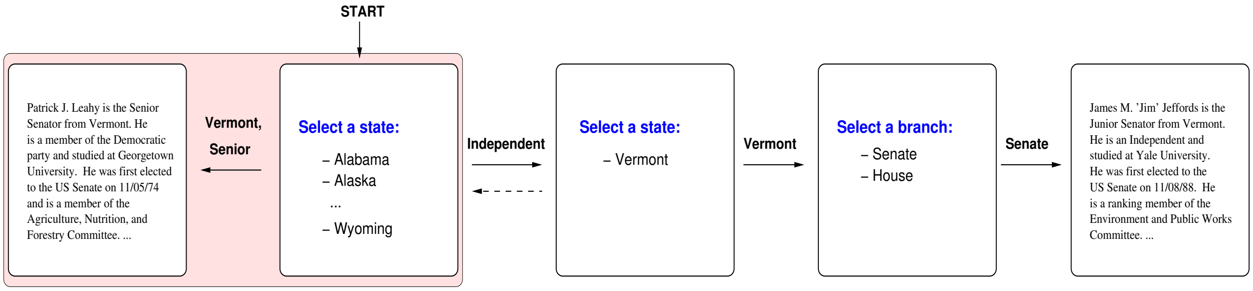
As Figure 6.2 shows, 10 of the 177 sequences fall in the I class, i.e., these are *browsing* sequences. As Table 6.1 (lower left) shows, all of the 10 browsing sequences were generated in response to non-oriented tasks, revealing that a 100% (81/81) of the sequences for out-of-turn-oriented tasks involved out-of-turn interaction. Therefore,

- users never attempted to achieve an out-of-turn oriented task via browsing; or in other words,
- users always employed out-of-turn interaction when presented with an out-of-turn-oriented task.

This is notable because it confirms that users are adept at discerning when out-of-turn interaction is necessary.

6.2.3 Detailed Analysis of Interaction Classes

Let us now study the interactions in classes O, OI, IO, and M. The 69 pure out-of-turn sequences (O) were observed only in out-of-turn-oriented tasks E, F, and G, and was used by all the 24 participants. This clustering of the O sequences around three tasks shows that, whenever participants completed these tasks, they did so in the shortest manner possible.



Patrick J. Leahy is the Senior Senator from Vermont. He is a member of the Democratic party and studied at Georgetown University. He was first elected to the US Senate on 11/05/74 and is a member of the Agriculture, Nutrition, and Forestry Committee. ...

James M. 'Jim' Jeffords is the Junior Senator from Vermont. He is an Independent and studied at Yale University. He was first elected to the US Senate on 11/08/88. He is a ranking member of the Environment and Public Works Committee. ...

Figure 6.3: Task H: the user is expected to first find 'Vermont' in one Interaction (third window from left) and use it as input in another interaction (shaded area) to find the party of the Senior Senator from that state. The two windows on the right depict unnecessary and irrelevant interactions for this task.

Refer again to Figure 6.1 for the sharp contrast in the length of the minimum out-of-turn sequence from the minimum in-turn sequence, for these tasks.

Classes IO, OI, and M contain the sequences exhibiting rich interaction strategies. Classes IO and OI were observed in near-equal numbers, and primarily in the non-oriented tasks (A, B, C, and D) with the exception of OI, which was also seen in task H. No particular clustering was observed with respect to participants. The 17 class M interactions exhibited only two types of patterns – 14 with an OIO form and 3 with an IOI form. Furthermore, like OI, these 17 mixed interactions also involved only the non-oriented tasks (A, B, C, D) and task H. It is interesting that we observed OIO and IOI sequences, even in a site with only four levels. Once again, no specific clustering was observed around participants.

To see if these classes correspond to specific information-seeking strategies, I plotted curves depicting the progressive narrowing down to a desired congressional official, as a function of interaction steps. All curves begin at the (0, 540) point because the PVS site indexes all 540 congressional officials. With each interaction, this number is gradually reduced until the user arrives at the desired set of officials. However, I was unable to observe major correlations between curve slopes and strategies; this is because in the PVS site, the slope is primarily dependent on the nature of the task, not the strategy. For instance if a task involved a state like 'Rhode Island,' even an in-turn input of this state information will cause greater pruning than most out-of-turn inputs. To qualify interaction classes better, I must study out-of-turn interaction in more sites.

6.2.4 Cascading Information across Subtasks

Recall that 15 interaction sequences led to incorrect answers; interestingly 12 of these 15 were generated in response to Task H. Notice that Task H is challenging, because it involves two subtasks and cascading information found in one into the other. The user is expected to first find the only state having Independent congressional officials (Vermont), and then find the political party of the Senior Senator from that state (Democrat). In other words, this task requires procedural, not just declarative, knowledge (a distinction motivated in the Strategy Hubs project [BBJ^{+}03]).

Most people were adept at finding that Vermont was the desired state (e.g., by saying 'Independent' at the outset), but did not realize that the task cannot be completed by *continuing* that interaction. As Figure 6.3 shows, clicking on the only available state link ('Vermont') now presents a choice of House vs. Senate. Clicking on Senate takes the user to the webpage of Jim Jeffords, who is the Junior Senator from Vermont, not the Senior Senator!

Some users immediately realized the problem, as identified in their retrospective interviews, e.g.:

> "This question was tricky. Cause it was, I was like wait, if he's Independent then his party is Independent . . . at first [I thought] it was the Senior Senator who was Independent . . . and I got this guy's webpage, and then I saw that he was the Junior . . . So then I eventually went back to Vermont and got the [Senior] guy."

Only 12 (50%) of the participants successfully completed this task. This result demonstrates that cascading information across subtasks is challenging. It was clear that all users wanted to continue the interaction, but some failed to realize that out-of-turn interaction as presented here is merely a pruning interaction, and not constructive. Investigating the incorporation of constructive interactions such as rollup/expansion is thus a worthwhile direction of future research.

6.2.5 Rationale and Qualitative Observations

Studying users' rationale revealed their reasons for interacting out-of-turn:

> "I can jump through all the levels . . . ."
>
> "Initially I thought I would prefer the hyperlinks . . . after reading the questions, it became apparent that the toolbar and voice interface would simplify the task."
>
> ". . . when you wanted to know all the states for the Republicans, then you would have to click on every single link. It would just get annoying after a while. You'd just give up I think. There'd be no way."
>
> "I guess I would have had to . . . wow, check every state."

demonstrated understanding of how Extempore/SALTII works (e.g., input expansion):

> "Its the easiest way cause there is only one Representative from District 17 in Florida and it takes you straight to the page."
>
> "If you click on the state then you get choices of House and whatever, but if you type in district 2 and it just goes right there."

presented advantages and judgments:

> ". . . allowed multiple pieces of information to be input at one time."
>
> "As much surfing as I do, it sort of makes me wish I had those options sometimes ya know instead of going to search engines and fooling around . . . having to come up with different search criteria . . . ."

and also brought out frustrations:

> "The voice interface feels a little awkward since I am not used to talking to myself . . . ."
>
> "I don't always trust the results, [so I went back] confirming that they are all republican."

Many users learned that out-of-turn interaction is best suited when they have a specific goal in mind, and not meant for exploratory information-seeking (as is browsing). For instance,

> "if I wanted to go the whole way down to a specific person, I would use [Extempore/SALTII], but if I was just looking around, I would use the links."
>
> "[Extempore/SALTII] is good when you know the site and know you have to go several layers deep. Links [are good] when you don't know the layout or don't know exactly what you want."

6.3 Discussion

There are significant lessons brought out by this study. First, this work validates my view of web interaction as a flexible dialog and shows that users actively interleaved out-of-turn interaction with browsing. Importantly, users were proficient at determining when out-of-turn interaction is called for. Studying the rationale and usage patterns has generated a body of knowledge that can be used, among other purposes, for introducing out-of-turn interaction in new settings and to new participants. Furthermore, we have seen that it is easy to target out-of-turn interaction in domains where tasks involve combinations of focused and exploratory behavior. Recall also that dialogs with purely declarative specifications are readily supported; others such as Task H will require further study. Therefore, this work not only improves our understanding of out-of-turn interaction, but also suggests further opportunities to enrich browsing experiences for users.

6.3.1 How do Users Know What to Say?

In order for out-of-turn interaction to be effective, the user must have a basic understanding of what can be said. This is a well-acknowledged issue by the speech interfaces community; as Yankelovich [Yan96] points out, 'the functionality of [such] applications is hidden, and the boundaries of what can and cannot be [said] are invisible.' The semantics of out-of-turn interaction are more specific than free-form conversational input, because it merely allows a site's existing navigation structure to be realized in a different order. Out-of-turn interaction is hence most effective (i) when users have a focused task, possess a basic understanding of the application domain, and know what aspects are addressable, or (ii) when users desire to expose dependencies implicit in a modeling. When users do not know what to say, my facility to enquire about legal utterances may induce information overload in large sites. While I have not encountered this problem in the PVS study, I suspect that applying out-of-turn interaction in large web directories (e.g., ODP) will involve new research directions.

6.3.2 From Interaction Techniques to Interaction Interfaces

This dissertation has been primarily concerned with studying the *interaction techniques* for personalization (how to model them, how to support them, and how do users utilize them), and not evaluating the *interfaces* that realize these techniques. However, in the course of the study presented here, I have learnt that effective interfaces are a crucial part of the user experience and must be evaluated in their own right. Such a study is outside the scope of this dissertation, but I make some preliminary observations – first on the nature of interfaces and then on how they might be compared.

The Nature of Interaction Interfaces

There are several interfaces for personalized interaction available to today's web users [BBJ$^+$03, DCC01, HEE$^+$02, OC03, BBJ$^+$03, PSC$^+$02]. To better study interfaces for personalized interaction, I showcase different projects in a three-dimensional space (see Figure 6.4) involving: (i) the nature of information exploited, (ii) the level of context supported, and (iii) the interaction technique.

The first axis distinguishes between the specification of partial vs. complete information. Supporting only the specification of complete information means that interaction is viewed as a one-shot activity (and is classified in tier one of my survey, ref. Chapter 3); supporting specification of partial information implies that information seeking is to be conducted over multiple steps. Since the complete information approach is more restrictive than the partial information approach, it is situated toward the origin. The second axis makes a distinction of whether input or results (or both) are contextually qualified in some manner. My contribution to this space is the third dimension of whether interaction occurs by in-turn or out-of-turn means.

Search engines (e.g., Google) are characterized by specification of complete information (in this case, the query), because the interaction is terminated by returning a flat list of results. Such a low-context, complete information approach is denoted by the origin in Fig. 6.4. Browsing, on the other hand, involves the incremental specification of partial information (right of origin in Fig. 6.4).

When I take context into account, two further clusters of projects emerge in the in-turn plane spanning the (information × context) axes. When only complete information is supported, results presentation provide the major opportunity for exhibiting context (front left corner of Figure 6.4). This is seen in site-specific search tools (e.g., at Amazon.com), in the contextual search of Dumais, Cutrell, and Chen [DCC01], and the personalized search strategies of Pitkow *et al.* [PSC$^+$02]. The more dense cluster (front right of Figure 6.4) forms in the partial information region. These are projects that support contextual information access by providing either greater input flexibility or adaptable display of results over the course of an interaction, or both. Faceted (flat or hierarchical) organizations [HEE$^+$02, RCCR02b], Dynamic Taxonomies [Sac00], Strategy Hubs [BBJ$^+$03], adaptive hypermedia [Bru01], and ScentTrails [OC03] are examples. I discuss these further.

Sites and systems exposing faceted browsing (e.g., epicurious.com) support multiple classifications by providing enumerated in-turn choices. This often leads to cumbersome site designs and a mushrooming of possible choices at each step. The Dynamic Taxonomies

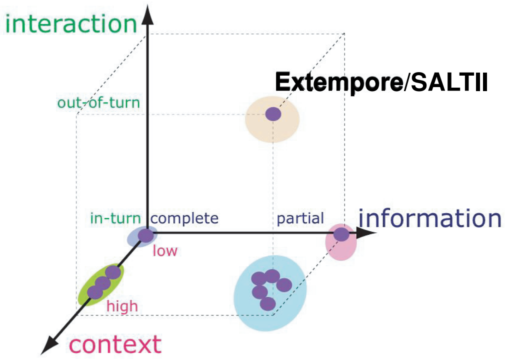


Figure 6.4: Three dimensional space showcasing related research. Each of the shaded clusters denotes a concerted group of projects discussed in the main text.

project provides in-turn operators for pruning information hierarchies, while Strategy Hubs enumerates templates for prolonged and detailed information-seeking tasks, again in-turn. The adaptive hypermedia projects employ user models (e.g., of past browsing behavior) to tailor the presentation of hyperlinks. ScentTrails argues that browsing may not be focused enough and that searching loses context, and aims to combine them in a single framework. However, its use of searching always precedes browsing and therefore limits the richness of supportable interactions. Out-of-turn interaction aims to provide precisely this combination of focused input and exploratory browsing in a single, flexible, framework.

My work can be viewed as complementary to these efforts in that it *lifts* the nature of interaction from in-turn to out-of-turn means (top of Figure 6.4). For instance, the example session shown in Figure 2.11 can be viewed as a lifted version of traditional browsing, yielding a (high context, out-of-turn) technique that exploits partial information. Such lifting, while attractive from a functional standpoint, presents challenges for interface comparison. In particular, it places the onus entirely on the experimenter to establish what a fair common ground for comparison might be, prior to undertaking a detailed evaluation of interfaces with users.

How can we evaluate lifted interfaces against others?

Establishing a common ground for evaluation can be done in many ways. At the modeling level, I can ensure that all compared interfaces assume the same vocabulary or granularity of addressable information. Second, the specification of the task should be independent of the vernacular of the interface. For instance, in the tasks presented earlier, the terms of information-seeking (and hence, the partial input) are salient, e.g., ‘E: Find the states which have at least one **Democratic Senator**.’ In order to address this criticism, I might develop a problem-solving or decision-making scenario that requires *the user* to cull the terms from the problem statement. Third, multimodality (e.g., using SALTII) introduces a new dimension of complexity and makes difficult the assignment of credit/blame to a specific interaction interface. One solution to this problem is to evaluate interfaces in a single modality (e.g., Extempore with Google, and SALTII with a conversation-based dialog engine). Fourth, users' familiarity with interfaces must be factored into the training. This is especially important when comparing Extempore with, say Google, that already enjoys a large userbase. Finally, speeds of interface response must be comparable if time for task completion is the chief evaluation metric. Studying information systems in such a multi-faceted manner, emphasizing their role in problem solving, has been suggested by Henninger and Belkin [HB96], and Marchionini [Mar97].

As a first step towards such a study, I present a simple problem-solving scenario involving a student developing a schedule of courses using the Virginia Tech Online Timetable of Courses (Fig. 4.10, left and center), subject to several constraints (see Appendix D). These constraints are realistic and, at the same time, do not draw attention to pertinent terms of information-seeking directly. In a comparative study, I might evaluate Extempore with (i) a search engine such as Google, (ii) faceted browsing interfaces as supported in Flamenco, and (iii) an interactive table-based interface such as employed in Apple's *iTunes*. Such a comparison is akin to the approach taken in [KB96]. Preliminary results [PPR+03] confirm that tasks requiring rich interaction and/or constraint satisfaction are not well-supported in a one-shot search engine and, while a faceted browsing interface supports the same functionality as Extempore, users found it cumbersome to interact with.

Chapter 7

Discussion

> "In the end, how well our approach performs is an empirical question. Whether users of the information superhighway prefer to build their own 'hot rods' . . . or take 'public transportation' that serves all uniformly, will ultimately be judged by history."
>
> D. Rus and D. Subramanian, in [RS97]

This research has made contributions to both users and designers of information systems. By providing users with out-of-turn interaction capability, my research has helped bring mixed-initiative interaction to the web. In addition, by studying personalization from the perspective of interaction, my research brings a user-centered approach to the subject [KNV00]. For designers, I have contributed a modeling methodology for personalized interaction. In addition, this work has resulted in support tools and software modules for rapid prototyping of personalization systems.

To the best of my knowledge, I am the first to approach personalized interaction from a programmatic-representational and -transformational perspective. Studying personalization from this viewpoint has provided several insights. For instance, my formalisms bring a theoretical approach to the subject. In addition, my use of program transformations provides a systematic and functional approach to designing systems as well as an implementation-neutral way to study software frameworks for personalization. These properties are absent from the young field of personalization and thus the model I developed for information personalization is my most significant contribution. Moreover, my functional approach makes mixed-initiative interaction, which typically requires dialog and state management mechanisms, feasible without any intended state maintenance. In summary, my programmatic model provides a new way of thinking about personalized interaction, especially with hierarchical hypermedia. Since the role of interaction in personalization is crucial in my opinion, yet often ignored, I believe that my contributions have particular intellectual merit and are especially timely.

7.1 Future Significance and Broader Impacts

Three basic ideas underly the work presented here:

1. explicit modeling of interaction
2. capture of partial information
3. use of program transformations

Any information systems context where one or more of the above ideas apply is fertile ground for implementing the techniques presented in this dissertation.

Modeling interaction is central to dialog management in conversational engines. Using the ideas described here, many important dialog standards and interactive applications can be revisited and studied by their support for personalizing interaction. For instance, consider the VoiceXML markup language designed to simplify the construction of voice-response applications [MBD$^+$01]. It describes interaction using a markup language not unlike that shown in Fig. 2.6: VoiceXML markup tags describe prompts, forms, and fields that constitute a dialog, and support both directed dialogs and mixed-initiative dialogs. In [RCPQ02] Ramakrishnan, Capra, and Pérez-Quiñones have shown that VoiceXML's form-interpretation algorithm is actually a partial evaluator in disguise! Such connections are significant because they reduce commercial technologies to well-established theoretical operations. As another example, user interaction with everyday software (e.g., word processors, spreadsheet applications) can also be modeled for personalization purposes. In [QHKM03] the concept of user interface continuations is introduced, and relies on explicit representations. These projects reinforce my viewpoint that information system design is best approached by first developing representations of interaction. As Marchionini says, 'information seeking is fundamentally an interactive process,' and hence the idea of modeling interaction will always be in vogue.

Until recently, the issue of context in information systems has received comparatively little attention. With the proliferation of mobile environments and information appliances [Ber00], coupled with an improved understanding of their usages [POS$^+$01], the importance of capturing and reusing context has become accentuated. Context can be viewed as a rich form of partial information, allowing the body of work presented here to be applied toward exploiting it. This property is attractive because the partial information can potentially be multi-faceted – information about a user's preferences (e.g., in a recommender system), a user's location (e.g., in a mobile environment [HB01]), or a user's partially completed interaction (e.g., a shopping cart at an e-commerce site). The representations studied in this dissertation can be generalized to accommodate such richer forms of partial information.

Finally, the use of program transformations as presented here is relatable to the larger community that aims to systematize the software engineering of complex web information systems. It finds relevance in many website restructuring/reengineering efforts [RT01a, RTB01, RTB02], especially in the adaptive and semantic web contexts, as well as declaratively specification of sites [FFK$^+$98] and improving their usability [Spi00]. Preliminary work [GZRF01] has also been done to marry the personalization methodology with models for digital libraries (e.g., 5S [GFWK04]).

7.2 Future Work

My modeling makes very few assumptions on the nature of interactions with information systems. While I have covered only hierarchical hypermedia in this thesis, any information system technology that affords the notion of interaction sequence or the idea of factorization [RP01] can be studied on similar lines. Extending the framework to other *focused* information-seeking paradigms is an easy step. More complicated, however, would be the modeling of sites that support *exploratory* interaction, such as websites based on a social network navigation metaphor (e.g., imdb.com).

I also plan to extend the modeling methodology in several directions. I would like to enhance it to address earlier aspects of the personalization system design life cycle, such as requirements gathering, verification, and validation. First steps toward this goal are described in [RRC01]. Another important direction of future work involves modeling context in personalization systems, as described above. I also intend to expand my use of program transformations to study broader concepts from programming languages, such as generalized partial evaluation [FNT91, Tak91], parameterized partial evaluation [CK91, CK93], generative programming [CE00] (e.g., reflection [Mae87]), continuations [FWH01], concept assignment [HGHB02], and program schemas [Ian60]; and the personalized interactions they might enable. I also am interested in relaxing my assumptions of bounded sequences that have separable structural and terminal parts. This will allow us to support more amorphous information-seeking activities through scenario-based design techniques.

My long-term goal is to develop a theory of reasoning about representations of information spaces, akin to [BCST95, BMC93], but in the programming languages spirit of [RT01b, RTB02]. This will allow us to formally study the design and implementation of information systems by the representations they employ.

Bibliography

[ABD$^+$01] J. F. Allen, D. K. Byron, M. Dzikovska, G. Ferguson, L. Galescu, and A. Stent. Towards Conversational Human-Computer Interaction. *AI Magazine*, Vol. 22(4):pp. 27–37, Winter 2001.

[ABS00] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann, 2000.

[ACL$^+$01] J. Axelsson, C. Cross, H. Lie, G. McCobb, T. Raman, and L. Wilson (eds.). Xhtml+voice profile 1.0. W3C Note, December 2001.

[ADW01] C. R. Anderson, P. Domingos, and D. S. Weld. Personalizing Web Sites for Mobile Users. In *Proceedings of the Tenth International World Wide Web Conference (WWW10)*, pp. 565–575, Hong Kong, China, May 2001. ACM Press.

[AGH99] J. F. Allen, C. I. Guinn, and E. Horvitz. Mixed-Initiative Interaction. *IEEE Intelligent Systems*, Vol. 14(5):pp. 14–23, September–October 1999.

[AK97] N. Ashish and C. Knoblock. Wrapper Generation for Semi-Structured Internet Sources. *SIGMOD Record*, Vol. 26(4):pp. 8–15, December 1997.

[AKB91] R. Aluri, D. A. Kemp, and J. J. Boll. Libraries Unlimited, Inc., Englewood, CO, 1991.

[All95] J. Allan. *Automatic Hypertext Construction.* Ph.D. dissertation, Cornell University, 1995.

[Anda] E-mail correspondence with P. Anderson, Senior Software Engineer at GrammaTech, Inc., the company which developed CodeSurfer. April 8, 2004.

[Andb] E-mail correspondence with P. Anderson, Senior Software Engineer at GrammaTech, Inc., the company which developed CodeSurfer. November 10, 2003.

[AR02] E. André and T. Rist. From Adaptive Hypertext to Personalized Web Companions. *Communications of the ACM*, Vol. 45(5):pp. 43–46, May 2002.

[ART03] P. Anderson, T. Reps, and T. Teitelbaum. Design and Implementation of a Fine-Grained Software Inspection Tool. *IEEE Transactions on Software Engineering*, Vol. 29(8):pp. 721–733, August 2003.

[BBE$^+$02] D. Billsus, C. A. Brunk, C. Evans, B. Gladish, and M. Pazzani. Adaptive Interfaces for Ubiquitous Web Access. *Communications of the ACM*, Vol. 45(5):pp. 34–38, May 2002.

[BBH99] P. De Bra, P. Brusilovsky, and G.-J. Houben. Adaptive Hypermedia: From Systems to Framework. *ACM Computing Surveys*, Vol. 31(4es), December 1999. Article No. 12.

[BBJ$^+$03] S. K. Bhavnani, C. K. Bichakjian, T. M. Johnson, R. J. Little, F. A. Peck, J. L. Schwartz, and V. J. Strecher. Strategy Hubs: Next-Generation Domain Portals with Search Procedures. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'03)*, pp. 393–400, Fort Lauderdale, FL, April 2003. ACM Press.

[BC85] J.-F. Bergeretti and B. A. Carré. Information-Flow and Data-Flow Analysis of While-Programs. *ACM Transactions on Programming Languages and Systems*, Vol. 7(1):pp. 37–61, January 1985.

[BC92] N. J. Belkin and W. B. Croft. Information Filtering and Information Retrieval: Two Sides of the Same Coin? *Communications of the ACM*, Vol. 35(12):pp. 29–38, December 1992.

[BC99] R. Bodner and M. Chignell. Dynamic Hypertext: Querying and Linking. *ACM Computing Surveys*, Vol. 31(4es), December 1999. Article No. 15.

[BCD$^+$00] T. Ball, C. Colby, P. Danielsen, L. J. Jagadeesan, R. Jagadeesan, K. Läufer, P. Mataga, and K. Rehor. Sisl: Several Interfaces, Single Logic. *International Journal of Speech Technology*, Vol. 3(2):pp. 93–108, June 2000.

[BCST95] N. J. Belkin, C. Cool, A. Stein, and U. Thiel. Cases, Scripts, and Information-Seeking Strategies: On the Design of Interactive Information Retrieval Systems. *Expert Systems with Applications*, Vol. 9(3):pp. 379–395, 1995.

[Bel97] N. J. Belkin. *Visualizing Subject Access for Twenty-first Century Information Resources*, An Overview of Results from Rutgers' Investigations of Interactive Information Retrieval, pp. 45–62. Number 35. 1997.

[Bel00] N. J. Belkin. Helping People Find What They Don't Know. *Communications of the ACM*, Vol. 43(8):pp. 58–61, August 2000.

[Ber00] E. Bergman, editor. *Information Appliances and Beyond*. The Morgan Kaufmann Series on Interactive Technologies. Morgan Kaufmann, San Francisco, CA, 2000.

[BG96] D. W. Binkley and K. B. Gallagher. Program Slicing. In M. V. Zelkowitz, editor, *Advances in Computers*, Vol. 43, pp. 1–50. 1996.

[BHRC00] D. W. Binkley, M. Harman, R. Raszewsk, and C.Smith. An Empirical Study of Amorphous Slicing as a Program Comprehension Support Tool. In *Proceedings of the IEEE Eighth International Workshop on Program Comprehension*, pp. 161–170, Limerick, Ireland, June 2000. IEEE Computer Society.

[BM02] P. Brusilovsky and M. T. Maybury. From Adaptive Hypermedia to the Adaptive Web. *Communications of the ACM*, Vol. 45(5):pp. 31–33, May 2002.

[BMC93] N. J. Belkin, P. G. Marchetti, and C. Cool. BRAQUE: Design of an Interface to Support User Interaction in Information Retrieval. *Information Processing and Management*, Vol. 29(3):pp. 325–344, May–June 1993.

[Bor86] C. Borgman. The User's Mental Model of an Information Retrieval System: An Experiment on a Prototype On-line Catalogue. *International Journal of Man-Machine Studies*, Vol. 24(1):pp. 47–64, 1986.

[Bru96] P. Brusilovsky. Methods and Techniques of Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, Vol. 6(2–3):pp. 87–129, 1996.

[Bru01] P. Brusilovsky. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, Vol. 11(1–2):pp. 87–110, 2001.

[BSD01] V. Bullard, K. T. Smith, and M. C. Daconta. *Essential XUL Programming.* John Wiley and Sons, Inc., New York, NY, 2001.

[Bus45] V. Bush. As We May Think. *The Atlantic Monthly*, Vol. 176(1):pp. 101–108, July 1945.

[CCDL98] G. Canfora, A. Cimitile, and A. De Lucia. Conditioned Program Slicing. *Information and Software Technology*, Vol. 40(11, 12):pp. 595–607, November/December 1998. Special issue on program slicing.

[CCTL01] W. B. Croft, S. Cronen-Townsend, and V. Larvrenko. Relevance Feedback and Personalization: A Language Modeling Perspective. In *Proceedings of the Joint DELOS-NSF Workshop on Personalisation and Recommender Systems in Digital Libraries*, pp. 49–54, Dublin, Ireland, June 2001. Dublin City University.

[CD97] S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technologies. In *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data (SIGMOD'97)*, pp. 65–74, Tucson, AZ, May 1997. ACM Press.

[CDA00] I. Cingil, A. Dogac, and A. Azgin. A Broader Approach to Personalization. *Communications of the ACM*, Vol. 43(8):pp. 136–141, August 2000.

[CE00] K. Czarnecki and U. W. Eisenecker. *Generative Programming: Methods, Tools, and Applications.* Addison-Wesley, 2000.

[CF94] J.-D. Choi and J. Ferrante. Static Slicing in the Presence of goto Statements. *ACM Transactions on Programming Languages and Systems*, Vol. 16(4):pp. 1097–1113, 1994.

[Cha99] S. S. Chawathe. Describing and Manipulating XML Data. *IEEE Data Engineering Bulletin*, Vol. 22(3):pp. 3–9, September 1999.

[Chi97] Y. Chiaramella. Browsing and Querying: Two Complementary Approaches for Multimedia Information Retrieval. In N. Fuhr, G. Dittrich, and K. Tochtermann, editors, *Proceedings of Hypertext, Information Retrieval, Multimedia (HIM'97)*, pp. 9–26, Dortmund, Germany, September–October 1997.

[CK91] C. Consel and S. C. Khoo. Parameterized Partial Evaluation. In *Proceedings of the Conference on Programming Language Design and Implementation*, Toronto, Canada, July 1991.

[CK93] C. Consel and S. C. Khoo. Parameterized Partial Evaluation. *ACM Transactions on Programming Languages and Systems*, Vol. 15(3):pp. 463–493, July 1993.

[CKPT92] D. Cutting, D. Karger, J. Pedersen, and J. W. Tukey. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. In N. J. Belkin, P. Ingwersen, and A. M. Pejtersen, editors, *Proceedings of the Fifteenth Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR'92)*, pp. 318–329, Copenhagen, Denmark, June 1992. ACM Press.

[CMN80a] S. K. Card, T. P. Moran, and A. Newell. Computer Text-Editing: An Information-Processing Analysis of a Routine Cognitive Skill. *Cognitive Psychology*, Vol. 12:pp. 32–74, 1980.

[CMN80b] S. K. Card, T. P. Moran, and A. Newell. The Keystroke-Level Model for User Performance Time with Interactive Systems. *Communications of the ACM*, Vol. 23(7):pp. 396–410, July 1980.

[CMN83] S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction.* Lawrence Erlbaum, Hillsdale, NJ, 1983.

[CR92] J. M. Carroll and M. B. Rosson. Getting Around the Task-Artifact Cycle: How to Make Claims and Design by Scenario. *ACM Transactions on Information Systems*, Vol. 10(2):pp. 181–212, April 1992.

[CT87] W. B. Croft and R. H. Thompson. I^3R: A New Approach to the Design of Document Retrieval Systems. *Journal of the American Society for Information Science*, Vol. 38(6):pp. 389–404, 1987.

[CXY01] Z. Chen, B. Xu, and H. Yang. Detecting Dead Statements for Concurrent Programs. In *Proceedings of the International Conference on Software Maintenance (SCAM'01)*, pp. 67–72, Florence, Italy, November 2001. IEEE Computer Society.

[DCC01] S. Dumais, E. Cutrell, and H. Chen. Optimizing Search by Showing Results in Context. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'01)*, pp. 277–284, Seattle, WA, April 2001. ACM Press.

[DE95] C. DiGiano and M. Eisenberg. Self-disclosing Design Tools: a Gentle Introduction to End-user Programming. In *Proceedings of the Conference on Designing Interactive Systems (DIS'95)*, pp. 189–197, Ann Arbor, MI, 1995. ACM Press.

[DFF$^+$99] A. Deutsch, M. Fernández, D. Florescu, A. Levy, D. Maier, and D. Suciu. Querying XML Data. *IEEE Data Engineering Bulletin*, Vol. 22(3):pp. 10–18, 1999.

[DOH$^+$02] M. Daoudi, L. Ouarbya, J. Howroyd, S. Danicic, Mark Harman, Chris Fox, and M. P. Ward. ConSUS: A Scalable Approach to Conditioned Slicing. In *Proceedings of the Ninth IEEE Working Conference on Reverse Engineering*, Richmond, VA, October–November 2002. IEEE Computer Society.

[ET99] H. Ehrig and G. Taentzer. Graphical Representation and Graph Transformation. *ACM Computing Surveys*, Vol. 31(3es), September 1999. Article No. 9.

[EV03] M. Eirinaki and M. Vazirgiannis. Web mining for Web Personalization. *ACM Transactions on Internet Technology*, 3(1):pp. 1–27, 2003.

[Fal01] D. C. Fallside, ed. XML Schema W3C Recommendation Document. Technical report, World Wide Web Consortium, May 2001.

[FFK$^+$98] M. Fernández, D. Florescu, J. Kang, A. Levy, and D. Suciu. Catching the Boat with Strudel: Experiences with a Web-Site Management System. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pp. 414–425, Seattle, WA, June 1998. ACM Press.

[FFLS97] M. Fernández, D. Florescu, A. Levy, and D. Suciu. A Query Language for a Web-Site Management System. *SIGMOD Record*, Vol. 26(3):pp. 4–11, September 1997.

[FLM98] D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record*, 27(3):pp. 59–74, September 1998.

[FNT91] Y. Futamura, K. Nogi, and A. Takano. Essence of Generalized Partial Computation. *Theoretical Computer Science*, Vol. 90(1):pp. 61–79, 1991.

[FR97] N. Fuhr and T. Rölleke. A Probabilistic Relational Algebra for the Integration of Information Retrieval and Database Systems. *ACM Transactions on Information Systems*, Vol. 15(1):pp. 32–66, January 1997.

[FRT95] J. Field, G. Ramalingam, and F. Tip. Parametric Program Slicing. In *Proceedings of the Twenty-Second ACM Symposium on Principles of Programming Languages (POPL'95)*, pp. 379–392, San Francisco, CA, January 1995. ACM Press.

[FWH01] D. P. Friedman, M. Wand, and C. T. Haynes. *Essentials of Programming Languages*. MIT Press, Second edition, 2001.

[GBMS99] C. H. Goh, S. Bressan, S. Madnick, and M. Siegel. Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. *ACM Transactions on Information Systems*, Vol. 17(3):pp. 270–293, July 1999.

[GFKF01] P. Graunke, R. Findler, S. Krishnamurthi, and M. Felleisen. Automatically Restructuring Programs for the Web. In *Proceedings of the Sixteenth IEEE International Conference on Automated Software Engineering (ASE'01)*, Coronado, CA, November 2001.

[GFWK04] M. A. Gonçalves, E. A. Fox, L. T. Watson, and N. A. Kipp. Streams, Structures, Spaces, Scenarios, Societies (5S): A Formal Model for Digital Libraries. *ACM Transactions on Information Systems*, 22(2):pp. 270–312, 2004.

[GGR$^+$00] M. N. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. XTRACT: A System for Extracting Document Type Descriptors from XML Documents. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, pp. 165–176, Dallas, TX, May 2000. ACM Press.

[GMPQ$^+$97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, Vol. 8(2):pp. 117–132, March–April 1997.

[Gre86] M. Green. A Survey of Three Dialogue Models. *ACM Transactions on Graphics*, 5(3):pp. 244–275, July 1986.

[GW00] R. Goldman and J. Widom. WSQ/DSQ: A Practical Approach for Combined Querying of Databases and the Web. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data (SIGMOD'00)*, pp. 285–296, Dallas, TX, May 2000. ACM Press.

[GZRF01] M. A. Gonçalves, A. A. Zaferi, N. Ramakrishnan, and E. A. Fox. Modeling and Building Personalized Digital Libraries with PIPE and 5SL. In A. F. Smeaton and J. Callan, editors, *Proceedings of the Joint DELOS-NSF Workshop on Personalisation and Recommender Systems in Digital Libraries*, Dublin, Ireland, June 2001.

[HAC$^+$99] J. M. Hellerstein, R. Avnur, A. Chou, C. Hidber, C. Olston, V. Raman, T. Roth, and P. J. Haas. Interactive Data Analysis: The Control Project. *IEEE Computer*, Vol. 32(8):pp. 51–59, August 1999.

[HB96] S. Henninger and N. J. Belkin. Interface issues and interaction strategies for information retrieval systems. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'96)*, pp. 352–353, Vancouver, Canada, 1996. ACM Press. Tutorial, in Conference Companion.

[HB01] J. Hightower and G. Borriello. Location Systems for Ubiquitous Computing. *IEEE Computer*, Vol. 34(8):pp. 57–66, August 2001.

[HBD03] M. Harman, D. W. Binkley, and S. Danic. Amorphous Program Slicing. *Journal of Systems and Software*, Vol. 68(1):pp. 45–64, October 2003.

[HC90] D. Hildum and J. Cohen. A Language for Specifying Program Transformations. *IEEE Transactions on Software Engineering*, Vol. 16(6):pp. 630–638, June 1990.

[HD97] M. Harman and S. Danicic. Amorphous Program Slicing. In *Proceedings of the Fifth IEEE International Workshop on Program Comprehension (IWPC'97)*, pp. 70–79, Dearborn, MI, May 1997. IEEE Computer Society.

[HDSS96] M. Harman, S. Danicic, Y. Sivagurunathan, and D. Simpson. The Next 700 Slicing Criteria. In M. Munro, editor, *Proceedings of the Second UK Program Comprehension Workshop*, Centre for Software Maintenance, University of Durham, July 1996.

[Hea00] M. A. Hearst. Next Generation Web Search: Setting Our Sites. *IEEE Data Engineering Bulletin*, Vol. 23(3):pp. 38–48, September 2000.

[HEE$^+$02] M. A. Hearst, A. Elliott, J. English, R. Sinha, K. Swearingen, and K.-P. Yee. Finding the Flow in Web Site Search. *Communications of the ACM*, Vol. 45(9):pp. 42–49, September 2002.

[HGHB02] M. Harman, N. Gold, R. Hierons, and D. W. Binkley. Code Extraction Algorithms which Unify Slicing and Concept Assignment. In *Proceedings of the Ninth IEEE Working Conference on Reverse Engineering (WCRE'02)*, pp. 11–21, Richmond, VA, October–November 2002.

[HGMC$^+$97] J. Hammer, H. García-Molina, J. Cho, A. Crespo, and R. Aranha. Extracting Semistructured Information from the Web. In *Proceedings of the NSF–ESPRIT Workshop on Management of Semistructured Data*, pp. 18–25, Tucson, AZ, May 1997.

[HH01] M. Harman and R. Hierons. An Overview of Program Slicing. *Software Focus*, Vol. 2(3):pp. 85–92, June 2001.

[HM97] S. Haller and S. McRoy. Computational Models for Mixed Initiative Interaction (Papers from the 1997 AAAI Spring Symposium). Technical Report SS-97-04, AAAI/MIT Press, 1997.

[Hof02] D. Hofstadter. Analogy as the Core of Cognition. September 2002. Colloquium.

[HR01] D. Hiemstra and S. Robertson. Relevance Feedback for Best Match Term Weighting Algorithms in Information Retrieval. In A. F. Smeaton and J. Callan, editors, *Proceedings of the Joint DELOS-NSF Workshop on Personalisation and Recommender Systems in Digital Libraries*, pp. 37–42, Dublin, Ireland, June 2001. Dublin City University.

[HRB90] S. Horwitz, T. Reps, and D. W. Binkley. Interprocedural Slicing Using Dependency Graphs. *ACM Transactions on Programming Languages and Systems*, Vol. 12(1):pp. 26–60, January 1990.

[Ian60] Y. I. Ianov. The Logical Schemes of Algorithms. In *Problems of Cybernetics*, Vol. 1, pp. 82–140. Pergamon Press, New York, NY, 1960.

[JFM97] T. Joachims, D. Freitag, and T. M. Mitchell. WebWatcher: A Tour Guide for the World Wide Web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pp. 770–777, Nagoya, Aichi, Japan, August 1997. Morgan Kaufmann.

[JGS93a] N. D. Jones, C. K. Gomard, and P. Sestoft. Aspects of Similix: A Partial Evaluator for a Subset of Scheme. In *Partial Evaluation and Automatic Program Generation*, Chapter 10, pp. 204–228. Prentice Hall International, 1993.

[JGS93b] N. D. Jones, C. K. Gomard, and P. Sestoft. *Partial Evaluation and Automatic Program Generation*. Prentice Hall International, 1993.

[JGS93c] N. D. Jones, C. K. Gomard, and P. Sestoft. Partial Evaluation for the C Language. In *Partial Evaluation and Automatic Program Generation*, Chapter 11, pp. 229–259. Prentice Hall International, 1993.

[JK96] B. E. John and D. E. Kieras. The GOMS Family of User Interface Analysis Techniques: Comparison and Contrast. *ACM Transactions on Computer-Human Interaction*, Vol. 3(4):pp. 320–351, December 1996.

[Jon96] N. D. Jones. An Introduction to Partial Evaluation. *ACM Computing Surveys*, Vol. 28(3):pp. 480–503, September 1996.

[Jon97] N. D. Jones. *Computability and Complexity from a Programming Perspective.* Foundations of Computing Series. M.I.T. Press, 1997.

[Jon98] C. V. Jones. Visualization and Optimization. *Interactive Transactions of ORMS*, Vol. 2(1), 1998.

[JPK98] A. Joshi, C. Punyapu, and P. Karnam. Personalization and Asynchronicity to Support Mobile Web Access. In *Proceedings of the Seventh International Conference on Information Knowledge Management (CIKM'98)*, Bethesda, MD, November 1998. ACM Press.

[JR94a] D. Jackson and E. J. Rollins. A New Model of Program Dependences for Reverse Engineering. In D. S. Wile, editor, *Proceedings of the Second ACM SIGSOFT Symposium on Foundations of Software Engineering*, pp. 2–10, New Orleans, LA, December 1994. ACM Press. Also appears in *ACM SIGSOFT Software Engineering Notes*, Vol. 19(5):pp. 2–10, December 1994.

[JR94b] D. Jackson and E. J. Rollins. Chopping: A Generalisation of Slicing. Technical Report CMU-CS-94-169, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1994.

[KB96] J. Koenemann and N. J. Belkin. A Case for Interaction: A Study of Interactive Information Retrieval Behavior and Effectiveness. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'96)*, pp. 205–212, Vancouver, Canada, 1996. ACM Press.

[KH02] S. Kumar and S. Horwitz. Better Slicing of Programs with Jumps and Switches. In *Proceedings of Fundamental Approaches to Software Engineering (FASE'02)*, Grenoble, France, April 2002. Springer-Verlag.

[KL88] B. Korel and J. Laski. Dynamic Program Slicing. *Information Processing Letters*, Vol. 29(3):pp. 155–163, October 1988.

[KL90] B. Korel and J. Laski. Dynamic Slicing of Computer Programs. *Journal of Systems and Software*, Vol. 13(3):pp. 187–195, November 1990.

[KLSS95] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In C. Knoblock and A. Y. Levy, editors, *Information Gathering from Heterogeneous, Distributed Environments*, pp. 85–91. AAAI Press, Stanford, CA, 1995. AAAI Spring Symposium Series Technical Report.

[KMA$^+$98] C. A. Knoblock, S. Minton, J. L. Ambite, N. Ashish, P. J. Modi, I. Muslea, A. G. Philpot, and S. Tejada. Modeling Web Sources for Information Integration. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pp. 211–218, Madison, WI, July 1998. AAAI Press.

[KNV00] J. Kramer, S. Noronha, and J. Vergo. A User-Centered Design Approach to Personalization. *Communications of the ACM*, Vol. 43(8):pp. 45–48, August 2000.

[KS99] D. L. Kreher and D. R. Stinson. *Combinatorial Algorithms: Generation, Enumeration, and Search*, Section 3.2: Set partitions, Bell and Stirling numbers. The CRC Series on Discrete Mathematics and Its Applications. CRC Press, 1999.

[KSS97] H. Kautz, B. Selman, and M. Shah. Referral Web: Combining Social Networks and Collaborative Filtering. *Communications of the ACM*, Vol. 40(3):pp. 63–65, March 1997.

[KWD97] N. Kushmerick, D. S. Weld, and R. Doorenbos. Wrapper Induction for Information Extraction. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI'97)*, pp. 729–737, Nagoya, Aichi, Japan, August 1997. Morgan Kaufmann.

[Lai00] J. Lai. Conversation Interfaces. *Communications of the ACM*, Vol. 43(9):pp. 24–27, September 2000.

[LFW01] H. Lieberman, C. Fry, and L. Weitzman. Exploring the Web with Reconnaissance Agents. *Communications of the ACM*, Vol. 44(8):pp. 69–75, August 2001.

[LG98] S. Lawrence and C. Lee Giles. Searching the World Wide Web. *Science*, Vol. 280:pp. 98–100, April 1998.

[LSCS97] Z. Lacroix, A. Sahuguet, R. Chandrasekar, and B. Srinivas. A Novel Approach to Querying the Web: Integrating Retrieval and Browsing. In D. W. Embley and R. C. Goldstein, editors, *Proceedings of the ER'97 Workshop on Conceptual Modeling of Multimedia Information Seeking*, Los Angeles, CA, November 1997. Springer.

[Luc01] A. De Lucia. Program Slicing: Methods and Applications. In *Proceedings of the First IEEE International Workshop on Source Code Analysis and Manipulation (SCAM'01)*, Florence, Italy, November 2001.

[LW87] J. R. Lyle and M. D. Weiser. Automatic Program Bug Location by Program Slicing. In *Proceedings of the Second International Conference on Computers and Applications*, pp. 877–882, Peking, China, June 1987.

[MAB00] M. D. Mulvenna, S. S. Anand, and A. G. Büchner. Personalization on the Net using Web Mining. *Communications of the ACM*, Vol. 43(8):pp. 122–125, August 2000.

[Mad94] K. H. Madsen. A Guide to Metaphorical Design. *Communications of the ACM*, Vol. 37(12):pp. 57–62, December 1994.

[Mae87] P. Maes. Concepts and Experiments in Computational Reflection. In *Conference Proceedings on Object-Oriented Programming Systems, Languages and Applications*, pp. 147–155, Orlando, FL, October 1987. ACM Press.

[Mar97] G. Marchionini. *Information Seeking in Electronic Environments*. Cambridge Series on Human-Computer Interaction. Cambridge University Press, 1997.

[MB00] P. Maglio and R. Barrett. Intermediaries Personalize Information Streams. *Communications of the ACM*, Vol. 43(8):pp. 96–101, August 2000.

[MBD$^+$01] S. McGlashan, D. Burnett, P. Danielsen, J. Ferrans, A. Hunt, G. Karam, D. Ladd, B. Lucas, B. Porter, K. Rehor, and S. Tryphonas. Voice eXtensible Markup Language: VoiceXML. Technical report, VoiceXML Forum, October 2001. Version 2.0.

[MBG$^+$01] F. Mintzer, G. W. Braudaway, F. P. Giordano, J. C. Lee, Karen A. Magerlein, S. D'Auria, A. Ribak, G. Shapir, F. Schiattarella, J. Tolva, and A. Zelenkov. Populating the Hermitage Museum's New Web Site. *Communications of the ACM*, Vol. 44(8):pp. 52–60, August 2001.

[MCS00] B. Mobashier, R. Cooley, and J. Srivastava. Automatic Personalization Based on Web Usage Mining. *Communications of the ACM*, Vol. 43(8):pp. 142–151, August 2000.

[MLP00] K. D. Munroe, B. Ludäscher, and Y. Papakonstantinou. Blending Browsing and Querying of XML in a Lazy Mediator System. In C. Zaniolo, P. C. Lockemann, M. H. Scholl, and T. Grust, editors, *Proceedings of Seventh International Conference on Extending Database Technology (EDBT'00)*, Konstanz, Germany, March 2000. Springer. In Exhibitions section.

[MM00] R. C. Miller and B. A. Myers. Integrating a Command Shell Into a Web Browser. In *Proceedings of the 2000 USENIX Annual Technical Conference*, pp. 158–166, San Diego, CA, June 2000. The USENIX Association.

[MMLP97] J. Mostafa, S. Mukhopadhyay, W. Lam, and M. Palakal. A Multilevel Approach to Intelligent Information Filtering: Model, System, and Evaluation. *ACM Transactions on Information Systems*, Vol. 15(4):pp. 368–399, October 1997.

[MP00] K. Munroe and Y. Papakonstantinou. BBQ: A Visual Interface for Browsing and Querying XML. In H. Arisawa and T. Catarci, editors, *Proceedings of Fifth Working Conference on Visual Database Systems (VDB5)*, Fukuoka, Japan, May 2000. Kluwer Academic Publishers.

[MP02] P. Mukhopadhyay and Y. Papakonstantinou. Mixing Querying and Navigation in MIX. In *Proceedings of the Eighteenth International Conference on Data Engineering (ICDE'02)*, San Jose, CA, February–March 2002.

[MPR00] U. Manber, A. Patel, and J. Robinson. Experience with Personalization on Yahoo! *Communications of the ACM*, Vol. 43(8):pp. 35–39, August 2000.

[MS01] H. Meuss and K. U. Schulz. Complete Answer Aggregates for Treelike Databases: A Novel Approach to Combine Querying and Navigation. *ACM Transactions on Information Systems*, Vol. 19(2):pp. 161–215, April 2001.

[MTW95] R. J. Miller, O. G. Tsatalos, and J. H. Williams. Integrating Hierarchical Navigation and Querying: A User Customizable Solution. In I. F. Cruz, J. Marks, and K. Wittenburg, editors, *Proceedings of ACM Workshop on Effective Abstractions in Multimedia Layout, Presentation, and Interaction*, San Francisco, CA, November 1995. ACM Press.

[MVPB93] P. G. Marchetti, S. Vazzana, R. Panero, and N. J. Belkin. BRAQUE (abstract): An Interface to Support Browsing and Interactive Query Formulation in Information Retrieval Systems. In *Proceedings of the Sixteenth Annual International ACM Conference on Research and Development in Information Retrieval (SIGIR'93)*, p. 358, Pittsburgh, PA, June–July 1993. ACM Press.

[NAM97] S. Nestorov, S. Abiteboul, and R. Motwani. Inferring Structure in Semistructured Data. *SIGMOD Record*, Vol. 26(4):pp. 39–43, December 1997. Special Issue on Management of Semi-Structured Data.

[NAM98] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting Schema From Semistructured Data. In L. M. Haas and A. Tiwary, editors, *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pp. 295–306, Seattle, WA, June 1998. ACM Press.

[NS97] D. G. Novick and S. Sutton. What is Mixed-Initiative Interaction? In S. Haller and S. McRoy, editors, *Proceedings of the AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction*, pp. 114–116. AAAI/MIT Press, 1997.

[NWPR04] M. Narayan, C. Williams, S. Perugini, and N. Ramakrishnan. Staging Transformations for Multimodal Web Interaction Management. In *Proceedings of the Thirteenth ACM International World Wide Web Conference (WWW'04)*, pp. 212–223, New York, NY, May 2004. ACM Press.

[OC03] C. Olston and E. H. Chi. ScentTrails: Integrating Browsing and Searching on the Web. *ACM Transactions on Computer-Human Interaction*, Vol. 10(3):pp. 177–197, September 2003.

[O'L97] D. O'Leary. The Internet, Intranets, and the AI Renaissance. *IEEE Computer*, Vol. 30(1):pp. 71–78, January 1997.

[Pan01] C. Pancake. The Ubiquitous Beauty of User-Aware Software. *Communications of the ACM*, Vol. 44(3):p. 130, March 2001.

[Par98] D. L. Parnas. Successful Software Engineering Research. *ACM SIGSOFT Software Engineering Notes*, Vol. 23(3):pp. 64–68, May 1998.

[PE00] M. Perkowitz and O. Etzioni. Adaptive Web Sites. *Communications of the ACM*, Vol. 43(8):pp. 152–158, August 2000.

[Ped00] E. P. D. Pednault. Representation is Everything. *Communications of the ACM*, Vol. 43(8):pp. 80–83, August 2000.

[PGF04] S. Perugini, M. A. Gonçalves, and E. A. Fox. Recommender Systems Research: A Connection-Centric Survey. *Journal of Intelligent Information Systems*, Vol. 23(2):pp. 107–143, September 2004.

[PMB96] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill and Webert: Identifying Interesting Web Sites. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 54–61, Portland, OR, August 1996. AAAI Press.

[PMR$^+$04] S. Perugini, K. McDevitt, R. Richardson, M. A. Pérez-Quiñones, R. Shen, N. Ramakrishnan, C. Williams, and E. A. Fox. Enhancing Usability in CITIDEL: Multimodal, Multilingual, and Interactive Visualization Interfaces. In *Proceedings of the Fourth ACM/IEEE Joint Conference on Digital Libraries (JCDL'04)*, pp. 315–324, Tucson, AZ, June 2004. ACM Press.

[Pok01] J. Pokorny. Static Pages are Dead: How a Modular Approach is Changing Interaction Design. *ACM Interactions*, Vol. 8(5):pp. 19–24, September–October 2001.

[POS$^+$01] M. Perry, K. O'Hara, A. Sellen, B. Brown, and R. Harper. Dealing with Mobility: Understanding Access Anytime, Anywhere. *ACM Transactions on Computer-Human Interaction*, 8(4):pp. 323–347, 2001.

[PPR$^+$03] S. Perugini, M. E. Pinney, N. Ramakrishnan, M. A. Pérez-Quiñones, and M. B. Rosson. Taking the Initiative with Extempore: Exploring Out-of-turn Interaction Interfaces with Faceted Websites. Technical Report cs.HC/0312016, Computing Research Repository (CoRR), December 2003. In communication to *ACM Transactions on Computer–Human Interaction*, May 2004.

[PQ96] M. A. Pérez-Quiñones. *Conversational Collaboration in User-initiated Interruption and Cancellation Requests*. Ph.D. dissertation, The George Washington University, May 1996.

[PRa] S. Perugini and N. Ramakrishnan. A Programmatic Model for Information Personalization. In communication to *IEEE Transactions on Software Engineering*, May 2004.

[PRb] S. Perugini and N. Ramakrishnan. Program Transformations for Information Personalization. In communication to *ACM Transactions on Internet Technology*, May 2004.

[PR03a] S. Perugini and N. Ramakrishnan. Personalizing Interactions with Information Systems. In M. V. Zelkowitz, editor, *Advances in Computers*, Vol. 57: Information Repositories, pp. 323–382. Academic Press, September 2003.

[PR03b] S. Perugini and N. Ramakrishnan. Personalizing Web Sites with Mixed-Initiative Interaction. *IEEE IT Professional*, Vol. 5(2):pp. 9–15, March–April 2003.

[PRF04] S. Perugini, N. Ramakrishnan, and E. A. Fox. Automatically Generating Interfaces for Personalized Interaction with Digital Libraries. Technical Report cs.DL/0402022, Computing Research Repository (CoRR), February 2004.

[PS83] H. Partsch and R. Steinbrüggen. Program Transformation Systems. *ACM Computing Surveys*, 15(3):pp. 199–236, 1983.

[PSC$^+$02] J. Pitkow, H. Schütze, T. Cass, R. Cooley, D. Turnbull, A. Edmonds, E. Adar, and T. Breuel. Personalized Search. *Communications of the ACM*, Vol. 45(9):pp. 50–55, September 2002.

[PSDB99] C. Plaisant, B. Shneiderman, K. Doan, and Tom Bruns. Interface and Data Architecture for Query Preview in Networked Information Systems. *ACM Transactions on Information System*, Vol. 17(3):pp. 320–341, July 1999.

[QHKM03] D. Quan, D. Huynh, D. R. Karger, and R. Miller. User Interface Continuations. In *Proceedings of the Sixteenth Annual ACM Symposium on User Interface Software and Technology (UIST'03)*, pp. 145–148, Vancouver, Canada, November 2003. ACM Press.

[Que00] C. Queinnec. The Influence of Browsers on Evaluators or, Continuations to Program Web Servers. In *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP'00)*, pp. 23–33, Montreal, Canada, September 2000. ACM Press. Also appears in *ACM SIGPLAN Notices*, Vol. 35(9), September 2000.

[Ram00] N. Ramakrishnan. PIPE: Web Personalization by Partial Evaluation. *IEEE Internet Computing*, Vol. 42(9):pp. 21–31, November–December 2000.

[RB69] B. Randell and J. N. Buxton, editors. *Software Engineering Techniques: Report of a Conference sponsored by the NATO Science Committee*, Rome, Italy, October 1969. Brussels, Scientific Affairs Division, NATO (1970). 164 pp..

[RCCR02a] G. G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins. Animated Visualization of Multiple Intersecting Hierarchies. *Information Visualization*, Vol. 1:pp. 50–65, April 2002.

[RCCR02b] G. G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins. Polyarchy Visualization: Visualizing Multiple Intersecting Hierarchies. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'02)*, pp. 423–430, Minneapolis, MN, April 2002. ACM Press.

[RCPQ02] N. Ramakrishnan, R. Capra, and M. A. Pérez-Quiñones. Mixed-Initiative Interaction = Mixed Computation. In P. Thiemann, editor, *Proceedings of the ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation (PEPM'02)*, pp. 119–130, Portland, OR, January 2002. ACM Press. Also appears in *ACM SIGPLAN Notices*, Vol. 37(3), March 2002.

[Rie00a] D. Riecken. Personal End-User Tools. *Communications of the ACM*, Vol. 43(8):pp. 89–91, August 2000.

[Rie00b] D. Riecken. Personalized Views of Personalization. *Communications of the ACM*, Vol. 43(8):pp. 27–28, August 2000.

[Rie01] J. Riedl. Personalization and Privacy. *IEEE Internet Computing*, Vol. 5(6):pp. 29–31, November-December 2001.

[Roc71] J. J. Rocchio. Relevance Feedback in Information Retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pp. 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.

[Ros99] M. B. Rosson. Integrating Development of Task and Object Models. *Communications of the ACM*, Vol. 42(1):pp. 49–56, January 1999.

[RP97] J. Rcuker and M. J. Polano. Siteseer: Personalized Navigation for the Web. *Communications of the ACM*, Vol. 40(3):pp. 73–75, March 1997.

[RP01] N. Ramakrishnan and S. Perugini. The Partial Evaluation Approach to Information Personalization. Technical Report cs.IR/0108003, Computing Research Repository (CoRR), August 2001. In communication to *Information Processing and Management*.

[RR95] T. Reps and G. Rosay. Precise Interprocedural Chopping. In G. E. Kaiser, editor, *Proceedings of the Third ACM SIGSOFT Symposium on Foundations of Software Engineering*, pp. 41–52, Washington, DC, October 1995. ACM Press. Also appears in *ACM SIGSOFT Software Engineering Notes*, Vol. 20(4):pp. 41–52, October 1995.

[RRC01] N. Ramakrishnan, M. B. Rosson, and J. M. Carroll. Explaining Scenarios for Information Personalization. Technical Report cs.HC/0312016, Computing Research Repository (CoRR), November 2001.

[RS97] D. Rus and D. Subramanian. Customizing Information Capture and Access. *ACM Transactions on Information Systems*, Vol. 15(1):pp. 67–101, January 1997.

[RT96] T. Reps and T. Turnidge. Program specialization via Program Slicing. In O. Danvy, R. Glueck, and P. Thiemann, editors, *Proceedings of the Dagstuhl Seminar on Partial Evaluation; Lecture Notes in Computer Science*, Vol. 1110, pp. 409–429, Schloss Dagstuhl, Wadern, Germany, February 1996. Springer-Verlag.

[RT01a] F. Ricca and P. Tonella. Understanding and Restructuring Web Sites with ReWeb. *IEEE MultiMedia*, Vol. 8(2):pp. 40–51, April–June 2001.

[RT01b] F. Ricca and P. Tonella. Web Application Slicing. In *Proceedings of the International Conference on Software Maintenance (ICSM'01)*, pp. 148–157, Florence, Italy, November 2001. IEEE Computer Society.

[RTB01] F. Ricca, P. Tonella, and I. D. Baxter. Restructuring Web Applications via Transformation Rules. In *Proceedings of the First International Workshop on Source Code Analysis and Manipulation (SCAM'01)*, pp. 150–160, Florence, Italy, November 2001. IEEE Computer Society.

[RTB02] F. Ricca, P. Tonella, and I. D. Baxter. Web Application Transformations based on Rewrite Rules. *Information and Software Technology*, Vol. 44(13):pp. 811–825, October 2002.

[RV97] P. Resnick and H. R. Varian. Recommender Systems. *Communications of the ACM*, Vol. 40(3):pp. 56–58, March 1997.

[SA99] A. Sahuguet and F. Azavant. Looking at the Web through XML Glasses. In *Proceedings of the Fourth IFCIS International Conference on Cooperative Information Systems (CoopIs'99)*, pp. 148–159, Edinburgh, Scotland, September 1999. IEEE Computer Society.

[Sac00] G. M. Sacco. Dynamic Taxonomies: A Model for Large Information Bases. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12(3):pp. 468–479, May–June 2000.

[Sac02] G. M. Sacco. Conventional Taxonomies vs. Dynamic Taxonomies, 2002. Communicated for publication.

[SAL02] Speech Application Language Tags (SALT) Specification. Technical report, SALT Forum, July 2002. Version 1.0.

[SB02] S. Srinivasan and E. Brown. Is Speech Recognition Becoming Mainstream? *IEEE Computer*, Vol. 35(4):pp. 38–41, April 2002.

[SCDT00] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data. *SIGKDD Explorations*, Vol. 1(2):pp. 12–23, January 2000.

[Shn92] B. Shneiderman. Tree Visualization with Tree-Maps: 2-D Space-Filling Approach. *ACM Transactions on Graphics*, Vol. 11(1):pp. 92–99, January 1992.

[Smi00] D. C. Smith. Building Personal Tools by Programming. *Communications of the ACM*, Vol. 43(8):pp. 92–95, August 2000.

[Spi00] M. Spiliopoulou. Web Usage Mining for Web Site Evaluation. *Communications of the ACM*, Vol. 43(8):pp. 127–134, August 2000.

[Suc87] L. A. Suchman. *Plans and Situated Actions: The Problem of Human-Machine Communication.* Cambridge University Press, 1987.

[SV99] C. Shapiro and H. R. Varian. *Information Rules: A Strategic Guide to the Network Economy.* Harvard Business School Press, November 1999.

[SW93] M. F. Schwartz and D. C. M. Wood. Discovering Shared Interests Using Graph Analysis. *Communications of the ACM*, Vol. 36(8):pp. 78–89, August 1993.

[SW01] B. Shneiderman and M. Wattenberg. Ordered Treemap Layouts. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'01)*, pp. 73–78, San Diego, CA, October 2001. IEEE Computer Society.

[SYV01] M. P. Singh, B. Yu, and M. Venkatraman. Community-Based Service Location. *Communications of the ACM*, Vol. 44(4):pp. 49–54, April 2001.

[Tak91] A. Takano. Generalized Partial Computation for a Lazy Functional Language. In *Proceedings of the ACM SIGPLAN Symposium on Partial Evaluation and Semantics-Based Program Manipulation (PEPM'91)*, pp. 1–11, New Haven, CT, 1991. ACM Press.

[THA+97] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. PHOAKS: A System for Sharing Recommendations. *Communications of the ACM*, Vol. 40(3):pp. 59–62, March 1997.

[Tho98] B. Thomas. URL Diving. *IEEE Internet Computing*, Vol. 2(3):pp. 92–93, May–June 1998.

[Tid01] D. Tidwell. *XSLT.* O'Reilly and Associates, Inc., Sebastopol, CA, 2001.

[Tip95] F. Tip. A Survey of Program Slicing Techniques. *Journal of Programming Languages*, Vol. 3(3):pp. 121–189, 1995.

[TL98] H. B. K. Tan and T. W. Ling. Correct Program Slicing of Database Operations. *IEEE Software*, 15(2), March/April 1998.

[TMK+02] J. J. Thomas, D. R. McGee, O. A. Kuchar, J. W. Graybeal, D. L. McQuerry, and P. L. Novak. What is your Relationship with your Information Space? In *Proceedings of Computer Graphics International*, Bradford, UK, July 2002.

[Van01] M. L. Van De Vanter. Preserving the Documentary Structure of Source Code in Language-based Transformation Tools. In *Proceedings of the International Conference on Software Maintenance (SCAM'01)*, pp. 131–141, Florence, Italy, November 2001. IEEE Computer Society.

[Ven91] V. A. Venkatesh. The Semantic Approach to Program Slicing. In *Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 107–119, Toronto, Canada, June 1991. ACM Press. Also appears in *ACM SIGPLAN Notices*, Vol. 26(6):pp. 107–109, 1991.

[War99] M. Ward. Assembler to C Migration using the FermaT Transformation System. In *Proceedings of the International Conference on Software Maintenance*, Oxford, England, August–September 1999.

[War00] M. Ward. Reverse Engineering from Assembler to Formal Specifications via Program Transformations. In *Proceedings of the Seventh Working Conference on Reverse Engineering*, Brisbane, Queensland, Australia, November 2000. IEEE Computer Society.

[War01a] M. Ward. The FermaT Assembler Re-engineering Workbench. In *Proceedings of the International Conference on Software Maintenance*, Florence, Italy, November 2001. IEEE Computer Society.

[War01b] M. Ward. The Formal Transformation Approach to Source Code Analysis and Manipulation. In *Proceedings of the First International Workshop on Source Code Analysis and Manipulation*, Florence, Italy, November 2001. IEEE Computer Society. Keynote speech.

[War02] M. Ward. Program Slicing via FermaT Transformations. In *Proceedings of the Twenty-sixth Annual International Computer Software and Applications Conference (COMPSAC'02)*, Oxford, England, August 2002. IEEE Computer Society.

[WB99] R. M. Wilson and R. D. Bergeron. Dynamic Hierarchy Specification and Visualization. In *Proceedings of the IEEE Symposium on Information Visualization (INFOVIS'99)*, pp. 65–72, San Francisco, CA, October 1999. IEEE Computer Society.

[Wei79] M. Weiser. *Program Slices: Formal, Psychological, and Practical Investigations of an Automatic Program Abstraction Method*. Ph.D. dissertation, University of Michigan, 1979.

[Wei82] M. Weiser. Programmers use Slices when Debugging. *Communications of the ACM*, 25(7):pp. 446–552, July 1982.

[Wei84] M. Weiser. Program Slicing. *IEEE Transactions on Software Engineering*, 10(4):pp. 352–357, July 1984.

[Wei91] M. Weiser. The Computer for the Twenty-First Century. *Scientific American*, pp. pp. 94–104, September 1991.

[Wid99] J. Widom. Data Management for XML: Research Directions. *IEEE Data Engineering Bulletin*, Vol. 22(3):pp. 44–52, September 1999.

[Wil84] M. D. Williams. What makes RABBIT run? *International Journal of Man-Machine Studies*, Vol. 21:pp. 333–352, 1984.

[Wil04] C. S. Williams. WS://IM: A Software Framework for Multimodal Web Interaction Management. Master's thesis, Department of Computer Science, Virginia Tech, May 2004.

[WM99] A. Wexelblat and P. Maes. Footprints: History-Rich Tools for Information Foraging. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'99)*, pp. 270–277, Pittsburgh, PA, May 1999. ACM Press.

[Wyn00] B. S. Wynar. In A. G. Taylor, editor, *Waynar's Introduction to Cataloging and Classification*, Library and Information Science Text Series. Libraries Unlimited, Inc., Englewood, CO, Ninth edition, 2000.

[WZ91] W. N. Wegman and F. K. Zadeck. Constant Propagation with Conditional Branches. *ACM Transactions on Programming Languages and Systems*, Vol. 13(2):pp. 181–210, April 1991.

[Xie02] H. Xie. Patterns between Interactive Intentions and Information-Seeking Strategies. *Information Processing and Management*, Vol. 38(1):pp. 55–77, January–February 2002.

[Yan96] N. Yankelovich. How Do Users Know What To Say? *ACM Interactions*, 3(6):pp. 32–43, November–December 1996.

[ZBC$^+$00] W. Zadrozny, M. Budzikowski, J. Chai, N. Kambhatla, S. Levesque, and N. Nicolov. Natural Language Dialogue for Personalized Interaction. *Communications of the ACM*, Vol. 43(8):pp. 116–120, August 2000.

[Zhu02] Q. Zhu. 5SGraph: A Modeling Tool for Digital Libraries. Master's thesis, Department of Computer Science, Virginia Tech, November 2002.

Appendix A

XSchema OTML Language Definition

```
<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">

  <xsd:annotation>
    <xsd:documentation>This is the XML Schema for OTML.</xsd:documentation>
  </xsd:annotation>

  <xsd:element name="otml" type="OTMLType"/>

  <xsd:complexType name="OTMLType">
    <xsd:element name="input-textbox-label" type="xsd:string" minOccurs="0"/>
    <xsd:element name="input-textbox-maxlength" type="xsd:positive-integer" minOccurs="0"/>
    <xsd:element name="submit-button-label" type="xsd:string" minOccurs="0"/>
    <xsd:element name="clear-button-label" type="xsd:string" minOccurs="0"/>
    <xsd:element name="interactions" type="Interactions" minOccurs="0"/>
  </xsd:complexType>

  <xsd:complexType name="Interactions">
    <xsd:element name="what" type="What" minOccurs="0"/>
    <xsd:element name="restructure-classification" type="RestructureClassification" minOccurs="0"/>
    <xsd:element name="collect-results" type="CollectResults" minOccurs="0"/>
    <xsd:element name="inverse-personalization" type="InversePersonalization" minOccurs="0"/>
  </xsd:complexType>

  <xsd:complexType name="What">
    <xsd:element name="label" type="xsd:string" minOccurs="0"/>
    <xsd:element name="tip" type="xsd:string" minOccurs="0"/>
  </xsd:complexType>

  <xsd:complexType name="RestructureClassification">
    <xsd:element name="label" type="xsd:string" minOccurs="0"/>
    <xsd:element name="tip" type="xsd:string" minOccurs="0"/>
  </xsd:complexType>

  <xsd:complexType name="CollectResults">
    <xsd:element name="label" type="xsd:string" minOccurs="0"/>
    <xsd:element name="tip" type="xsd:string" minOccurs="0"/>
  </xsd:complexType>

  <xsd:complexType name="InversePersonalization">
    <xsd:element name="label" type="xsd:string" minOccurs="0"/>
    <xsd:element name="tip" type="xsd:string" minOccurs="0"/>
  </xsd:complexType>

</xsd:schema>
```

Appendix B

Some Miscellaneous Program Transformations

This dissertation has primarily focused on the use of partial evaluation and static program slicing for information personalization. In this appendix, I identify some extensions to these transformations which can potentially be incorporated into a personalization model.

B.1 Dynamic Slicing

Unlike static slicing, the computation of a *dynamic slice* [KL88, KL90] considers program input. The slicing criterion for a dynamic slice is thus augmented for the program's input and therefore a triple. In addition, the statement component of the criterion becomes a particular occurrence (e.g., first, second, or third, denoted with a superscript) of a statement w.r.t. the execution history of the program. Fig. B.1 (right) illustrates a dynamic slice (of left). For an introduction to program slicing, techniques for computing slices, and applications, I refer the interested reader to [BG96, HH01, Tip95].

Program slicing was originally observed as a task which programmers manually performed while debugging, analyzing, and understanding programs [Wei82]. It has since been automated and thus reapplied to those tasks and additionally applied in many diverse areas of software engineering. Application areas include (semantic) program differencing and integration, testing, software maintenance and metrics, reverse engineering, (component) re-use, (functional) cohesion measurement and management, software quality assurance, software conformance certification, software safety, parallelization, and compiler tuning. The application of slicing in these domains is collectively covered or referenced in [BG96, HBD03, HH01, Tip95]. A modified version of slicing has been developed to address shortcomings when applied to database operations involving I/O [TL98]. Lastly and useful for my purposes, static slicing methods can detect dead-code [BC85]. Detection is typically conducted in a debugging context since such statements are usually unexecutable due to the presence of a bug [Tip95].

Current research in program slicing entails relaxing or removing constraints of characteristics of slices which adversely affect their accuracy [Tip95]. Rather than developing new algorithms for constructing existing slices, some researchers are focused on articulating new

```
line | program            | dynamic (a=1; b=3; c=2, 11^1, max)
-----+--------------------+------------------------------------
 (1) | read (a, b, c);    | read (a, b, c);
 (2) | if (a > b)         | if (a > b)
 (3) |   if (a > c)       |
 (4) |     max = a;       |
 (5) |   else             |
 (6) |     max = a;       |
 (7) | else if (b > c)    | else if (b > c)
 (8) |        max = b;    |        max = b;
 (9) |      else          |      else
(10) |        max = c;    |
(11) | write (max);       | write (max);
```

Figure B.1: Illustration of dynamic slicing. (left) A program which computes the maximum of three integers. (right) A dynamic slice (of left) w.r.t. (a=1; b=3; c=2, 11^1, `max`).

slicing criteria [HDSS96], which typically results in a new type of slice. A survey of some new slices and a comparison of a variety of slicing criteria for these are presented in [HDSS96]. This research is important because it provides opportunity for new application domains.

B.2 More Variants of Slicing

While [BG96, HH01, Tip95] focus primarily on static and dynamic program slicing (backward and forward; executable and closure), there are many other variants of slicing [Luc01, HDSS96, HH01] (I refer the reader to [Ven91] for an introduction, including semantic definitions, of these eight combinations.). For instance, *quasi-static slicing*, introduced in [Ven91], was the first attempt at a hybrid slice between static and dynamic slicing [Luc01]. Others have articulated richer and more generalized versions of quasi-static slicing. For example, *conditioned slicing* [CCDL98] is also a hybrid between static and dynamic slicing. While the slicing criteria for a static slice contains no conditions on the initial state (i.e., specific values for program inputs are not considered), the criteria of a dynamic slice contains a particular initial state (i.e., assignment of values to *all* inputs; see Fig. B.1). The criteria for a conditioned slice places a condition on the initial state of the program. In other words, rather than a total assignment of program inputs, only a *partial* assignment is required. Conditioned slicing is therefore similar to partial evaluation [Luc01, War02]. As opposed to the traditional direction in which slices are computed (i.e., backward), conditioned slicing typically uses forward conditioning [Luc01]. Analogously, backward conditioning [Ven91] has been proposed for program comprehension. For more information on conditioned slicing, see [HDSS96, HH01]. Another generalization is *parametric slicing* [FRT95], which yields *constrained slices*, where 'parametric' refers to the program input and the 'constraint' refers to the amount of input given [BG96].

Amorphous slicing [HD97], which is a combination of slicing and non-syntax preserving transformations, is yet another slicing method. Executable backward static slices have been called *syntactic slices* since they are syntax-preserving [War01b]. In other words, they are

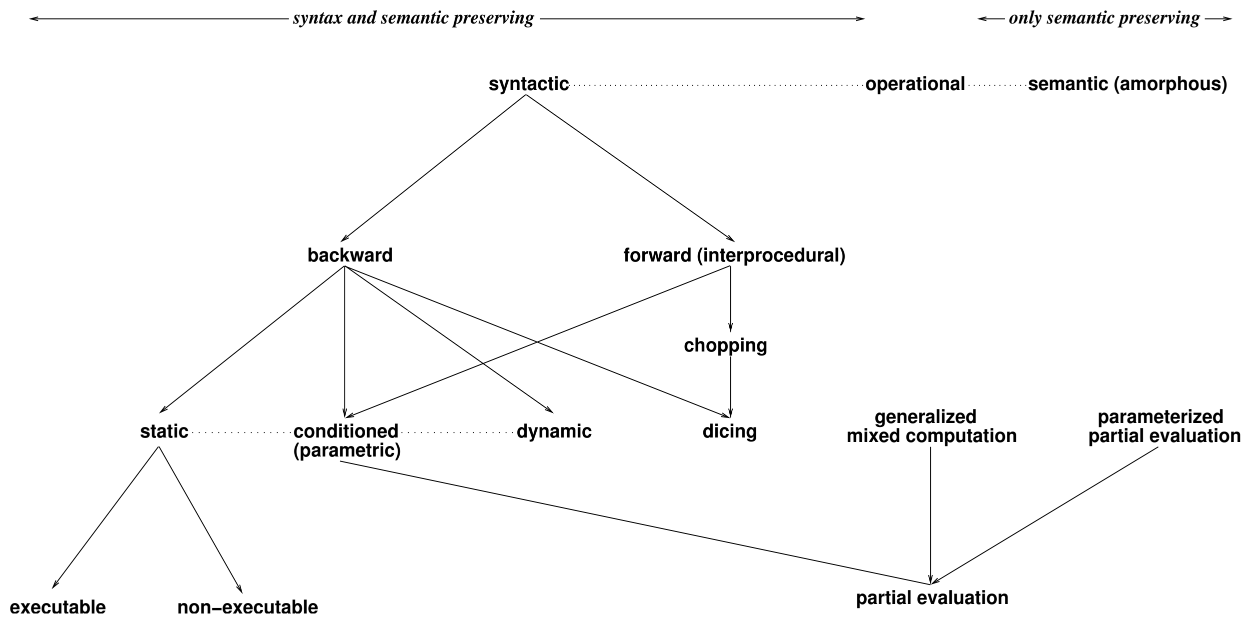


Figure B.2: A taxonomy of program transformations, including partial evaluation and slicing. A directed arrow represents a specialization relation. An undirected line represents an association, while a dotted line represents a range.

created by only deleting (irrelevant) statements from the original program. An amorphous slice on the other hand, which has analogously been called a *semantic slice* [War01b], need not be a subset of the original program. It can be realized by any transformation which preserves the original program's semantics w.r.t. the slicing criteria. Amorphous slicing makes bugs salient in programs and has been used as a program comprehension tool to help students find array out-of-bounds errors [BHRC00, HBD03].

Analogous to conditioned slicing, *operational slicing* [War02] is a hybrid between syntactic and semantic slicing. Operational slicing is an intermediate operation to preserve operational semantics [War02]. I refer the interested reader to [BG96, BHRC00, Luc01, HDSS96, HH01, Tip95, War01b, War02], which collectively discuss or reference these derivatives.

B.3 Variants of Partial Evaluation

While traditional partial evaluation may only exploit static values for program input, *generalized partial computation* may utilize additional properties of a program, such as logical structure, abstract data types, and primitive functions [FNT91]. Similarly, *parameterized partial evaluation* is 'a generic form of partial evaluation w.r.t. user-defined static properties' [CK93]. These two variants of partial evaluation may be helpful to support information seeking which entails pursuing multiple interaction sequences simultaneously. Such information-seeking activities require transformations capable of preserving multiple flows of control. Fig. B.2 presents a taxonomy of program transformations, including some of those discussed in this appendix.

Appendix C

Program Transformation Systems

Program transformation systems conduct source-to-source transformations which are typically implemented as tree-to-tree rewrites [ET99]. I used C-Mix and Similix, two partial evaluators for C and (a large higher-order subset of) Scheme, respectively, to experiment with the program transformation techniques involving partial evaluation (not-syntax-preserving, semantic-preserving transformations). C-Mix [JGS93c] is an off-line partial evaluator meaning that it specializes programs at compile time, but not at run time. It is developed and maintained by the TOPPS (which approximately abbreviates 'Semantics-based Program Analysis and Manipulation' in Danish) group at DIKU (the Department of Computer Science at the University of Copenhagen, Denmark). Neil D. Jones, the author of many highly cited articles on partial evaluation [Jon96], is a member of TOPPS which is the leading research group on partial evaluation in Europe. C-Mix is available at http://www.diku.dk/forskning/topps/activities/cmix/. Similix [JGS93a] also is an off-line partial evaluator developed and maintained by the TOPPS group at DIKU. Similix is available at http://www.diku.dk/forskning/topps/activities/similix.html. I used CodeSurfer and the FermaT Transformation Engine, two popular program transformation systems, to experiment with program slicing (and other syntax-preserving, not-semantic preserving transformations).

CodeSurfer [ART03] is the academic and industry state of the art tool for program slicing. It analyzes ANSI C and relies on 'system dependence graphs' as a fundamental intermediate structure to represent programs [ART03]. In addition to slicing, CodeSurfer can perform program dicing and chopping [JR94b, JR94a, RR95]. CodeSurfer has been used for applications ranging from computing amorphous slices for student program comprehension [BHRC00] to security and safety assurance. For more information, see http://www.grammatech.com/products/codesurfer/.

FermaT [War02, War01a] also is an academic and industry strength transformation system. It is based on the WSL (Wide Spectrum Language) [War01b, War02] transformation theory. FermaT has been used commercially in assembler to C migration projects [War99] and to reverse engineer from assembler to formal specifications [War00]. FermaT can compute several slices, including conditioned slices [DOH$^+$02]. For more information, see http://www.cse.dmu.ac.uk/∼mward/fermat.html.

I refer the interested reader to [PS83] for a detailed survey of program transformation systems.

Appendix D

Problem-Solving Task for Evaluating Interaction Interfaces

It is course scheduling time, once again, at Virginia Tech, and you need to develop your schedule of courses for the following semester. You have a full-time job and are working on your undergraduate degree as a part-time student. You will complete your degree next semester. Your task is to develop a schedule of courses which meets *all* of the following requirements.

Hint: This problem is a puzzle. Please read the entire set of conditions completely and carefully before you begin to develop your schedule. Not only must the schedule you develop meet all of the following conditions, but it also must have no time/day conflicts (i.e., the schedule must permit you to attend every class meeting in full). You may assume that you have the prerequisite(s) and corequisite(s) for any course.

1. You are an undergraduate student and therefore are eligible for only undergraduate courses.
2. To accommodate your full-time work schedule you want to take as many online courses as possible. However, a departmental policy states that no student may take more than one online course per semester.
3. You do not want to take more than 4 courses.
4. Your work schedule permits you to come to campus only twice per week:
    - Tuesdays at or after 4p.
    - Thursdays at any time after 10a.
5. A departmental policy forbids you from taking more than 2 math courses in one semester.
6. In an effort to minimize your trips to campus, you want a course that meets only on Tuesday. The course must start at 4p or later & should not be a lab or recitation.

7. You only need 11 more credits to graduate. You are on a strict budget. Since tuition is proportional to number of credits and costly, you can only afford to register for the minimum number of credits necessary to fulfill your credit requirement for your degree.
8. You want to take a statistics course as you feel it will be useful in the future.
9. You are not keen on taking courses 'for fun' and therefore do not want to schedule any 0-credit courses.
10. You are not pursuing a thesis of any form and therefore are not eligible for research credits.
11. You have been told that independent studies are flexible, but consume an enormous amount of time. Since you work full-time, you cannot afford to trade time for flexibility, and therefore do not want to take an independent study.
12. You are very busy with your full-time job and don't want to be bothered by arranged courses.

Provide your answers in the following table:

	Course Abbrev. and No.	**Day(s)**	**Time**	**Credit(s)**
1.		online	online	
2.		Tuesday		
3.				
4.				
			Total:	11

Vita

Saverio Perugini was born in Waterbury, Connecticut on December 15, 1976. After garnering an interest in computers while casually programming in Pascal on an IBM XT during his middle school years, his interests shifted to basketball during high school. Although still a passion, his hoop dreams ended with his final high school varsity game in March 1994 at Sacred Heart High School (but provided copious amounts of stress relief while in graduate school!).

In August 1994, he relocated to Main Line Philadelphia to study at Villanova University with the Augustinians. He spent the next four years there studying computer science, and theatre to escape the vacuum of a technical major. While at Villanova, Perugini contributed to the development of a leading CS1 textbook: *Java Software Solutions*, by J. A. Lewis (his undergraduate advisor) and W. P. Loftus (his employer during his undergraduate years). Perugini completed a Bachelor of Science in Computer Science in May 1998. His work with John Lewis at Villanova ultimately led him to pursue graduate studies at Virginia Tech.

In August 1998, he joined the Department of Computer Science at Virginia Tech. While in graduate school, Perugini worked on personalization with Naren Ramakrishnan, his thesis advisor, and was funded by him as a research assistant. He also assisted the department in its teaching mission by serving as the teaching assistant and primary instructor for several courses. In May 2001, Perugini completed a Master of Science in Computer Science.

Perugini is a member of the ACM, IEEE, IEEE Computer Society, and ΥΠΕ honor society. His research interests include the application of concepts from programming languages and data mining to problems in interactive information retrieval and web modeling.

In June 2000, Perugini was recognized by the Washington, D.C. Chapter of the ACM with the Samuel N. Alexander ACM Fellowship. Upsilon Pi Epsilon and the IEEE Computer Society recognized him with an ΥΠΕ/IEEE-CS Award for Academic Excellence (one of only four such awards given internationally each year) in April 2001. In March 2004, Perugini was recognized by Virginia Tech as the College of Engineering Ph.D. Student Researcher of the Year, with the P. E. Torgersen Graduate Student Research Excellence Award (first-place winner). He credits his parents, grandparents, and thesis advisor for providing him with a blueprint for success which has ultimately shaped him into the person he is today.

In the Fall 2004, Perugini will join the faculty of the Department of Computer Science at the University of Dayton, a Marianist community in Ohio, as an Assistant Professor of Computer Science. New hopes for knowledge and science; and colleagues and friends are there and beyond. And therefore, as he sets sail, he asks God's blessing for the greatest adventure on which he has ever embarked.