# The Partial Evaluation Approach to Information Personalization

Naren Ramakrishnan and Saverio Perugini
Department of Computer Science
Virginia Tech, Blacksburg, VA 24061
Email:{naren,sperugin}@cs.vt.edu

August 8, 2001

**Abstract**

Information personalization refers to the automatic adjustment of information content, structure, and presentation tailored to an individual user. By reducing information overload and customizing information access, personalization systems have emerged as an important segment of the Internet economy. This paper presents a systematic modeling methodology — PIPE ('Personalization is Partial Evaluation') — for personalization. Personalization systems are designed and implemented in PIPE by modeling an information-seeking interaction in a programmatic representation. The representation supports the description of information-seeking activities as partial information and their subsequent realization by *partial evaluation*, a technique for specializing programs. We describe the modeling methodology at a conceptual level and outline representational choices. We present two application case studies that use PIPE for personalizing web sites and describe how PIPE suggests a novel evaluation criterion for information system designs. Finally, we mention several fundamental implications of adopting the PIPE model for personalization and when it is (and is not) applicable.

**Categories and Subject Descriptions:** F.3.2 [**Theory of Computation**] Semantics of Programming Languages–*Partial Evaluation.* H.4 [**Information Systems**] Information Systems Applications. H.5.2 [**Information Systems**] Information Interfaces and Presentation–*User Interfaces.* H.5.4 [**Information Systems**] Information Interfaces and Presentation–*Hypertext/Hypermedia.*

# Contents

# 1 Introduction

One of the main contributions of information systems research is the development of models that allow the specification and realization of information-seeking activities. Besides formalizing important operations, such models provide a vocabulary with which to reason about the information-seeking activity. For instance, if an information space is modeled as a term-document matrix, then the vector-space model permits the view of retrieval as measuring similarities between document vectors. Similarly, the modeling of data as a set of relations in a database system affords expressive query languages such as SQL. Other models and modeling methodologies can be found in interactive information retrieval applications [CKPT92, Sac00, Wil84]. Our goal in this paper is to present a modeling methodology for information personalization.

Personalization constitutes the mechanisms and technologies required to customize information access to the end-user. It can be defined as the automatic adjustment of information content, structure, and presentation tailored to an individual user. The reader will be familiar with instances of personalization such as web sites that welcome a returning user and recommender systems [AT01, RV97] at sites such as `amazon.com`. The scope of personalization today extends beyond web pages and web sites [THA99] to many different forms of information content and delivery [CDA00, KBM+00, MB00]. The underlying algorithms and techniques range from simple keyword matching of consumer profiles, to explicit [AWWY99, KMM+97, THA+97] or implicit [MCS00, Spi00] capture of user interaction.

Despite its apparent popularity in reducing information overload on the Internet, personalization suffers from a lack of any rigorous model or modeling methodology. One of the main reasons is that there are 'personal views of personalization [Rie00].' There are hence as many ways to design and build a personalization system as there are interpretations for what personalization means. Such a diversity presents a difficulty when studying conceptual models of personalization, in general.

We present the first (to the the best of our knowledge) systematic modeling methodology for information personalization. Termed PIPE ('Personalization is Partial Evaluation') [Ram00], our methodology makes no commitments to a particular algorithm, format for information resources, type of information-seeking activities or, more basically, the nature of personalization delivered. Instead, it emphasizes the modeling of an information space in a way where descriptions of information-seeking activities can be represented as partial information. Such partial information is then exploited (in the model) by *partial evaluation*, a technique popular in the programming languages community [Jon96].

While our ideas and results apply to many forms of computerized information systems (e.g., web-based, voice-activated), we restrict our attention to web sites in this paper. Later in our discussion, we qualify the range of information systems technologies to which PIPE can be applied.

### Reader's Guide

Section 2 introduces the basic concepts of PIPE with the example of personalizing a browsing hierarchy on the web. Section 3 outlines the PIPE modeling methodology and how it can be used for representing a variety of situations. Section 4 describes two application studies that use PIPE for personalizing web sites. Evaluation aspects implied by PIPE as a modeling methodology are also described here. Section 5 describes connections between PIPE and other approaches, and carefully qualifies situations where PIPE is (and is not) applicable. Finally, Section 6 summarizes the major contributions of this work.

## 2 Motivating Example

Consider a consumer visiting an automobile dealership to purchase a vehicle. Here are two possible scenarios.

*Scenario 1*

**Dealer:** Madam, are you looking to purchase a passenger vehicle?

**Buyer:** Yes.

**Dealer:** Do you have a particular manufacturer in mind?

**Buyer:** I know that cars made by Honda have the highest safety approval rating.

**Dealer:** That is true. Honda comes in seven colors. Do you have a preference for color?

**Buyer:** The 'cyclone blue' looks pleasing.

(conversation continues to ascertain further details of the vehicle)

*Scenario 2*

**Dealer:** Sir, may I interest you in anything?

**Buyer:** I am looking for a sport utility vehicle.

**Dealer:** Sure, do you have a particular manufacturer in mind?

**Buyer:** Not really, but the vehicle should be Red and made in 2001.

**Dealer:** I see.

**Buyer:** And by the way, I don't care for the fancy doormats and fittings.

**Dealer:** Of course.

(conversation continues)

In the first scenario, the conversation is directed by the dealer, and the buyer merely answers questions posed by the dealer. The second scenario resembles the first upto a point, after which the buyer takes the initiative and provides answers 'out of turn.' When queried about manufacturer, the buyer responds with information about color and year of manufacture instead. Nevertheless, the conversation is not stalled and both parties continue the dialog to (eventually) complete the information assessment task. At each stage in the above conversations, the buyer has the choice of proceeding along the lines of inquiry initiated by the dealer or can shift gears and address a different aspect of information assessment. Scenarios that 'mix' these two modes of inquiry in such arbitrary ways constitute the scope of *mixed-initiative* interaction [NS97].

Can we support a similar diversity of interaction in an online information system? In other words, the system should have a default mode of interaction where a user would fill in forms (or click on choices) in a specified order. A more enterprising user should be able to supply any piece of information out of turn. Finally, it should be possible to mix these two modes of interaction in any order. At each stage of the interaction (whether system-initiated or user-requested), the system should respond with the appropriate set of choices available. For instance, notice the restriction to seven colors once the decision on Honda is made in *Scenario 1*. If the choice of color was made at the outset, presumably more selections would have been available. A system that supports such a diversity of interaction would be personalized to a user's individual preference(s) for information-seeking.

The typical solution involves anticipating the forms of interactions that have to be supported and designing interfaces to support the implied scenarios (in this paper, we use the term 'scenarios' to mean scenarios of interaction). Fig. 1 describes four typical solutions that make various assumptions on the scenarios that will be supported. Fig. 1 (top left) can only support situations such as *Scenario 1* above, in that the user is forced to make a choice of manufacturer at the outset (and all remaining levels are similarly fixed). We refer to this as a design that hardwires scenarios. Fig. 1 (top right) also hardwires scenarios, but provides a choice of two such hardwired scenarios (i.e., search by model or search by price). Fig. 1 (bottom left) is what we refer to as *complete enumeration*, which involves

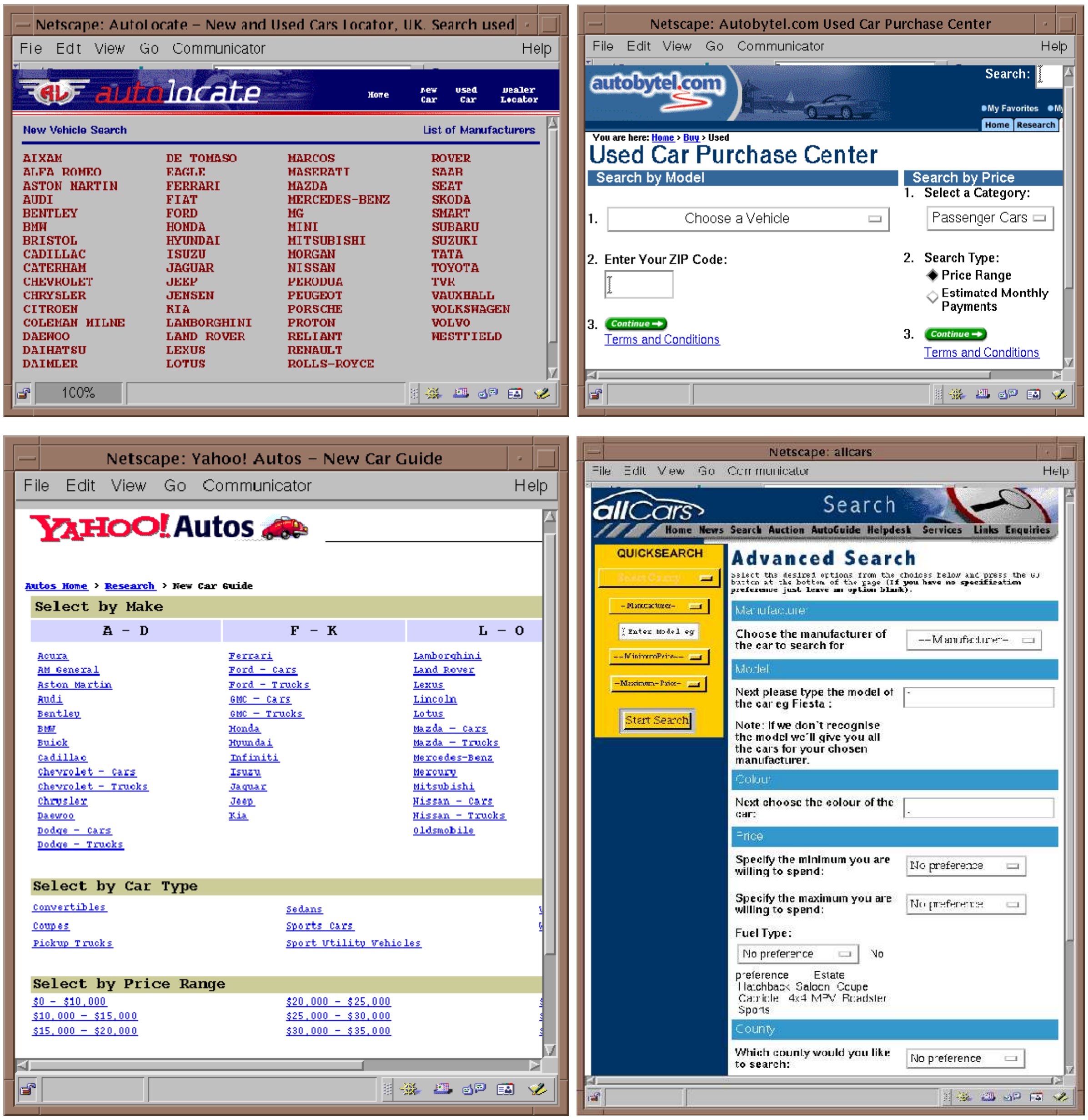


Figure 1: Four typical solutions to organizing web catalogs. (top left) A hardwired scenario. (top right) A choice of two hardwired scenarios. (bottom left) Complete enumeration involving all possible scenarios of interaction. (bottom right) A 'power-search' form that hides details of enumeration.

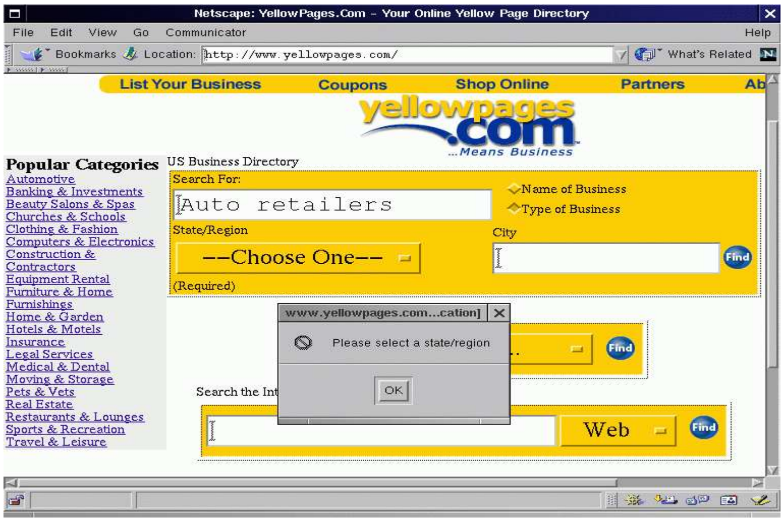


Figure 2: An interface that prohibits certain information-seeking activities from being decribed.

enumerating all possible scenarios and providing interfaces to all of them [Hea00]. While the interface in Fig. 1 (bottom left) only depicts the top-level choice, we could imagine that such multiplicity of choices are duplicated at all lower levels. It is clear that enumeration could involve an exponential number of possibilities and correspondingly cumbersome site designs. And finally, Fig. 1 (bottom right) provides the same functionality as Fig. 1 (bottom left) but masks the details of enumeration in a convenient 'power-search' form.

All of these solutions rely on anticipating the points where an out-of-turn interaction can occur and provide mechanisms to support it. When opportunities for out-of-turn interaction are too restrictive, information systems cause major frustrations to users. The basic problem is the representational mismatch between the user's mental model of the information-seeking activity and the facilities that are available for describing the activity.

In Fig. 2, the user is attempting to decide on an automotive retailer based on the services offered. He is open to the possibility of traveling to a different city in order to make his purchase. He is thus unsure of providing information about the location of the retailer, but the system insists that he make this choice first. The reader can identify with examples such as these from other personal experiences.

## 2.1 The PIPE Approach

We present an alternative design approach, one that promotes out-of-turn interaction without predefining the points where such interaction can take place. Consequently, the interfaces produced by our approach are, at once, both more expressive and simpler than the ones in Fig. 1.

| | |
|---|---|
| `int pow(int base, int exponent) {`<br>`  int prod = 1;`<br>`  for (int i=0;i<exponent;i++)`<br>`    prod = prod * base;`<br>`  return (prod);`<br>`}` | `int pow2(int base) {`<br>`  return (base * base)`<br>`}` |

Figure 3: Illustration of the partial evaluation technique. A general purpose `power` 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.

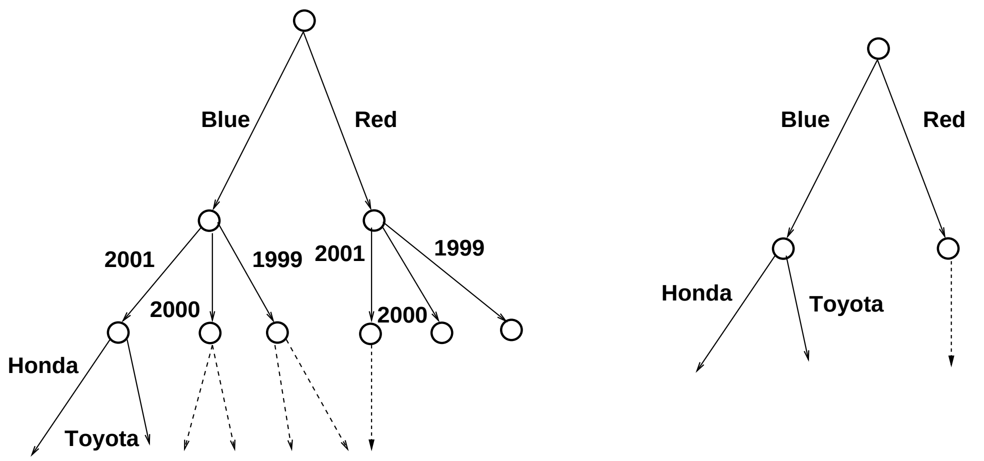


Figure 4: Personalizing a browsing hierarchy. (left) Original information resource. (right) Personalized hierarchy with respect to vehicles made in 2001. Notice that not only the pages, but also their structure is customized for (further browsing by) the user.

```
if (Blue)
 if (2001)
  if (Honda)
   .........
  else if (Toyota)
   .........
 else if (2000)
  .........
else if (Red)
 if (2001)
  .........
 else if (2000)
  .........
```

```
if (Blue)
 if (Honda)
  .........
 else if (Toyota)
  .........
else if(Red)
 .........
```

Figure 5: Using partial evaluation for personalization. (left) Programmatic input to partial evaluator, reflecting the organization of information in Fig. 4 (left). (right) Specialized program from the partial evaluator, used to create the personalized information space shown in Fig. 4 (right).

Let us begin by considering the scenario where a user obediently supplies information attributes in the order requested. For ease of presentation, we assume that there are three attributes — color, year of manufacture, and manufacturer — and that the information system ascertains values for them in this order. The key contribution of PIPE is to cast this seemingly inflexible and hardwired scenario in a representation that allows its automatic transformation into other scenarios. In particular, PIPE represents an information space as a program, partially evaluates the program with respect to (any) user input, and recreates a personalized information space from the specialized program.

The input to a partial evaluator is a program and (some) static information about its arguments. Its output is a specialized version of this program (typically in the same language), that uses the static information to 'pre-compile' as many operations as possible. A simple example is how the C function `pow` can be specialized to create a new function, say `pow2`, that computes the square of an integer. Consider for example, the definition of a `pow`er function shown in the left part of Fig. 3. 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. Thus, `pow2` is obtained automatically (not by a human programmer) from `pow` by precomputing all expressions that involve `exponent`, unfolding the for-loop, and by various other compiler transformations such as *copy propagation* and *forward substitution*. Automatic program specializers are available for C, FORTRAN, PROLOG, LISP, and several other important languages. The interested reader is referred to [Jon96] for a good introduction. While the traditional motivation for using partial evaluation is to achieve speedup and/or remove interpretation overhead [Jon96], it can also be viewed as a technique for simplifying program presentation, by removing inapplicable, unnecessary, and 'uninteresting' information (based on user criteria) from a program.

Consider the hardwired scenario depicted in Fig. 4 (left). We can abstract this hierarchy by the program in Fig. 5 (left) whose structure models the information resource (in this case, a hierarchy of web pages) and whose control-flow models the information-seeking activity within it (in this case, browsing through the hierarchy by making individual selections). The link labels are represented as program variables and semantic dependencies between links are captured by the mutually-exclusive `if..else` dichotomies. As it is modeled in Fig. 5 (left), the program reflects the assumption that the choice of year is usually made at the second level, after a color selection has been made. However, to personalize for the user who says '2001' at the outset, we partially evaluate the program with respect to the variable `2001` (setting it to one and all conflicting variables such as `2000` to zero). This produces the simplified program in Fig. 5 (right), which can be used to recreate web pages with personalized web content (shown in Fig. 4, right). The second level of the hierarchy is simplified, bringing the originally third level as the new second level. The user is able to provide the value of any deeply nested variable out of turn, thus achieving mixed-initiative interaction.

### 2.2 Some Preliminary Observations

Personalization systems are thus designed and implemented in PIPE by modeling an information-seeking interaction in a programmatic representation. The above example has been carefully constructed to highlight the many advantages and opportunities provided by PIPE. Before we describe PIPE in detail, it will be helpful to summarize the lessons from the above example.

1. *PIPE equates personalization to specializing representations.* As a methodology, PIPE asserts that if interaction in an information space can be represented as a program, then a personalized information space can be automatically generated by partial evaluation. It is upto the designer to supply the representation as a program and reinterpret the program in information systems terms. The meaning of the programmatic representation is thus external to the basis for personalization (partial evaluation).

   For instance, the act of clicking on the 2001 hyperlink is captured in Fig. 5 by just the expression `if (2001)`. The conditional construct `if` is thus used as a logical point where the state of information is tested before proceeding any further. By partial evaluation, we are able to simplify (actually remove) the test `if (2001)` since, by declaring 2001 as his interests, the user has performed the functional equivalent of clicking on the 2001 hyperlink.

2. *The effectiveness of PIPE depends on what is modeled (and how).* The effectiveness of a PIPE implementation depends on the the particular modeling choices made *within* the programmatic representation (akin to [Wil84]). We cannot overemphasize this aspect — the example in Fig. 5 can be made 'more personalized' by conducting a more sophisticated modeling of the underlying domain. For instance, information such as vehicle VIN numbers, history of ownership, mileage on the vehicle, and photos of the car can be further modeled as a browsable hierarchy and 'attached' (functionally invoked) at various places in the program of Fig. 5 (left). Conversely the example in Fig. 5 (left) can be made 'less personalized' by, for instance, requiring categorical information along with user input. Replacing `if (2001)` in Fig. 5 (left) with `if (Year=2001)` implies that the specification of the type of input (namely that '2001' refers to the year of manufacture) is required

in order for the statement to be partially evaluated. Personalization systems built with PIPE can thus be distinguished by what they model and the forms of customization enabled by applying partial evaluation to such a modeling.

Similarly, the way in which program variables are associated with user input can influence the effectiveness of a PIPE implementation. Values for program variables could come from a content-based technique or a so-called collaborative technique. For instance, the variable `Honda` could be set to true, either because the user explicitly said so, or because 'Honda' was recommended to the user by an automatic recommender system. In addition, different variables could afford different interpretations.

Sometimes we can take advantage of a domain semantics when associating values with program variables or in modeling the program. Fig. 5 models a 'strict' semantics of variable assignment by the `if..else` dichotomies. If `Blue` is evaluated to true, then every other option qualified by the `else` constructs (such as `Red`) would be automatically removed from further consideration. This is due to our assumption that if the user declares 'Blue' as his preference, then he would not be interested in Red cars. If such a semantics is not appropriate, then we would not have `else` clauses in our conditionals. Thus, PIPE doesn't dictate what the domain semantics (for assigning program variables) should be or even that it should be available. But it can take advantage of a domain semantics, if one exists.

Finally, the translation of the program from and back to the information space could be done in different ways. In Fig. 5 (left) we modeled the program by abstracting hyperlinks across pages as conditionals. When we recreate personalized pages from Fig. 5 (right) we are not obliged to this design choice. We could cascade all the interactions to within a single page, for instance. PIPE only requires that the designer of the information system has a way of going from an information space to a programmatic representation, and back again. Section 3 covers modeling options in detail.

3. *PIPE separates modeling for a personalization system from the operational aspect of personalization.* Personalization systems are usually described in terms of the techniques that provide personalization or the level at which the information is tailored. Due to the variety possible, comparisons of personalization systems have been difficult to make. PIPE, on the other hand, shifts the focus to modeling for a personalization system. Any form of personalization is possible if the modeled program allows the pertinent scenarios to be expressible as partial inputs. In Fig. 5 we cannot personalize cars with respect to occupancy, not because of any funda-

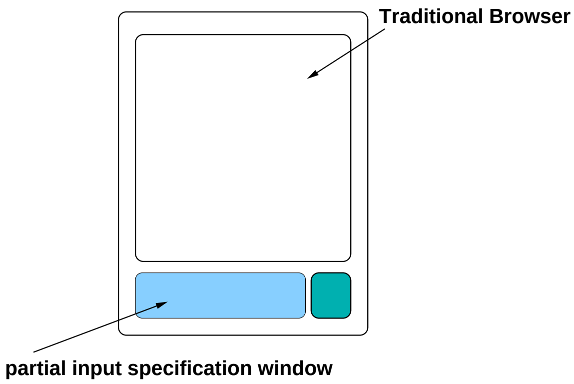


Figure 6: Sketch of a PIPE interface to a traditional browser. The interface retains the existing browsing functionality at all times. At any point in the interaction, in addition, the user has the option of supplying personalization parameters and conducting personalization (bottom two windows). Such an interface can be implemented as a toolbar option in existing systems.

mental limitation in our personalization methodology, but because `occupancy` is not available as a program variable. Similarly, we cannot personalize cars with respect to the *Edmund's Car Guide* recommendations, because the latter information resource has not been modeled. Only the site graph was modeled and hence only personalizations pertaining to manipulating the site graph are possible.

The separation of modeling from the operational aspect of conducting personalization means that we can devote our attention to modeling the interaction in as sophisticated a manner as required. It also means that we have to distinguish between evaluating an implementation of the PIPE methodology from an evaluation of the methodology itself.

4. *The PIPE personalization operator is closed.* Since the partial evaluation of a program results in another program, the PIPE personalization operator is closed. In terms of interaction, this means that any modes of information-seeking (such as browsing, in Fig. 5) 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. Executing the program in the order and form in which it was modeled amounts to the system-initiated mode of 'browse as I say.' 'Jumping ahead' to nested program segments by partially evaluating the program amounts to the user-directed mode of personalization. In Fig. 5, the simplified program can be browsed in the traditional sense, or partially evaluated further with additional user inputs. PIPE's use of

```
if (Returning Customer)
 /* be nice to her */
else
 /* just show usual catalog */
```

Figure 7: A modeling of an information space that involves only one level of interaction.

partial evaluation is thus central to realizing a mixed-initiative mode of information-seeking, without explicitly hardwiring all possible scenarios of interaction (including out-of-turn interactions). A sketch of an interface design for such mixed-initiative interaction is provided in Fig. 6.

5. *PIPE is most advantageous in information spaces that afford nested representations of interactions and where information-seeking activities can involve out-of-turn interactions.* For browsing hierarchies, a nested programmatic model can be trivially built by a depth-first crawl of the site (as in Fig. 5). Not only is this modeling appropriate, it is also concise and makes the advantages of partial evaluation obvious.

   On the other hand, consider a web site that determines (perhaps by a cookie [Ber01]) if a user is a returning customer and does something different based on this information. Modeling (only this) interaction can be done by the program in Fig. 7. While partial evaluation is still applicable, it cannot do anything fancy since there is only one variable (`Returning Customer`) to specify values for. There is no deeply nested variable whose value can be supplied out of turn.

   Similarly, if all users would like to browse through the catalog in Fig. 5 by a color-year-model motif, then there is really only one way in which the catalog is being used. This usage mirrors the way in which the catalog is modeled, without any out-of-turn interactions. Partial evaluation is thus not necessary to support the information-seeking goals of any user.

   The presence of out-of-turn interactions implies different rates of specification for different aspects of information seeking, causing a rich variety of possible interactions. In such a case, PIPE can be viewed as a technique that realizes a particular interaction sequence by combinations of simplification and normal execution. In Section 5.2, we show more formally which representations (and which information spaces) are best suited for personalization by partial evaluation.

# 3 Essential Aspects of PIPE

We now describe the PIPE methodology in more detail and outline choices available for modeling typical situations. While partial evaluation permits formal specification with mathematical notation [Jon97], we do not take this approach here. Instead, for the *ACM TOIS* audience, we aim to emphasize the larger context in which partial evaluation is used in PIPE and describe its advantages for information systems. We intend to present the formal aspects of the PIPE methodology in a second paper.

## 3.1 Modeling Methodology

As a modeling methodology, PIPE only makes the weak assumption that information is organized along a motif of interaction sequences. For our purposes, an interaction sequence is a list of primitive inputs used to describe the information-seeking activity. For instance in Fig. 5, information about vehicles is organized along a color-year-model motif with the primitive inputs corresponding to specific choices of color, year, or model. The interaction sequence in this example involves the the choice of `2001` for `year`, in support of the user's goals.

Information is embodied in an interaction sequence in two forms — *structural* and *terminal*. Structural information is what helps us refer to an interaction sequence; it is explicitly represented in PIPE and specified via program variables. In Fig. 5, the structural information corresponds to choices of color, year, and model. This form of information thus captures the partial information supplied by the user by instantiating parts of the motif. When the user specifies '2001' in Fig. 5, the `year` part of the motif is turned on and set to this value.

Terminal information is also represented in PIPE, but is not directly manipulatable or even directly addressable. Programs in PIPE are not explicitly parameterized by this information and so the user cannot specify personalization in these terms. In Fig. 5, terminal information corresponds to the leaves, which would be information about particular vehicles. In a different application, terminal information could reside at every step in the interaction sequence.

Structural information provides the 'backbone' that strings together terminal information. However, it is important to note that structural information is considered first-class information in PIPE and not merely 'features' with which we index the 'real information' (although it is tempting to view it this way). To see why, observe that partial evaluation does not provide a mapping from structural to terminal information (unless it was a complete evaluation specifying all program variables). After a partial evaluation (e.g., Fig. 5 (right)) the specialized program might still contain structural information. This does not necessarily mean that the user's information-seeking activity is incomplete. The residual structural information contributes to the programmatic modeling of interaction, *which is* the personalized information space in PIPE. Another way to see this is to note that PIPE simplifies *interaction* with an information space. Thus interaction can be seen to be the determiner of information (both structural and terminal). The view of structural information as first-class information is also natural if we think of the program in logic programming terms, rather than imperative programming.

Since information can be organized all along the interaction sequence, in both structural and terminal forms, we need a way to define the state of information described by the sequence as a whole. It is useful to assume a 'combining function' for defining the state of information at the end of the sequence. A simple example of a combining function is the additive operator which mirrors the accumulation of information by following an interaction sequence. In Fig. 5, if the color and model parts of the motif are turned on, then the state of information known about that sequence is a set of values for $\{\texttt{color,model}\}$. Another example is to just retain information from the most recent step(s) in the sequence. This would be appropriate when information-seeking has an exploratory nature to it and we wish to discount some earlier steps in an interaction sequence as being 'tentative' (the applications presented in this paper do not have this flavor). Combining functions for terminal information can be defined similarly.

Since PIPE only emphasizes the design and implementation of personalization systems, it doesn't pay any attention to how the interaction sequences are obtained and how the choice between terminal and structural parts is made. In particular, PIPE is not a complete lifecycle model for personalization system design and doesn't address issues such as requirements gathering. Interaction sequences could come from explaining users' behavior [RRC01, WM99], by identifying all possible paths through a given site, or from our conceptual understanding of the information-seeking activity. They also depend on the targeting goals of the personalization system. In [RRC01], we have presented a systematic methodology for obtaining interaction sequences and identifying structural and terminal parts, by 'operationalizing' scenarios of interaction; we refer the reader to this reference for details. In this paper, we assume that they are available and proceed to further characterize and represent them.

**Characterizing Interaction Sequences**

Information seekers forage in different ways [Pir01] and the existing design of the information system also influences their interaction sequences. An important aspect of an interaction sequence is its length, which affects its subsequent representation in PIPE.

In many applications, interaction sequences are bounded. For instance, in Fig. 5 an interaction sequence of length at most 3 describes the information-seeking activity. Such sites and applications are characterized by their support for a goal-oriented, opportunistic view of information-seeking. Hierarchies, recommender systems, and scrolling to a specific location on a page are examples. In general, any information-seeking activity that has clear start and end states and which relies on perceptual, display-driven clues that focus attention can be represented as a bounded sequence.

In other important cases, interaction sequences can be unbounded. The trivial example is when we allow the possibility that a user may click ‘back buttons.’ If we undo these steps before representation, we can proceed as if they never happened. Alternatively, we can model back buttons using a finite-state machine (FSM), but we have to find a characterization of applications where modeling at this level of detail would be useful. A more interesting example of unbounded sequences involves browsing at a site based on social network navigation, such as `www.imdb.com`. There are no leaves in this site and the site graph resembles a social network. Users are encouraged to systematically explore relationships between actors, movies, and directors by ‘jumping connections.’ Such a site is characterized by an exploratory nature of information-seeking, akin to data mining. Goals are articulated less clearly and cognitive knowledge is used from various resources to decide on how to conduct information-seeking. In fact, there is no distinction between structural and terminal information in this site! Any particular web page could be used to address other items or thought of as the result of an information-seeking activity.

Both bounded and unbounded interaction sequences can be described using constructs such as regular expressions, grammars, FSMs, and programs; unbounded interaction sequences require special handling, due to the reasons mentioned above. In this paper, we concentrate on personalization applications describable by bounded interaction sequences and which have a clear separation between structural and terminal parts.

**Representing Interaction Sequences in PIPE**

Given that we can represent information-seeking activities as interaction sequences, the set of scenarios that are likely to be encountered (over all users, perhaps) can be represented by a corresponding set of interaction sequences. Once again, PIPE doesn’t indicate what the set of interaction sequences should be: whether it is defined across all users [WM99], whether it is for a group of users [PE00], or whether it comes from our conceptual understanding of information-seeking. PIPE only emphasizes the representation of interaction sequences in a programmatic representation.

For instance, Fig. 5 uses a nested representation to form the program for subsequent partial evaluation. Not only does it model the color-year-model motif (as it would have been observed), it also allows us to model the year-color-model motif (by one partial evaluation). Since PIPE provides out-of-turn interaction, it is not necessary to represent every interaction sequence explicitly in the program.

Such compaction of interaction sequences is important for two reasons. The first is that it preserves the inherent structure of the (unpersonalized) information-seeking activity (such as browsing, in Fig. 5). This is useful in realizing mixed-initiative interaction with PIPE. Another reason is that compaction permits scalable personalization solutions.

Structural parts of interaction sequences can be represented using constructs in a full-fledged programming language, such as C (as done in Fig. 5) or LISP. A programming language provides many facilities that can help in compaction of interaction sequences. For example, if we notice that all interaction sequences at a site require registration at some point in the interaction, then the steps associated with registration could be factored out and procedurally invoked from various other locations. Off-the-shelf partial evaluators (such as C-Mix) can then be used for specializing the representations.

It is important that we also model terminal parts of interaction sequences. In the example of Fig. 5, if there is text anchoring every hyperlink, then we can define a program variable to start accumulating text once every conditional is evaluated to be true. This could be achieved using associate arrays or by dynamic memory allocation constructs (e.g., pointers). After partial evaluation, we can inspect the contents of this data structure at every stage to present personalized (terminal) content. Inspecting the contents of the sequence as a whole will provide an overall summary of the terminal information. Inspecting the contents of subsequences will provide more fine-grain summaries of terminal information.

**Creating a Personalization System**

To effect the creation of a personalization system, we define ways for the user to specify values for program variables and a procedure by which personalized information content is presented back to the user. Every construct used in the programmatic modeling (terminal or structural) should be translatable into information systems terms, and vice versa.

Typically, there is a one-one mapping between interactions and programming constructs. In Fig. 8, the textbox corresponds to a conditional, the listbox to a `switch` construct, and the unit convertor to a function in a PIPE modeling.

Such mappings have to be revisited after partial evaluation. For instance, the `if` construct in Fig. 8 will either

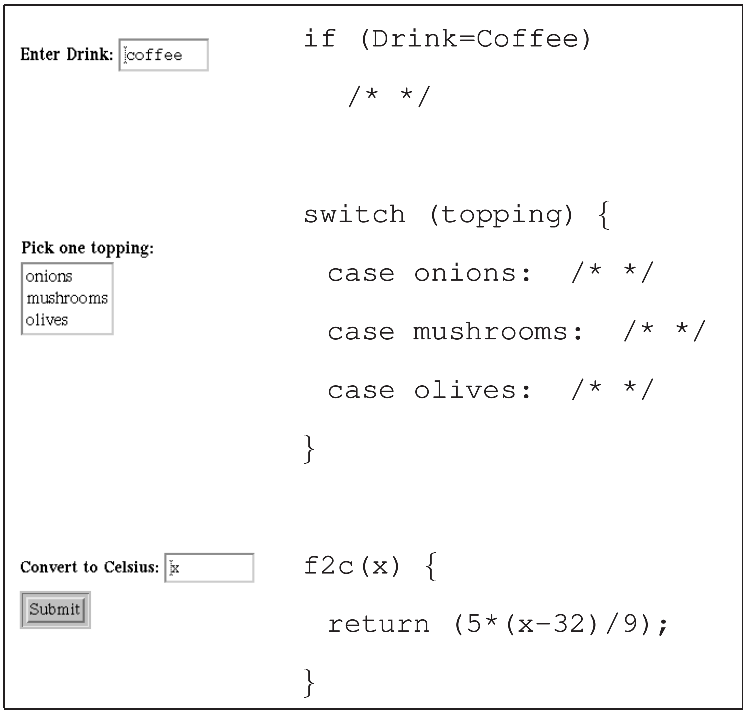


Figure 8: Choices for representing aspects of interaction in PIPE.

be removed or left as-is by a partial evaluation. This will just correspond to removing or retaining the textbox in the personalized web site. The `switch` construct in Fig. 8 corresponding to a listbox is more interesting. After partial evaluation, it might be the case that only one of the three topping options are left. Perhaps the person is allergic to mushrooms and olives and we set those variables to zero. In this case, the partial evaluator might remove the `switch` altogether and replace it with a simple `if`. We can view this as a hint to render the listbox as a hyperlink in the personalized site. Finally, the unit conversion utility in Fig. 8 can be modeled in several ways. We can view it as a functional black-box and model in PIPE the act of getting a value and passing it to, say, a server-side script that performs the conversion. If we take this approach, we should ensure that partial evaluation either retains the black-box representation or removes it; it shouldn't 'open' it up. Alternatively, we can explicitly open up this black-box and model its contents as a function in a PIPE modeling (as done in Fig. 8). As a functional modeling, PIPE thus enables the view of information systems as transducers.

In some cases partial evaluators, by their sophisticated support for program specialization, cause difficulties. For instance, the technique of *program-point specialization* [Jon96] introduces copies of functions at various places in the specialized program, tailored to specific situations. In information systems terms, this amounts to creating content (structural as well as terminal) that didn't exist before. In such a case, we need to carefully interpret the meaning of the specialized representation.

Another caveat is that partial evaluation can sometimes induce `gotos` in the specialized program. We can view `gotos` as suggesting means by which the site design could be structured. If there is a `goto` from a point $A$ in the program to another point $B$, it just means that the information system corresponding to point $B$ can be arrived at in many ways via interaction sequences and hence is advantageous if factored out.

Finally, a semantics of values for program variables has to be defined. In partial evaluation, values may be either specified or left unspecified. By default, variable values cannot be weighted unless explicitly modeled in the PIPE program. However, techniques such as query expansion can be employed to obtain values for other program variables. For instance, if a user says 'Honda' and a PIPE program models Honda cars under 'Japanese automakers,' then we can turn both these variables on for the purposes of personalization. Semantics for program variables can also be defined to take advantage of other taxonomical relationships in hierarchies [Res99].

#### A Salient Feature of PIPE

An important advantage of PIPE is that while we provide options for modeling, there is is no explicit step for describing how to implement personalization. Due to the sophistication of our representation, personalization will be achieved if program variables (which correspond to structural information) are available for partial evaluation. This is in contrast to other modeling methodologies [dOTM01, FP00, RS97] where personalization has to be provided as an explicit function from the conceptual design stage.

### 3.2 Representational Choices

Our primary example of modeling thus far addressed navigation down a hierarchy via nested conditionals (see Fig. 5). This is one of the most common sources of bounded sequences; it can be obtained either by explicit crawling or as graph representations of site structure from website management tools [ALW99, FFK+98]. In the former, extra care should be used to address purely navigational links (like a 'Go Back' button) and irregularities in web page authoring. Representations obtained from the latter case are more robust since they directly enable the modeling of interaction sequences in terms of directed labeled graphs [ABS00] or web schema [FP00].

In this section, we present a number of other modeling options for personalization applications described by bounded interaction sequences.

### Interacting with Recommender Systems

A recommender system can be viewed in PIPE as a way to set values for program variables or as a function to be modeled. In the first case, the recommender is abstracted as a black-box and is external to the program. Consider a recommender system at a third-party site that suggests automobile dealers based on experiences of its users. In such a case, we can invoke the facility to obtain values for program variables which are then subsequently used for personalization. Alternatively, the functioning of the recommender can be explicitly modeled in PIPE. This allows the possibility that even its operation could be personalized. For instance, if the recommender system can suggest dealers all across the United States, we can personalize its operation to only recommend dealers in a particular geographical region. This will not be possible in the black-box modeling unless the recommender allows such explicit specification.

### Information Integration

Effective personalization scenarios require the integration of information from multiple sites. Consider personalizing stock quotes for potential investors. The Yahoo! Finance Cross-Index at `quote.yahoo.com` provides a ticker symbol lookup for stock charts, financial statistics, and links to company profiles. It is easy to model and personalize this site by the methods described above. However, consider the user who desires to browse through this site based on recommendations from an online brokerage. Besides supporting the cascading of information flows, we have to ensure that structural information across multiple sites is correctly cross-referenced. The online brokerage might refer to its recommendations by company name (e.g., 'Microsoft'), while the Yahoo! cross-index uses the ticker symbol ('MSFT'). Information integration solutions based on wrappers [Kus00] and mediators can be employed here [FLM98, KMA$^+$98]. In PIPE, the individual interaction sequences from multiple sites can be cascaded to provide support for such integration scenarios, as shown in Fig. 9.

### Modeling Clickable Maps

Many web sites provide clickable image maps (e.g., JAVA/GIF) as interfaces to information. This is especially true for weather sites, bioinformatics resources, and sites that involve modeling spatial information. Interpretation is attached to clicking on particular locations of the map (for instance, 'click on the state for which you would like the weather'). Using data mining techniques [FMMT99] and by sampling clicks on the map (and determining which

```
main() {
 /* invoke online brokerage */
 /* transforms from company name to ticker symbol */
 /* modeling of yahoo cross-index */
}
```

Figure 9: Modeling information integration in PIPE.

pages they lead to), we can functionally model a clickable map in PIPE to arrive at constructs such as: 'Choosing Wyoming on the United States map corresponds to clicking within $[a, b] \times [c, d]$.' Non-rectangular areas are described by unions of isothetic regions by the data-mining technique described in [FMMT99]. Given such a representation, partial evaluation can remove portions of the image map based on user preferences. At this stage, we can reconstruct a personalized clickable map by reversing the mapping or use attributes such as color and shade to highlight the selected regions (for instance, to show only those regions on the map where air travel is delayed). We can also represent the personalized information in non-graphical terms. This option is useful not just for personalization but for improving the accessibility of information systems. A mobile handheld device incapable of presenting graphical content can take advantage of such modeling.

**Modeling within a Page**

In some cases, it is necessary to model interaction sequences within a web page. For instance, if a user is eyeballing a web page to look for telephone numbers of an individual, then modeling the web page at this level of granularity and providing a program variable for telephone number would be useful. Algorithms for mining structure within a web page (e.g., DTDs) [AK97, CDF$^{+}$00, GGR$^{+}$00] and for document segmentation [RS97] can be used to arrive at compact representations of within-page interaction sequences. This provides a richer set of features with which to conduct personalization. For instance, partial evaluation can be used to remove complete sections of documents (e.g., intrusive advertisement banners) when rendering the personalization.

**Program Compaction**

The naive rendition of a PIPE model by the above mechanisms might result in lengthy programs, with duplications of interaction sequences. Techniques for program compaction are hence important. This topic has been studied

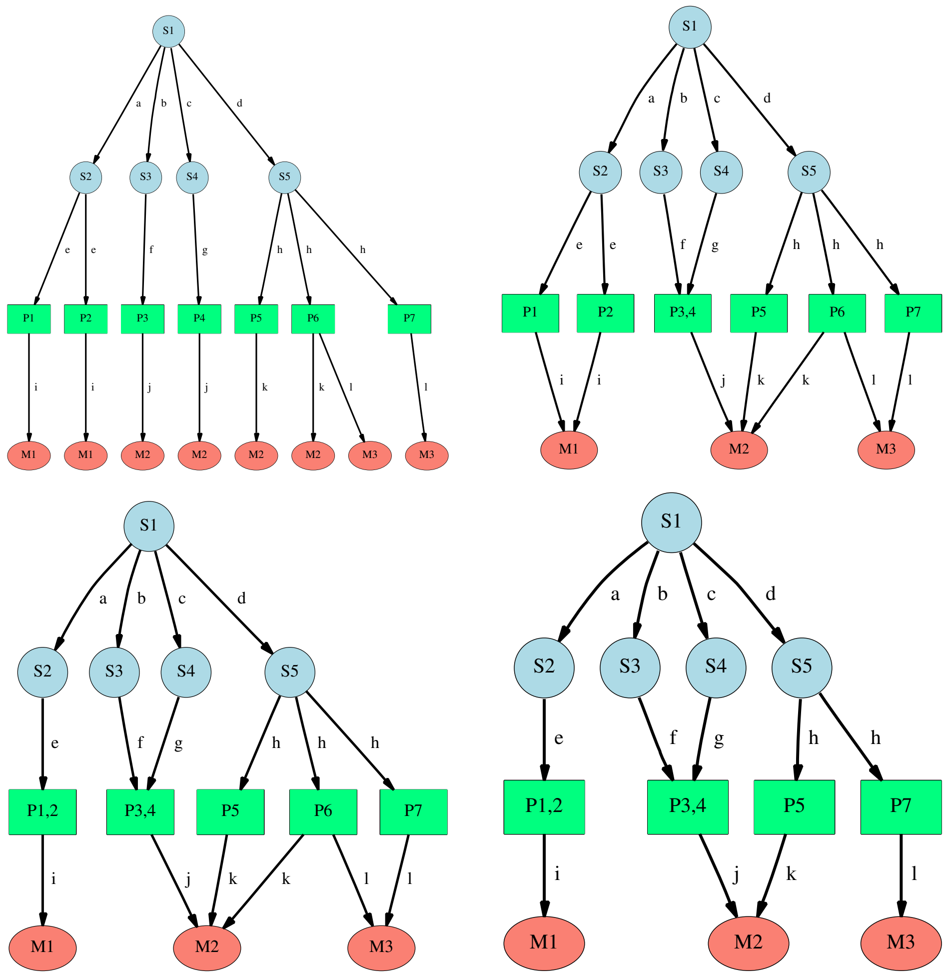


Figure 10: Four stages in extracting structure from a semistructured data source, by the algorithm of [NAM98]. (top left) Original semistructured resource with labeled and directed edges modeling interaction sequences. (top right) Factorization of commonalities encountered in crawling. (bottom left) A 'minimal perfect typing' of the data. (bottom right) Final output of data mining algorithm, after modeling 'multiple roles' [NAM98].

extensively in the data mining and semistructured modeling communities [ABS00, NAM98, WL00]. Of particular relevance to PIPE is the algorithm of Nestorov et al. [NAM98] whose modeling of semistructure closely resembles our representation of an interaction sequence in terms of program variables. This algorithm works by identifying graph constructs that could be factored, simplified, or approximated. Fig. 10 describes four stages in a procedure for program compaction. The starting point is the schema in Fig. 10 (top left) obtained by a naive crawl of a site. Fig. 10 (top right) factors commonalities encountered in crawling. There are only three leaf nodes and the internal nodes `P3` and `P4` are collapsed because they are really the same page. Fig. 10 (bottom left) is a 'minimal perfect typing [NAM98]' of the data, which means that the fewest internal nodes needed to describe the schema are used. In this example, `P1` and `P2` are collapsed, not because they are the same but because they exhibit the same schema. Both have an incoming edge labeled `e` from the same type of page (`S2`) and display an outgoing edge labeled `i` to the same type of page (`M1`). While their contents may not be the same, interaction sequences involving them can be compacted. Care must be taken to ensure that any accompanying text with these nodes are not lost. And finally, Fig. 10 (bottom right) casts `P6` as redundant for the purpose of modeling interaction sequences. The role of `P6` in Fig. 10 (bottom right) is to establish connections from `S5` to `M2` and `M3`, which are already embodied in `P5` and `P7` respectively. Thus, we can remove `P6`, once again after ensuring that any contents of that node are suitably represented elsewhere. In [NAM98], `P6` is referred to as a node that exhibits 'multiple roles.'

**Miscellaneous Optimizations**

Finally, the success of a personalization system relies on those finer touches that deliver a compelling experience to the user. Options in this category are ad-hoc by nature and are not technically modeling choices since they involve post-processing of the specialized program. For instance, assume that we personalize the automobile example in Fig. 5 with respect to the variables `Honda` and `2001`. This might produce a construct such as:

```
if (Green) {
        /* two empty code blocks */

        /* the first is empty because Honda and 2001 evaluated to true,
           but there were no green Honda cars made in 2001 */

        /* the second is empty because other models and other years were set
```

```
            to be evaluated to false */
}
```

While semantically correct, such code blocks are useless for information presentation. They can be perceived as dead-ends and safely omitted during web page reconstruction. It would also be confusing to the user who clicks on 'Green' and receives nothing (or an empty page) in return!

A second form of optimization arises when partial evaluation results in a nested conditional with no `else` clauses:

```
if (Blue) {
   if (2001) {
      if (Honda) {
          /* something here */
      }
   }
}
/* nothing here */
```

In such a case, we need to pay attention to how the simplified program is presented back to the user. Forcing the user to continue clicking on items when there is only one choice at every level is undesirable. Rather, we could just reveal to the user that according to his personalization criteria, the only type of cars remaining are 'Blue Honda 2001' and directly link to the items of information. This example reinforces our idea that structural information is first-class information. We are working on a customized partial evaluator that can perform such optimizations.

## 4 Application Case Studies

We now describe two applications that use PIPE to personalize collections of web sites. They are presented in increasing order of complexity, as evidenced by the forms of modeling they conduct (Table 1). In each of these applications, we state the conceptual model of interaction sequences and the specific choices made in modeling. Evaluation methodologies are outlined after the descriptions. Since PIPE only specializes representations, we are able to personalize even third-party sites by forming suitable representations. More personalization systems designed with PIPE are described in [Ram00, RRC01]; we present only two here for space considerations.

| | |
|---|---|
| Congressional Officials | Modeling Site Structure |
| | Modeling within a Page |
| | |
| Mathematical and Scientific Software | Modeling Site Structure |
| | Interacting with Recommender Systems |
| | Information Integration |
| | Modeling within a Page |
| | Program Compaction |

Table 1: Modeling options used in the application case studies.

### 4.1 Congressional Officials

Our first application customizes access to the Project Vote Smart website (`http://www.vote-smart.org`), an independent resource for information about United States governmental officials. The site caters to people interested in politicians' backgrounds, committee memberships, and positions on major political issues. While Project Vote Smart reports on state and local governments as well as the federal government, we focused only on the congressional subsection of the site in our experiments.

The conceptual model of information-seeking involves browsing through the congressional subsection to retrieve individual web pages of politicians. Interaction sequences at this site consist of choices of state (e.g., California, Virginia, etc.), branch of congress (House or Senate), party (Democrat, Republican, or Independent), and district information (numbers of districts). The terminal information involved 540 home pages (for 100 Senate members and 440 House members) and resides at the ends of interaction sequences.

Fig. 11 describes a typical interaction sequence. At the root congressional page (Fig. 11 (top)), users are directed to select a state of interest. Selection of state transfers the user to that particular state's web page (Fig. 11 (bottom left)). A state web page is semistructured, listing both senators and representatives as well as their party, district affiliations, and other associated information. Finally, a user arrives at a politician's webpage (Fig. 11 (bottom right)) by making a selection at the state page. Thus, the congressional section of Project Vote Smart is three levels deep (with a two-step interaction sequence).

Since many of the choices made by the user in browsing through Project Vote Smart are independent of each


Figure 11: A typical interaction sequence at the Project Vote Smart web site. (top) Start page for congressional officials. Making a selection of state at this level reaches a state-level page (bottom left). Finally, individual politicians' web pages are accessed by making selections at the state-level page (bottom right).

other (e.g., selecting Virginia as state does not imply a particular political party), the site is highly amenable to personalization by partial evaluation. Currently the site hardwires interaction sequences in the order shown in Fig. 11. We modeled the two-step interaction sequence (as shown in Fig. 11) as actually a four-step interaction sequence by conducting a more detailed modeling of the state-level page. In particular, the semistructure on state-level pages was abstracted to yield independently addressable information about branch of congress, party, and district.

The site graph is not a balanced tree. For instance, every state has exactly two senators but the number of representatives varies from 1 in South Dakota to 52 in California (this is dependent on state population). Our modeling of data at state pages expanded the original 3-level tree shown in Fig. 11 consisting of 596 nodes (1 root page + 55 state pages + the previously mentioned 540 leaves of the tree) to 5 levels comprising 857 nodes (317 internal nodes + 540 leaf nodes). This amounts to a approximately 44% percent explosion in the site schema.

The programmatic representation of the new site schema was in C and it captured miscellaneous domain semantics about interaction at the site (e.g., if the user says 'District 21,' he is referring to a Representative, not a Senator). The partial evaluator C-Mix was used for this study.

### 4.2 Mathematical and Scientific Software

Our second application is a personalization system for recommending mathematical software on the web for scientists and engineers. Consider a scientist studying stress in a helical spring; he formulates the problem mathematically in terms of a partial differential equation (PDE) and proceeds to find software that can help in solving his PDE. He uses a collection of three web sites to conduct his information-seeking activity.

First, he accesses the GAMS (Guide to Available Mathematical Software) cross-index of mathematical software (`http://gams.nist.gov`), a tree-structured taxonomy that covers nearly 10,000 algorithms (from over 100 software packages) for most areas of scientific software. GAMS functions in an interactive fashion, guiding the user from the top of a classification tree to specific modules as the user describes his problem in increasing detail. During this process, many important features of the software (e.g., 'are you looking for a software to solve elliptic problems?') are determined, from the user. However at the ends of the interaction sequences at GAMS, there still exist several choices of algorithms for a specific problem. Now, the scientist consults a recommender system or a performance database server (for his category of scientific software) to pick an appropriate algorithm for his problem. An example is the PYTHIA recommender system for selecting solvers for PDEs [HCR+00]. At this point, the scientist supplies additional information to the recommender such as his performance constraints (on the time to solve his PDE). Systems like PYTHIA use previously archived performance data to arrive at recommendations such as ‘Use the second-order 9-point finite differences code from the ELLPACK module.’ After such a recommendation, the scientist conducts the final step of downloading the recommended software module from repositories such as Netlib (`http://www.netlib.org`) housed at the Oak Ridge National Laboratory (ORNL) or other packages at the National Institute of Standards and Technology (NIST). The conceptual model involved the information flow from the GAMS site, to a repository such as Netlib, through a recommender such as PYTHIA.

The choices made in GAMS will affect the choice of recommender which in turn affect the choice of repository. This application thus presents an interesting information flow for modeling. Since PIPE permits partial instantiation of the information flow, the scientist can directly access a repository such as Netlib if he is sure of the specific software he needs.

We modeled the entire GAMS web site, used the PYTHIA recommender (that addresses software for the domain of PDEs), and established connections with individual software modules at the various repositories. After an initial expansion of GAMS (e.g., by within-page modeling), we applied the program compaction algorithm described in Section 3.2. Cross-references in GAMS and duplication of common module sets (which are now revealed by our initial expansion) helped compress the site schema to 60% of its original size. In particular, the GAMS subtree relevant to describing PDEs provided for a 11% compression. There was no terminal information alongside intermediate nodes, and hence there was no need for any special handling. PYTHIA’s details are described in [HCR+00] and we conducted a white-box modeling in PIPE to better associate program variables from GAMS with variables in PYTHIA (one of the authors of this paper was also the co-designer of the PYTHIA recommender). Finally, the step to reach individual software modules was a simple one-step interaction sequence leading to terminal information about the code (in FORTRAN) and its documentation. The entire composite program was represented in the CLIPS programming language [Gia98] and we employed its rule-based interface for partial evaluation. More modeling details on this case study can be found in [PLR00].

### 4.3 Evaluation

We now describe procedures for evaluation. There are three possible types of evaluation:

1. Evaluating PIPE applications

2. Evaluating our modeling of information-seeking activities in PIPE

3. Evaluating PIPE

The first type of evaluation is what is usually described in the literature and there are many ways of conducting it. The accepted practice is to measure improvements in revenues, site visits, and user satisfaction (e.g., via surveys). In [Ram00], we have described the evaluation of PIPE applications using traditional user interviews followed by statistical validation (they have yielded good results). Commercial ventures such as *NetPerceptions* emphasize the scalability and speed-of-response of personalization systems. The second and third types of evaluation criteria highlight the role of PIPE as a modeling methodology. We concentrate on them since we have already described traditional user-response evaluation of PIPE applications in [Ram00]. This section covers the evaluation of modeling and Section 5.2 helps identify shortcomings of the PIPE methodology itself.

We evaluate a PIPE modeling by the extent to which it allows users' information-seeking activities to be described as partial inputs. This is in keeping with the view that PIPE's services are only as good as the modeling conducted in it. If a faulty recommender system is modeled in PIPE, then no amount of partial evaluation can provide satisfactory results.

Recall that our modeling was conducted with respect to a set of interaction sequences. For evaluation purposes, we identified an independent 'external examiner' model, which was also a set of interaction sequences. We then evaluated our PIPE modeling by the fraction of interaction sequences in the external examiner model that can be realized by an appropriate partial evaluation operation. We discounted optimizations such as described in Section 3.2 when determining the 'unrealizable' interaction sequences.

In the first study, the examiner model was obtained from users. They were provided knowledge of the functional specification of our original conceptual modeling, not its details. For instance, they were told about the nature of structural and terminal information (and any functional dependencies among them), but not the exact interaction sequences that constitute the conceptual model. Formal methodologies for this activity are described in [Gan97].

We identified 25 user subjects who were predominantly graduate students from Virginia Tech (but not necessarily computer science majors). The ages of the subjects ranged from 19 to 49, with the average age being 26. A majority of the subjects rated their computer and web familiarity as above average. All subjects acquainted themselves with the Project Vote Smart site by browsing for about ten minutes. Each subject was then asked to describe 1-2 personalization scenarios. Notice that these are different from 'queries,' as they specified constraints on interaction e.g., 'I would like to browse by state, and then I will make a choice of party, and then I would click any remaining

| | |
|---|---|
| PROBLEM #28 | $(w\,u_x)_x + (w\,u_y)_y = 1,$ where $w = \begin{cases} \alpha, & \text{if } 0 \le x, y \le 1 \\ 1, & \text{otherwise.} \end{cases}$ |
| DOMAIN | $[-1, 1] \times [-1, 1]$ |
| BC | $u = 0$ |
| TRUE | unknown |
| OPERATOR | Self–adjoint, discontinuous coefficients |
| RIGHT SIDE | Constant |
| BOUNDARY CONDITIONS | Dirichlet, homogeneous |
| ERROR CONSTRAINT | 1.0E-05 |
| TIME CONSTRAINT | 60s |

Figure 12: A problem from the examiner model for the second case study.

hyperlinks to browse the site.'

In total, 32 interaction sequences were identified, of which 25 were realizable in our modeling. One of the unmodelable scenarios was 'I would like to see all politicians who represent Los Angeles,' a request that was not faithful to our conceptual model. We do not discuss this further. The other six unmodelable scenarios are not shortcomings of our modeling, but rather shortcomings of the PIPE personalization methodology itself. They involved restructuring operations on interaction sequences that are not describable as partial evaluations. Section 5.2 analyzes these in detail.

For the second study, the examiner model was derived from a benchmark set of problems that are used in mathematical software evaluation (the set is described in [HCR+00]). Each of these problems describes scenarios in terms of features of the PDE problem (e.g., is it Laplace?, is it Helmholtz?) any constraints on its solution (e.g., relative error should be $< 10^{-9}$), and any restrictions on software modules (e.g., 'I would like to use the package NAG' or 'ELLPACK modules are prefered.'). Fig. 12 describes an example scenario that places constraints on the type of software to be used (for instance, it should be applicable to 'Dirichlet' problems) and the basis for recommendation (namely, that it should satisfy the time and error constraints specified). This scenario does not give any preferences for software modules or packages. Such mathematical descriptions are translated into parameters for personalization (a process is described in [HCR+00]). The examiner model comprised of 35 such interaction sequencs, of which all are modelable. More details on this case study can be obtained from [PLR00].

# 5 Discussion

## 5.1 Related Research

As a systematic methodology for personalization, PIPE is a unique research project. Most research on personalization emphasizes the nature of information being modeled [RV97, THA+97] (content-based [BS97] versus collaborative [AT01, AWWY99, KMM+97, RP97]), the level at which the personalized information is targeted (is it by user [MPR00], by topic [PE00] or for everybody [HH94, WM99]), or the specific algorithms that are involved in making recommendations.

In contrast, PIPE models interaction with an information system as the basis for personalization. Most of recommender systems research can be viewed as modeling options for PIPE. The systems that make distinctions among targeting constitute making different assumptions on the possible set of interaction sequences. They can hence be tied to requirements analysis, as described in [RRC01]. Systems that conduct web usage mining [MCS00, MAB00] also address the earlier parts (and sometimes, later parts [Spi00]) of the personalization system design lifecycle, and can be viewed as methodologies to suggest and refine interaction sequences.

Other connections to information systems research can be made by observing that PIPE contributes both a way to model information-seeking activities as well as a closed transformation operator for personalization i.e., partial evaluation. RABBIT [Wil84] is an early interactive information retrieval methodology that resembles PIPE in this respect. It proposes the model of 'retrieval by reformulation' to address the mismatch between how an information space is organized and how a particular user forages in it. Several closed transformation operators are provided in RABBIT to enable the user to specify and realize information-seeking goals. Like RABBIT, PIPE assumes that 'the user knows more about the generic structure of the [information space] than [PIPE] does, although [PIPE] knows more about the particulars ([terminal information]) [Wil84].' For instance, personalization by partial evaluation is only as effective as the ease with which program variables could be set (on or off) based on information supplied by the user. Unlike RABBIT, PIPE emphasizes the modeling of an information space as well as an information-seeking activity in a unified programmatic representation. Its single transformation operator is expressive enough to simplify a variety of interaction sequences.

The closed nature of transformation operators is central to interactive modes of information seeking, as shown in projects such as Scatter-Gather [CKPT92] and Dynamic Taxonomies [Sac00]. PIPE is novel in that it contributes a transformation operator for *representations of interactions* in information spaces, and does not transform documents or web pages directly.

The 'larger' approach to personalization taken in this paper is reminiscent of the integration of task models in software design [Ros99]. Typically such integration has utilized object oriented methodologies and symbolic modeling approaches e.g., UML. This idea has been used for designing personalization systems as well [CCR00, GSV00, KNV00, RSGa01]. However, in all of these projects, personalization is introduced a function from the conceptual design stage. PIPE's support for personalization, on the other hand, is built into the programmatic model of the information space and doesn't require any special handling.

## 5.2 When PIPE does not Work: Reasoning about Representations

We now address limitations and some fundamental implications of the PIPE methodology. We will explain why the six unmodelable interaction sequences in Section 4.1 are shortcomings of the PIPE methodology itself. Let us first recall why examples such as Fig. 5 and the other application study in Section 4 work so well: Information-seeking activities in these scenarios were describable as partial inputs in the modeling. Since the modeling was parameterized in terms of program variables, another way to explain the success of these applications is to say that 'the representation of the information space is factored in terms of structural information.'

This suggests that it will be useful to understand how information spaces are factored, in general. If the representation of the information space is not factored at all, it means that no program variables are available to be turned on or off and hence the space is not personalizable by PIPE. What is counterintuitive is that 'too much factoring' could also render PIPE inapplicable or useless.

**Over-factored Representations**

Consider our automobile example from Fig. 5 in Section 2; it is reproduced in Fig. 13 (right) with the addition of some line numbers (to denote particular points in the program). We can think of this as a factorization in terms of variables such as `Blue` and `Honda`, which in turn allow us to describe user requests. The left part of Fig. 13 describes an alternative factorization of the same information space. In this case, the program variables and their connections are stored in a 'structure table' and an explicit generator is used to construct the information space in Fig. 13 (right). For instance, the structure table associates the `Blue` program variable as the condition that gets us

| **Structure Table** | | |
|---|---|---|
| From Line | To Line | Program Variable |
| 1 | 2 | Blue |
| 1 | 3 | Red |
| 2 | 4 | 2001 |
| 2 | 5 | 2000 |
| ... | ... | ... |

× Site Generator (e.g., DFS) =

```
L1:
if (Blue)
L2:
 if (2001)
L4:
  if (Honda)
   .........
  else if (Toyota)
   .........
 else if (2000)
L5:
  .........
else if (Red)
L3:
 if (2001)
  .........
 else if (2000)
  .........
```

Figure 13: Reasoning about representations for personalization by partial evaluation. (left) Modeling the generation of an information space. (right) Modeling the interaction in an information space.

from line 1 to line 2 in the modeling. We can view the structure table as modeling the site graph and the generator as a depth-first search (DFS) algorithm that walks the site graph to construct the information space.

Rather than think of the left part of Fig. 13 as the *generator of an information space* and contrast it with the right side (which describes it directly), let us temporarily think of both the left and right sides of Fig. 13 as *alternative representations* of the same information space. The word ‘representation’ does not imply the mechanical aspect of constructing the information space (left of Fig. 13) or the interaction with the information space (right of Fig. 13). Since partial evaluation merely specializes programs, it doesn’t pay any attention to whether the program is meant to represent interaction or generation. By losing this distinction (temporarily), we will be able to reason about representations in general.

Let us consider how we will personalize the new design in Fig. 13 (left) w.r.t. '2001.' We cannot specify 2001 as input to the DFS algorithm since it is not parameterized in terms of specific variables like `2001`. The DFS is meant to work for all types of trees and graphs, not just an automobile browsing hierarchy. We also cannot specify `2001` in terms of the structure table since we have to manually readjust the line numbers to conform to the request. The only way we can obtain the same result as in Fig. 5 of Section 2 is to change the structure table in Fig. 13 completely to reflect the tree shown in Fig. 4 (right). But by then, we have done most of the work needed for personalization! In fact, the personalization request is no longer describable as partial evaluation, but as a *complete evaluation* (specifying all arguments). We say that such a design is *over-factored*, for the given information-seeking activity.

Attempting to use an over-factored representation in PIPE (for the type of information-seeking activities in Fig. 5) appears fruitless. The reason is that over-factorization divorces two crucial elements out, which really have to interplay for partial evaluation to be beneficial. Fig. 13 (left) is like two sides of the PIPE coin separated: the structure table contains the structural information (with which we connect user requests) and the DFS contains the logic flow (which is simplified by partial evaluation for the user). Neither is useful in PIPE without the other and yet they cannot be represented distinctly.

In contrast to partial evaluation, complete evaluation actually provides a functional mapping from structural to terminal information. This means that every possible variable is specified in advance, *obviating the need for any interaction* (with the information resource). Since PIPE emphasizes the representation and specialization of interaction, complete evaluation implies that PIPE's services are no longer required. *This is why over-factorization is not a good representation for PIPE.*

It is important to note that an information system design is not just over-factored; it is over-factored for a particular information-seeking activity. The design in Fig. 13 (left) is really meant to address scenarios such as:

> **Buyer:** I am here to buy a car. Ask me the questions for year, model, and color, in that order.

In this case, the user is not attempting to personalize an information space. Rather, he is seeking to personalize the *generation* of an information space (it is like creating a personalized automobile dealer agent). Assume that the site generator in Fig. 13 is parameterized in arguments describing what the first level of the hierarchy should be, what the second level should be, and so on. Presumably the generator would walk the tree described by the structure table and restructure it based on the arguments. In such a case, we can use partial evaluation with Fig. 13 (left) to answer requests such as:

> **Buyer:** I am here to buy a car. I don't care in what order you ask the questions, but the second question should be about year.

(It is a different issue if such scenarios are likely. For now, we are only exploring the PIPE concept theoretically.)

In other words, the design in Fig. 13 (left) is factored 'just right' for this type of activity. Our original modeling in Fig. 13 (right) cannot handle such requests. It can let the user give values out of turn, but it can't change the default order in which the questions are asked. We say that the design in Fig. 13 (right) is *under-factored* (for this activity).

**Defining Personable Information Spaces**

We can thus state the following three definitions:

> A representation $\mathcal{I}$ of an information space is well-factored for an information-seeking activity $\mathcal{G}$ if all interaction sequences in $\mathcal{G}$ can be realized by partial evaluations of $\mathcal{I}$. In this case, we also say that $\mathcal{I}$ is personable for $\mathcal{G}$.
>
> A representation $\mathcal{I}$ of an information space is over-factored for an information-seeking activity $\mathcal{G}$ if all interaction sequences in $\mathcal{G}$ can be realized by complete evaluations of $\mathcal{I}$. In this case, we also say that $\mathcal{I}$ is not personable for $\mathcal{G}$.
>
> A representation $\mathcal{I}$ of an information space is under-factored for an information-seeking activity $\mathcal{G}$ if no interaction sequences in $\mathcal{G}$ can be realized by partial (or complete) evaluations of $\mathcal{I}$. In this case, we also say that $\mathcal{I}$ is not personable for $\mathcal{G}$.

Thus, a given representation could be well-factored for one information-seeking activity but over-factored for another. Fig. 13 (left) is well-factored for generation but over-factored for interaction. Fig. 13 (right) is well-factored for interaction but over-factored for users who specify choices of all variables — color, year, and model — in advance.

The 6 unmodelable scenarios from the politicians case study in Section 4.3 involved requests such as 'I would like to have the choice of party as the first level of the hierarchy, the choice of state as the second level.' Our design was obviously under-factored for such interaction sequences. We can define the *personability* of a representation as the fraction of interaction sequences in a (external examiner) model that are describable as partial evaluations. For the external examiner model described in Section 4.3, the personability of the PIPE modeling (presented in Section 4.1) is thus 25/32.

Notice that all of these statements assume that the model for transforming representations is *partial evaluation.* There are other program-transformation techniques which might be able to address the unmodelable requests above, but PIPE only provides partial evaluation as the operator for personalization. Our statements should only be interpreted in the context of personalization by partial evaluation.

**Representations in Practice**

In practice, the decision of choosing a factoring will depend on which situations are more likely and also the composition of the space of interaction sequences $\mathcal{G}$. It is acceptable to have some interaction sequences that involve complete evaluation, as long as they are a small fraction of the total number of interaction sequences.

Thus far, we have fixed the representation and analyzed the information-seeking activities for which it was over-factored, the ones for which it was under-factored, and so on. This is the designer's viewpoint. For a given site design, it allows the designer to pose questions such as 'What are the information-seeking activities for which my site is personable?'

An alternate viewpoint is user-driven. Given an information-seeking activity, the user asks 'Which sites are most personable for my activity?' This allows the user to take different site designs (along with representations), analyze them w.r.t. a conceptual model of information seeking, and rank them in order of personability. For instance, consider again the external examiner model described in Section 4.3 for the politicians case study. One information system design was described in Section 4.1. The personability of this design is, as stated earlier, 25/32. Seven interaction sequences were not modelable. Another information system design is the representation in Fig. 13 (left). The personability of this design is 6/32. While it accommodates six of the seven sequences, it is no longer personable for the original 25 sequences! This is because those 25 sequences are now describable as complete evaluations, which also violate the partial evaluation model! Thus, both over-factorization and under-factorization lead to unpersonable information spaces (see the definitions above). We hypothesize that the most interesting representations are in between.

An open research issue is if we have to cross the barrier from interaction to generation to arrive at over-factored representations.

## 6 Concluding Remarks

This paper makes several major contributions. We have presented a novel modeling methodology for information personalization. PIPE enables the view of personalization as specializing representations. It models interactions with information systems and uses partial evaluation to simplify the interactions. PIPE also contributes a novel evaluation criterion for information system designs. It relates personalization to the way an information system design is factored. This has implications for how web applications are developed and deployed [FP00]. Many web sites today are based on the generator model; the results in this paper indicate that they might not be directly personable for interaction scenarios (under partial evaluation). One solution is to re-represent the generated information spaces in terms of interaction.

Our modeling makes very weak assumptions on the nature of interactions with information systems. While we have covered only web sites (and collections of web sites) in this paper, any information system technology that affords the notion of interaction sequence or the idea of factorization can be studied on similar lines. This especially applies to designs for voice-activated systems (e.g., VoiceXML), directory access protocols (e.g., LDAP), information systems that provide a dialog model of interaction, and models for organizing digital libraries (e.g., 5S).

We plan to extend the PIPE methodology in several directions. We would like to extend the modeling methodology 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 a companion paper [RRC01]. Another important direction of future work involves modeling *context* in personalization systems. The programmatic modeling provided in PIPE suggests that context can be usefully viewed as partial information. We believe that more sophisticated forms of modeling partial information will be needed for describing context, besides values for program variables. We are also interested in relaxing our assumptions of bounded sequences that have separable structural and terminal parts. This will allow us to address other information-seeking activities such as social network navigation. In addition, we are investigating program transformation techniques that can help reason about terminal information (e.g., program slicing [BG96]), in addition to structural information.

Our long-term goal is to develop a theory of reasoning about representations of information spaces. This will allow us to formally study the design and implementation of information systems in terms of the representations they employ.

## Acknowledgements

Many of the ideas in this paper were developed during the Spring 2001 offering of the CS 6604 course (on recommender systems and personalization) at Virginia Tech. We acknowledge helpful discussions with Rob Capra, Jack Carroll, Marcos Gonçalves, Ed Fox, Dennis Kafura, Priya Lakshminarayanan, Dick Nance, Manuel Perez, and Mary Beth Rosson. In addition, Rob Capra helped establish connections between PIPE and mixed-initiative interaction and provided ideas for evaluating the modeling of personalization systems. Ed Fox suggested the usage of 'structural' and 'terminal' information to qualify interaction sequences. Comments from several anonymous referees helped improve the presentation of the article.

## References

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

[AK97] N. Ashish and C. Knoblock. Wrapper Generation for Semi-Structured Internet Sources. *ACM SIGMOD Record*, Vol. 26(4):pages 8–15, December 1997.

[ALW99] C. Anderson, A. Levy, and D. Weld. Web-Site Management with Tiramisu. In *Proceedings of the Web/DB Workshop, SIGMOD 1999*, pages 19–24, 1999.

[AT01] G. Adomavicius and A. Tuzhilin. Using Data Mining Methods to Build Customer Profiles. *IEEE Computer*, Vol. 34(2):pages 74–82, February 2001.

[AWWY99] C.C. Aggarwal, J.L. Wolf, K.-L. Wu, and P.S. Yu. Horting Hatches an Egg: A New Graph-Theoretic Approach to Collaborative Filtering. In *Proceedings of the Fifth ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'99)*, pages 201–212. San Diego, CA, 1999.

[Ber01] H. Berghel. Caustic Cookies. *Communications of the ACM*, Vol. 44(5):pages 19–22, May 2001.

[BG96] D.W. Binkley and K.B. Gallagher. Program Slicing. *Advances in Computers*, Vol. 43:pages 1–50, 1996.

[BS97] M. Balabanović and Y. Shoham. Fab: Content-Based, Collaborative Recommendation. *Communications of the ACM*, Vol. 40(3):pages 66–72, 1997.

[CCR00] K. Claypool, L. Chen, and E.A. Rudensteiner. Personal Views for Web Catalogs. *IEEE Data Engineering Bulletin*, Vol. 23(1):pages 10–16, March 2000.

[CDA00] I. Cingil, A. Dogac, and A. Azgin. A Broader Approach to Personalization. *Communications of the ACM*, Vol. 43(8):pages 136–141, August 2000.

[CDF+00] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to Construct Knowledge Bases from the World Wide Web. *Artificial Intelligence*, Vol. 118:pages 69–113, 2000.

[CKPT92] D.R. Cutting, D. Karger, J. Pedersen, and J.W. Tukey. Scatter/Gather: A Cluster-Based Approach to Browsing Large Document Collections. In *Proceedings of the Fifteenth Annual International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 318–329. Copenhagen, Denmark, June 1992.

[dOTM01] M.C.F. de Oliveira, M.A.S. Turine, and P.C. Masiero. A Statechart-Based Model for Hypermedia Applications. *ACM Transactions on Information Systems*, Vol. 19(1):pages 28–52, January 2001.

[FFK+98] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. Catching the Boat with Strudel: Experience with a Web-Site Management System. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD'98)*, pages 414–425. 1998.

[FLM98] D. Florescu, A. Levy, and A. Mendelzon. Database Techniques for the World-Wide Web: A Survey. *SIGMOD Record*, Vol. 27(3):pages 59–74, September 1998.

[FMMT99] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Mining Optimized Association Rules for Numeric Attributes. *Journal of Computer and Systems Sciences*, Vol. 58(1):pages 1–12, 1999.

[FP00] P. Fraternali and P. Paolini. Model-Driven Development of Web Applications: The AutoWeb System. *ACM Transactions on Information Systems*, Vol. 18(4):pages 323–382, October 2000.

[Gan97] J.D. Gannon. Verification and Validation. In A.B. Tucker, editor, *The Computer Science and Engineering Handbook*, chapter 109, pages 2352–2378. CRC Press, 1997.

[GGR+00] M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. XTRACT: A System for Extracting Document Type Descriptors from XML Documents. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD'2000)*, pages 165–176. 2000.

[Gia98] J.C. Giarratano. *Expert Systems: Principles and Programming*. Brooks/Cole Publishing, 1998.

[GSV00] N. Guell, D. Schwabe, and P. Villain. Modeling Interactions and Navigation in Web Applications (Extended Version). In *Lecture Notes in Computer Science: Proceedings of the World Wide Web and Conceptual Modeling'00 Workshop, ER'00 Conference*, volume 1921. Springer, Salt Lake City, 2000.

[HCR+00] E.N. Houstis, A.C. Catlin, J.R. Rice, V.S. Verykios, N. Ramakrishnan, and C.E. Houstis. PYTHIA-II: A Knowledge/Database System for Managing Performance Data and Recommending Scientific Software. *ACM Transactions on Mathematical Software*, Vol. 26(2):pages 227–253, June 2000.

[Hea00] M. Hearst. Next Generation Web Search: Setting Our Sites. *IEEE Data Engineering Bulletin*, Vol. 23(3):pages 38–48, September 2000.

[HH94] W.C. Hill and J.D. Hollan. History-Enriched Digital Objects. *The Information Society*, Vol. 10:pages 139–145, 1994.

[Jon96] N.D. Jones. An Introduction to Partial Evaluation. *ACM Computing Surveys*, Vol. 28(3):pages 480–503, September 1996.

[Jon97] N.D. Jones. *Computability and Complexity: From a Programming Perspective*. MIT Press, Cambridge, Massachusetts, 1997.

[KBM+00] P.B. Kantor, E. Boros, B. Melamed, V. Menkov, B. Shapira, and D.J. Neu. Capturing Human Intelligence in the Net. *Communications of the ACM*, Vol. 43(8):pages 112–115, August 2000.

[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)*, pages 211–218, 1998. Madison WI.

[KMM+97] J.A. Konstan, B.N. Miller, D. Maltz, J.L. Herlocker, L.R. Gordon, and J. Riedl. GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, Vol. 40(3):pages 77–87, March 1997.

[KNV00] J. Kramer, S. Noronha, and J. Vergo. A User-Centered Design Approach to Personalization. *Communications of the ACM*, Vol. 43(8):pages 45–48, August 2000.

[Kus00] N. Kushmerick. Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence*, Vol. 118:pages 15–68, 2000.

[MAB00] M.D. Mulvenna, S.S. Anand, and A.G. Buchner. Personalization on the Net Using Web Mining. *Communications of the ACM*, Vol. 43(8):pages 123–125, August 2000.

[MB00] P. Maglio and R. Barrett. Intermediaries Personalize Information Streams. *Communications of the ACM*, Vol. 43(8):pages 96–101, August 2000.

[MCS00] B. Mobasher, R. Cooley, and J. Srivastava. Automatic Personalization Based on Web Usage Mining. *Communications of the ACM*, Vol. 43(8):pages 142–151, August 2000.

[MPR00] U. Manber, A. Patel, and J. Robison. Experience with Personalization on Yahoo! *Communications of the ACM*, Vol. 43(8):pages 35–39, August 2000.

[NAM98] S. Nestorov, S. Abiteboul, and R. Motwani. Extracting Schema from Semistructured Data. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD'98)*, pages 165–176, 1998.

[NS97] D.G. Novick and S. Sutton. What is Mixed-Initiative Interaction? In S. Haller and S. McRoy, editors, *Procedings of the AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction*. AAAI/MIT Press, 1997.

[PE00] M. Perkowitz and O. Etzioni. Adaptive Web Sites. *Communications of the ACM*, Vol. 42(8):pages 152–158, August 2000.

[Pir01] P. Pirolli. Exploring and Finding Information. In J.M. Carroll, editor, *Toward a Multidisciplinary Science of Human-Computer Interaction*. Morgan Kaufmann, San Francisco, CA, 2001. to appear.

[PLR00] S. Perugini, P. Lakshminarayanan, and N. Ramakrishnan. Personalizing the GAMS Cross-Index. Technical Report TR-00-01, Department of Computer Science, Virginia Tech, March 2000.

[Ram00] N. Ramakrishnan. PIPE: Web Personalization by Partial Evaluation. *IEEE Internet Computing*, Vol. 4(6):pages 21–31, Nov-Dec 2000.

[Res99] P. Resnick. Semantic Similarity in a Taxonomy: An Information-Based Measure and its Application to Problems of Ambiguity in Natural Language. *Journal of Artificial Intelligence Research*, Vol. 11:pages 95–130, 1999.

[Rie00] D. Riecken. Personalized Views of Personalization. *Communications of the ACM*, Vol. 43(8):pages 26–28, 2000.

[Ros99] M.B. Rosson. Integrating Development of Task and Object Models. *Communications of the ACM*, Vol. 42(1):pages 49–56, 1999.

[RP97] J. Rucker and M.J. Polano. Siteseer: Personalized Navigation for the Web. *Communications of the ACM*, Vol. 40(3):pages 73–75, 1997.

[RRC01] N. Ramakrishnan, M.B. Rosson, and J.M. Carroll. Explaining Scenarios for Information Personalization. *ACM Transactions on Computer-Human Interaction*, August 2001. Communicated for publication. Also available as Technical Report, Computing Research Repository (CoRR) at `http://xxx.lanl.gov`.

[RS97] D. Rus and D. Subramanian. Customizing Information Capture and Access. *ACM Transactions on Information Systems*, Vol. 15(1):pages 67–101, 1997.

[RSGa01] G. Rossi, D. Schwabe, and M. Guimarães. Designing Personalized Web Applications. In *Proceedings of the World Wide Web Conference (WWW'10)*. Hong Kong, May 2001.

[RV97] P. Resnick and H.R. Varian. Recommender Systems. *Communications of the ACM*, Vol. 40(3):pages 56–58, 1997.

[Sac00] G.M. Sacco. Dynamic Taxonomies: A Model for Large Information Bases. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12(3):pages 468–479, May/June 2000.

[Spi00] M. Spiliopoulou. Web Usage Mining for Web Site Evaluation. *Communications of the ACM*, Vol. 43(8):pages 127–134, August 2000.

[THA+97] L. Terveen, W. Hill, B. Amento, D.W. McDonald, and J. Creter. PHOAKS: A System for Sharing Recommendations. *Communications of the ACM*, Vol. 40(3):pages 59–62, March 1997.

[THA99] L. Terveen, W. Hill, and B. Amento. Constructing, Organizing, and Visualizing Collections of Topically Related Web Resources. *ACM Transactions on Computer-Human Interaction*, Vol. 6(1):pages 67–94, March 1999.

[Wil84] M.D. Williams. What makes RABBIT run? *International Journal of Man-Machine Studies*, Vol. 21:pages 333–352, 1984.

[WL00] K. Wang and H. Liu. Discovering Structural Association of Semistructured Data. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12(3):pages 353–371, May/June 2000.

[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)*, pages 270–277. Pittsburgh, PA, 1999.