


2006

# Realtime Query Expansion and Procedural Interfaces for Information Hierarchies

Saverio Perugini

*University of Dayton*, [sperugini1@udayton.edu](mailto:sperugini1@udayton.edu)

Follow this and additional works at: [http://ecommons.udayton.edu/cps\\_fac\\_pub](http://ecommons.udayton.edu/cps_fac_pub)

 Part of the [Databases and Information Systems Commons](#), [Graphics and Human Computer Interfaces Commons](#), [Numerical Analysis and Scientific Computing Commons](#), [OS and Networks Commons](#), [Other Computer Sciences Commons](#), and the [Theory and Algorithms Commons](#)

---

## eCommons Citation

Perugini, Saverio, "Realtime Query Expansion and Procedural Interfaces for Information Hierarchies" (2006). *Computer Science Faculty Publications*. Paper 23.

[http://ecommons.udayton.edu/cps\\_fac\\_pub/23](http://ecommons.udayton.edu/cps_fac_pub/23)

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

# Real-time Query Expansion and Procedural Interfaces for Information Hierarchies

[Track B: Demos]

Saverio Perugini  
Department of Computer Science  
University of Dayton  
Dayton, OH 45469-2160  
saverio@udayton.edu

Demo website: <http://oot.cps.udayton.edu>

## ABSTRACT

We demonstrate the use of two user interfaces for interacting with web hierarchies. One uses the dependencies underlying a hierarchy to perform real-time query expansion and, in this way, acts as an *in situ* feedback mechanism. The other enables the user to cascade the output from one interaction to the input of another, and so on, and, in this way, supports procedural information-seeking tasks without disrupting the flow of interaction.

## Categories and Subject Descriptors

H.5.4 [Information Interfaces and Presentation]: Hypertext/Hypermedia—*navigation*; H.5.2 [Information Interfaces and Presentation]: User Interfaces—*interaction styles*

## General Terms

information hierarchies, query expansion

## Keywords

out-of-turn interaction, mixed-initiative interaction

## 1. INTRODUCTION

Information hierarchies are ubiquitous on the web. They present site designers with a natural way to organize information and users with an intuitive and familiar metaphor for navigation. The following two aspects of information hierarchies lead to user interfaces for interacting with them which we demonstrate:

- Information hierarchies are teeming with dependencies between facet values.
- Traditional interaction with information hierarchies is destructive.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR'06 Workshop on Faceted Search Seattle, Washington USA  
Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

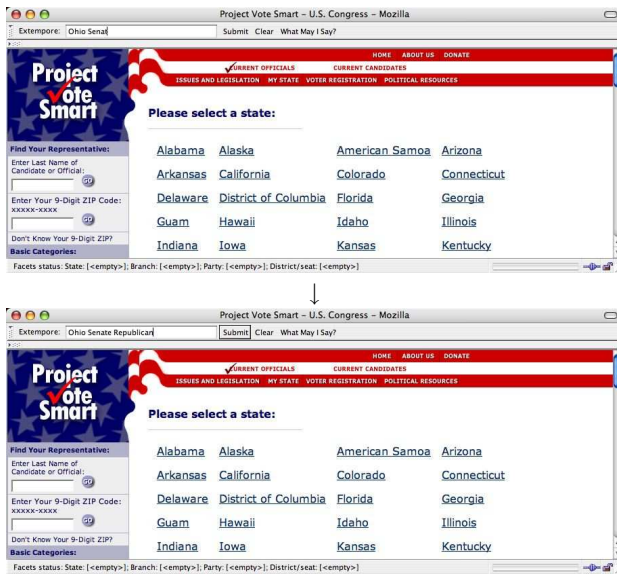
The goal of this demonstration is to illustrate user interfaces to information hierarchies which address these issues.

In what follows we describe how dependencies arise in web hierarchies and demonstrate a real-time query expansion interface which helps to expose dependencies during information-seeking. In addition, we describe how destructive navigation inhibits the completion of procedural information-finding tasks in hierarchies and demonstrate an interface for supporting such tasks. We also describe *out-of-turn interaction* which is a simple approach to integrating querying and browsing in web hierarchies. We use a hierarchy induced from the congressional portion of Project Vote Smart (PVS; [vote-smart.org](http://vote-smart.org)), a website which indexes the webpages of US politicians, to help illustrate these concepts. This hierarchy is a tree consisting of four levels: *state*, *branch* of Congress (House or Senate), *party* (Democrat, Republican, or Independent), and *district number* or *seat* (junior or senior), in that order. Users retrieve the webpage of a congressperson by progressively specifying values for these four facets, which uniquely identify a congressperson, by drilling-down the hierarchy.

## 2. QUERY EXPANSION INTERFACE

There are several ways which to think of dependencies in information hierarchies [7]. For example, we can think of dependencies between hyperlink labels (i.e., the text between the `<a href="...">` and `</a>` called a *term* here) based on how they co-occur with each other along paths through a site. Specifically, we say a *web functional dependency* (FD) ' $x \rightarrow y$ ' is satisfied by site  $S$  if all paths from the root to a leaf in  $S$  which contain a hyperlink labeled  $x$  also contain a hyperlink labeled  $y$ . Note that a site which satisfies ' $x \rightarrow y$ ' does not necessarily satisfy ' $y \rightarrow x$ .' Note also that  $x$  and  $y$  may represent sets of terms rather than only single terms.

We mined web FDs of this type in five commercial and research websites [7]. We found that web FDs occur frequently in these sites, even in those with a small number of terms, nodes, paths, and levels. For example, while PVS has only 116 terms, 856 nodes, 540 paths, and four levels, it satisfies 715 web FDs. We posit that the complete set of web FDs satisfied by a hierarchy encodes the domain knowledge implicit in the taxonomy. For example, some FDs satisfied by PVS, such as 'District 1  $\rightarrow$  House,' 'Senior seat  $\rightarrow$  Sen-



**Figure 1: Illustration of a real-time query expansion interface to PVS.**

ate,’ and ‘Washington DC → House, Democrat, District at Large,’ are obvious from the domain. Moreover, most political pundits might also know that ‘Hawaii → Democrat’ (because all congresspeople from Hawaii are members of the Democratic party) and ‘Ohio, Senate → Republican’ (because both senators representing Ohio are members of the Republican party). However, most users would be hard-pressed to know that ‘New York, District 18 → Democrat.’

We can use web FDs to expand queries over the hyperlink labels of a site. For example, using the ‘Ohio, Senate → Republican’ FD, we can expand a query for ‘Ohio Senate’ over PVS to ‘Ohio Senate Republican.’ Fig. 1 illustrates an interface which expands a user’s query in real-time before it is submitted to the system. Here a user has entered the string ‘Ohio Senat’ into the toolbar embedded into the web browser (see Fig. 1, top). As soon as the user enters the final ‘e’ in the specification of ‘Ohio Senate,’ the system expands the query to ‘Ohio Senate Republican’ (see Fig. 1, bottom). Such a toolbar can be implemented using XUL (XML User-interface Language; xulplanet.com), a markup language for developing cross-platform user interfaces for use with the Mozilla Firefox web browser.

Over the years researchers have developed several ways to search over an information hierarchy [4, 9]. Since web FDs, as presented, are computed from hyperlink labels, it is most appropriate to use them to expand queries for searches over site schema, rather than non-hyperlink content, if any, from the pages of the hierarchy.

### 3. OUT-OF-TURN INTERACTION

Out-of-turn interaction [5] is a technique for navigating hierarchical websites which augments traditional browsing by empowering the user to supply a hyperlink label which is presented beyond the current webpage (hence out-of-turn) to initiate a search over the site’s hierarchical schema. When the system receives an out-of-turn input, it removes all paths through the site which do not contain a hyperlink labeled with the input and removes the hyperlink labeled with the



**Figure 3: Illustration of an out-of-turn interaction with PVS through the *Extempore* toolbar.**

input from the remaining paths. Fig. 2 illustrates how a sample hierarchy with a structure similar to that of PVS would be pruned based on supplying ‘Republican’ out-of-turn. Notice that all paths leading to the webpages of Democratic politicians (nodes 20, 21, 24, 25, 26, 27, 30, 31, 32, and 33) have been removed. In addition, the hyperlinks labeled ‘Republican’ in the remaining paths (those leading to nodes 22, 23, 28, 29, 34, 35, 36, and 37) have been removed.

Fig. 3 illustrates an out-of-turn interaction through a browser toolbar we call *Extempore* (as it permits the user to supply terms *extemporaneously*). Here the user supplies ‘Republican’ out-of-turn (see Fig. 3, top). This causes some of the hyperlinks presented on the root page (e.g., Hawaii), those which do not lead to the webpages of Republican congresspeople, to be pruned out (see Fig. 3, bottom). When used in conjunction with traditional browsing, the unsolicited reporting [1] involved in supplying an out-of-turn input supports a simple form of mixed-initiative interaction [8] and can be viewed as an approach to integrating querying and browsing in information hierarchies [2].

In sites where each level of the hierarchy corresponds to a facet of information assessment, such as PVS, out-of-turn interaction permits the user to explore the facets in any order without the designer enumerating all possible paths of navigation. In hierarchies where each level does not correspond to a facet, such as Yahoo! and the Open Directory Project (ODP) at dmoz.org, out-of-turn interaction behaves more as a pruning operator and reveals to the user the portions of the taxonomy pertaining to their query. For example, ODP contains 16 top-level categories and a user starting from the homepage would be hard-pressed to know that only four (Home, Shopping, Business, and Regional) contain links to information about ‘ice cream makers.’ An out-of-turn interaction reveals these categories. In fact, the search feature provided in ODP is similar to out-of-turn interaction with the exception that ODP flattens the hierarchical structure in response to a query (see Fig. 4) whereas out-of-turn interaction preserves the hierarchical nature in order to retain

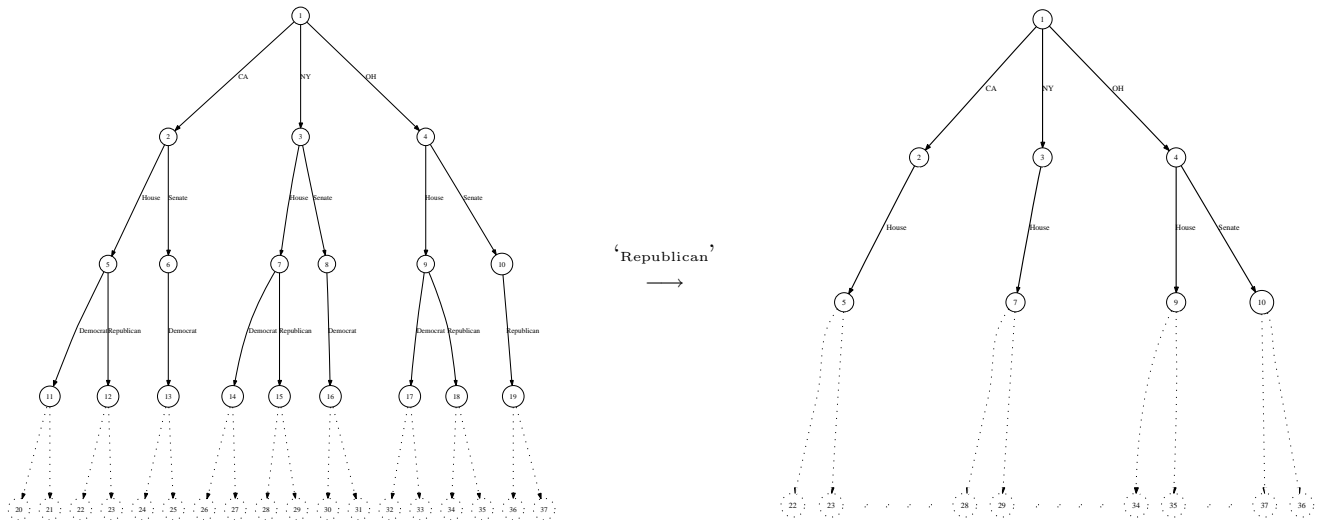


Figure 2: Illustration of the pruning conducted as a result of out-of-turn interaction. (left) Sample hierarchy, simplified for purposes of presentation, with characteristics similar to those in PVS. (right) Site schema resulting from supplying 'Republican' out-of-turn.

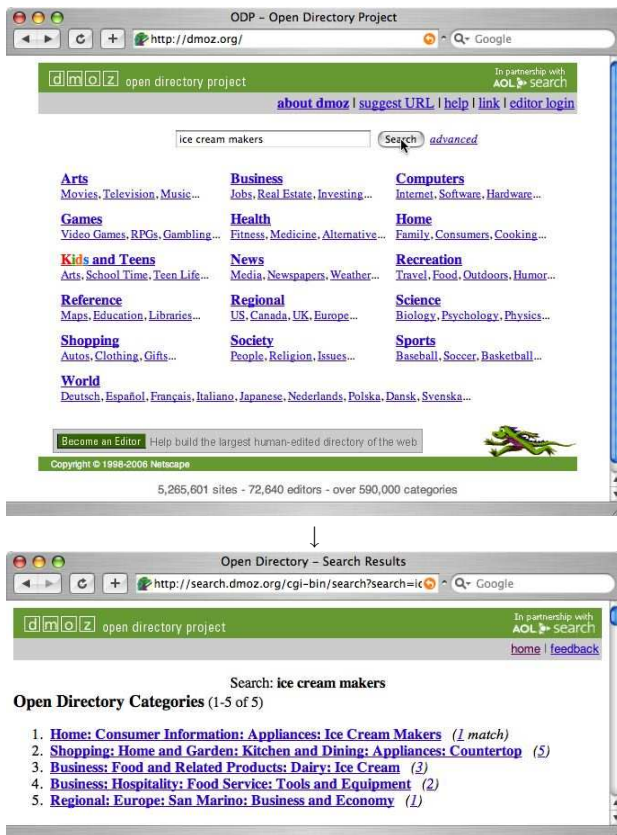


Figure 4: (top) A query for 'ice cream makers' in the Open Directory Project and (bottom) its result as a flat list.

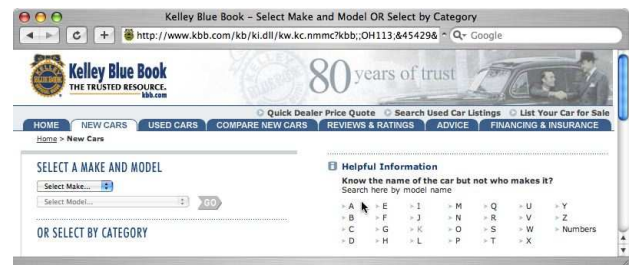


Figure 5: Facility for automobile-make lookup by model in the online Kelley Blue Book.

context. Notice that with the interpretation of out-of-turn interaction presented here, an unexpanded query will yield the same result as its expanded version and therefore query expansion here is simply a feedback mechanism to expose dependencies, unlike its use in traditional IR.

There are other means of exposing dependencies underlying information hierarchies during information-seeking. For example, the Kelley Blue Book (KBB) online at kbb.com provides a facility for automobile-make lookup by model (see under heading titled 'Helpful Information' in Fig. 5) since FDs of the form ' $model \rightarrow make$ ' are implicit in the domain of automobiles. When browsing new cars in KBB, users are first asked to make a selection for automobile make (see under heading titled 'SELECT A MAKE AND MODEL' in Fig. 5). The lookup facility allows the user to search for the make of an automobile based on the model so that they can proceed with the information-gathering dialog on the left side of the window in Fig. 5. A more sophisticated example of support for dependency exploration is *Sony's Advisor* facility available through sonystyle.com when browsing products such as digital cameras and camcorders.

## 4. SUPPORTING PROCEDURAL TASKS

We say a *procedural task* is one involving multiple sub-tasks. The sub-tasks typically must be completed in a particular order because the results of one may be required to begin another. For instance, consider completing the following procedural information-finding task in PVS:

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

This task involves two distinct sub-tasks. First the user must identify the only state with congresspeople from the Independent party. Only once that requisite information is found can the user tackle the second sub-task – finding the political party of the senior Senator from that state.

Attempting to complete such a task manually in the PVS hierarchy is challenging, even given a faceted interface. For example, the user could simply supply ‘Independent’ from the root page. In response, the site would present a list of only those states with congresspeople from the Independent party (and since there is only one it would be inferred by functional dependency). However, neither the hierarchy nor a faceted interface to it support the user in naturally cascading the information found (i.e., the only state with congresspeople from the Independent party) onto the activity required to complete the second sub-task. The underlying reason why such procedural tasks are challenging to pursue manually is because drill-down navigation with hierarchies is fundamentally *destructive*. Once a user clicks on a hyperlink, he is now working from new, reduced hierarchy – that rooted at the target page of the hyperlink clicked, and so on. The only way to return to a prior, more complete version of the hierarchy is to roll-up by clicking the back button or using purely navigation links, if provided. Users need *constructive* information-seeking operators, which permit them to connect two disjoint paths through the hierarchy, for procedural tasks.

In an exploratory study of out-of-turn interaction [6], we gave users the procedural task above without revealing to them that it was procedural. We equip users with only the PVS hierarchy and the ability to interact with it out-of-turn. Only 12 of 24 participants were able to figure out that the task was procedural and solved it by manually rolling-up to the root of the taxonomy once they found the requisite information and began a new interaction with the complete hierarchy to pursue the second sub-task thereby completing the entire task. Fig. 6 gives a schematic which helps illustrate the approach taken by the 12 participants who solved the task manually as well as the 12 who did not solve the task. The participants who did not solve the task followed the path indicated by interactions labeled 1 and 2b. Once they supplied ‘Independent’ out-of-turn (see the interaction labeled 1; notice that ‘Vermont’ was inferred by functional dependency), they simply clicked on ‘Senate’ (see the interaction labeled 2b), and were directed to the webpage of the junior, rather than senior, Senator from Vermont! It was clear that these participants wanted to *continue* their current interaction with the site, but the destructive nature of drill-down inhibited them from doing so in a manner leading to task completion. The other 12 participants solved the task through back-tracking. Some of them never reached the webpage of the junior Senator from Vermont because they

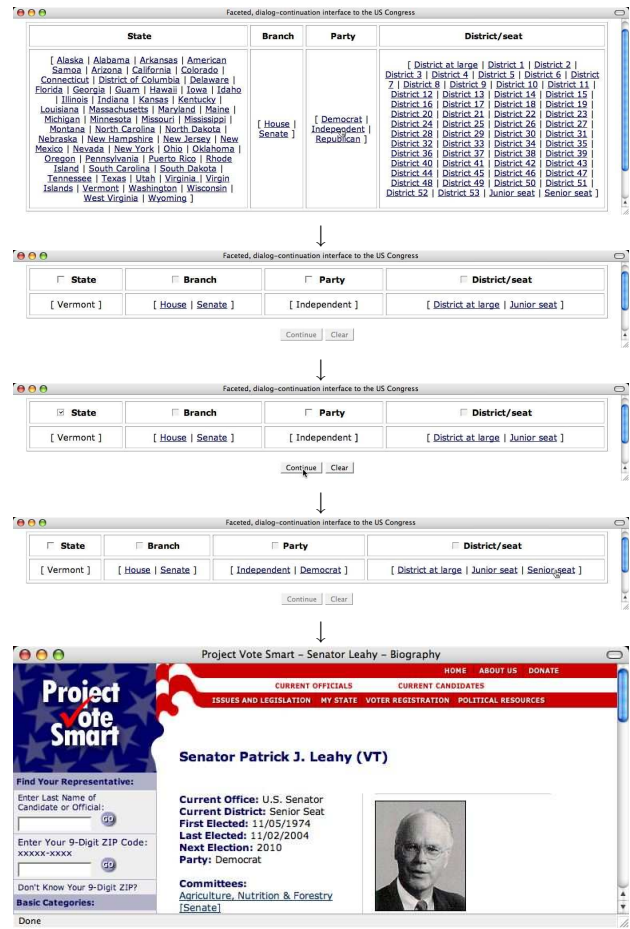
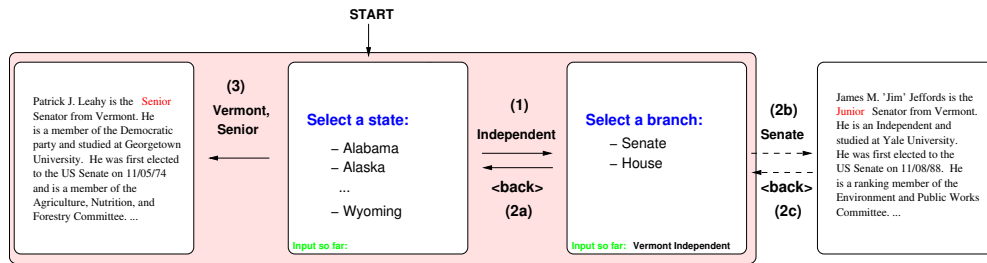


Figure 7: Cascading information across sub-tasks using user-initiated continuation.

decomposed the task correctly from the start and, therefore, followed the path indicated by the interactions labeled 1, 2a, and 3. Others first reached the webpage of the junior Senator from Vermont, realized there that the task was procedural by observing that it was not the correct page, and then from there rolled-up to the root to start a new interaction. These participants followed the path indicated by the interactions labeled 1, 2b, 2c, 2a, and 3.

In Fig. 7 we demonstrate how a user might complete the procedural task above using an interface which allows her to supply the output from any interaction with the PVS hierarchy as the input to another. We use a faceted design in Fig. 7 for clarity. In the first window the user clicks on the hyperlink labeled ‘Independent’ to start the first sub-task. As a result of this interaction, we see that state and party are fixed as Vermont and Independent, respectively, in the second window. The party (Independent) was supplied by the user and the state (Vermont) was inferred through functional dependency. Notice that two congresspeople representing Vermont are from the Independent party. However, the Senator representing Vermont from the Independent party holds the junior seat rather than the senior seat in which the user is ultimately interested. Therefore, the user will be unable to complete the composite task by continuing the interaction from the hyperlinks available from





**Figure 6: A procedural task:** the user is expected to first supply ‘Independent’ out-of-turn (interaction 1), notice that choice of state is automatically triggered to Vermont as the only state having Independents (as shown in the ‘Input so far:’ notification), return to the root page (interaction 2a), and supply ‘Vermont’ and ‘Senior’ out-of-turn to complete the task (interaction 3). Interactions 2b and 2c depict unnecessary and irrelevant interactions for this task.

the second window or, in other words, continuing *this* line of inquiry. However, since the requisite (state) information is now found, the user instead submits a new query for only ‘Vermont’ over the complete version of the hierarchy by selecting the checkbox labeled ‘State’ and clicking the ‘Continue’ button in the third window. Notice that while a faceted (or out-of-turn interaction) interface allows the user to *supply* partial information, incrementally and in any order, en route to a leaf page, the procedural interface allows the user to *retrieve* any subset of partial information supplied (or inferred) thus far and submit it as a *new* query over the *pristine* version of the hierarchy. We say that the user is replacing the system’s *continuation* [3] of the information-seeking activity with their own, *user-initiated* version of it. In the fourth window, the system again prompts the user for branch, but notice now that the only facet fixed is ‘State’ (on Vermont). Notice further that a hyperlink labeled ‘Senior seat’ is an available choice for the District/seat facet. At this point the user is tackling the second sub-task and, thus, clicks on the hyperlink labeled ‘Senior seat’ in the fourth window. This brings the user directly to the webpage of the senior Senator from Vermont in the fifth window. Notice that branch (Senate) and party (Democrat) were inferred by functional dependency. The user has now successfully completed the composite task. Lastly, note that real-time query expansion, support for out-of-turn input, and the ability to cascade the output from one interaction to the input of another are orthogonal features in an interface to an information hierarchy.

## 5. FUTURE WORK

We have demonstrated two interfaces for interacting with hierarchies. The first expands the user’s query in real-time based on the dependencies implicit in the hierarchy. We posit that this interface, by exposing dependencies, can help the user assimilate a new domain. We intend to conduct a user study to evaluate how expansion affects assimilation. We plan to recruit 40 participants unfamiliar with a particular domain (determined by pre-questionnaire screening), such as politics or automobiles, ask half of them to explore the hierarchy with the expansion interface and half without it, and finally re-evaluate their understanding of the domain.

The second interface we showed supports procedural tasks by permitting the user to supply the output from one interaction as the input to another. We are optimistic that this interface will be helpful for solving constraint satisfaction

problems (CSP), such as course scheduling or planning a vacation, where users need to integrate information from multiple web sources. We plan to conduct a user study to evaluate how the procedural interface affects constraint satisfaction. We intend to recruit 40 participants, ask half of them to solve a CSP with the procedural interface and half without it, and finally evaluate which group was able to satisfy more constraints.

## 6. REFERENCES

- [1] J. F. Allen, C. I. Guinn, and E. Horvitz. Mixed-Initiative Interaction. *IEEE Intelligent Systems*, Vol. 14(5):pp. 14–23, 1999.
- [2] R. Bodner and M. Chignell. Dynamic Hypertext: Querying and Linking. *ACM CSUR*, Vol. 31(4es), Dec. 1999. Article No. 15.
- [3] D. P. Friedman, M. Wand, and C. T. Haynes. *Essentials of Programming Languages*. MIT Press, Second edition, 2001.
- [4] H. Meuss and K. U. Schulz. Complete Answer Aggregates for Treelike Databases: A Novel Approach to Combine Querying and Navigation. *ACM TOIS*, Vol. 19(2):pp. 161–215, 2001.
- [5] M. Narayan, C. Williams, S. Perugini, and N. Ramakrishnan. Staging Transformations for Multimodal Web Interaction Management. In *Proc. of WWW’04*, pp. 212–223.
- [6] S. Perugini. *Program Transformations for Information Personalization*. Ph.D. dissertation, Department of Computer Science, Virginia Tech, 2004.
- [7] S. Perugini and N. Ramakrishnan. Mining Web Functional Dependencies for Flexible Information Access. Under review for the Special Issue of *JASIST* on Mining Web Resources for Enhancing IR.
- [8] S. Perugini and N. Ramakrishnan. Personalizing Web Sites with Mixed-Initiative Interaction. *IEEE IT Professional*, Vol. 5(2):pp. 9–15, 2003.
- [9] G. M. Sacco. Dynamic Taxonomies: A Model for Large Information Bases. *IEEE TKDE*, Vol. 12(3):pp. 468–479, 2000.