Subjunctive Interfaces for the Web

Aran Lunzer
Meme Media Laboratory, Hokkaido University,
North 13 West 8, Sapporo 060-8628, Japan
aran@meme.hokudai.ac.jp

Kasper Hornbæk
Dept of Computer Science, University of Copenhagen,
Njalsgade 128-132, Bldg 24, 5th Fl
DK-2300 Copenhagen, Denmark
kash@diku.dk

ABSTRACT
The data resources and applications accessible through today's Web offer tremendous opportunities for exploration: ask a slightly different question, receive a correspondingly different answer. However, typical browser-based mechanisms for accessing the Web only enable users to pose one such question at a time, placing a heavy operational and cognitive burden on any user who wants to explore and compare alternatives. A subjunctive-interface approach may reduce this burden. Subjunctive interfaces support the setting up, viewing and adjustment of multiple scenarios in parallel, allowing side-by-side instead of temporally separated viewing, and more efficient iteration through alternatives. We have implemented an environment in which end users can set up custom-built Web-access interfaces that include such multi-scenario support. In this chapter we describe three modes of use of this environment – parallel retrieval, coordinated manipulation, and tentative composition – and explain how these may help to alleviate typical challenges in Web-based tasks. At the same time, we acknowledge that the increased scope for exploration made possible through this environment can itself present a form of cognitive burden to users, and we outline our plans for evaluating the impact of this effect.

Keywords
Subjunctive interfaces, exploratory information visualisation

INTRODUCTION
How inconvenient it is that so many applications for accessing Web resources only deliver results in response to explicit, pinpoint requests. For example, interfaces for flight enquiries typically require the user to specify exactly one destination city, which is fine for users with precisely formulated travel plans but a bore for everyone else. A user who wants to compare the deals and schedules available for a range of destinations must embark on an exploration, submitting a succession of requests and analysing their respective results.

One problem in such cases is that users have poor support for covering a range of requests, even if the details of those requests follow some regular pattern. A user searching for flights might request information for several routes on a given date, or for a single route over many dates, or combinations of a few routes and dates. But if the interface only supports the handling of a single request at a time, this burdens the user not only with a potentially high number of interface actions to specify and submit the requests, but also with increased mental effort in planning the requests, remembering which requests have been made so far, and remembering where interesting results were found.

Furthermore, one-at-a-time interfaces provide poor support for comparing results [18], in that making comparisons requires the user to remember – or to have written down, or to request again – the details of those results that are currently out of sight. This again can constitute both a physical and a mental burden. We believe that these burdens can be reduced by enabling the user to carry out a number of requests at the same time, which we refer to as parallel retrievals.

Consider now a second kind of Web interaction. A doctor who has access to her patients' records through a secure Web connection wants to retrieve images from specific stages in the treatment of a single patient, for example to observe progress of disease within an organ. On obtaining each abdominal image study she goes through the same operations of selecting and scaling the desired sub-portion of the images, in three orthogonal planes, then adjusting the greyscale mapping so as to emphasise the boundary of the diseased region, and finally selecting display of just the overlays containing her own annotations. She accumulates browser windows, one for each imaging study, so as to be able to switch between them to help grasp the disease's changes over time. If she finds that the diseased region has spread beyond the bounds of the focus area she selected for the earlier studies, she re-adjusts those earlier views so that she can still compare like with like.

In this situation, which could equally apply to retrieving and manipulating weather maps, or archived pictures from a Web cam, it is frustrating to have to perform the same image manipulations many times over, especially given the risk that information gained from later views will upset earlier decisions. This frustration could be alleviated if there were a way to manipulate the various retrieved resources in concert, continuously confirming that the results will be valid for all of them. This we refer to as coordinated manipulation.

As a third example, consider a holidaymaker who (having successfully selected some flights) is now installed in a foreign city and is planning a day of sightseeing. The city is served by a navigation-service Web site that provides estimates of point-to-point journey times on foot or by public transport. Having made a list of addresses he would like to visit, the visitor can use this site to plan an itinerary for the day.

The challenge here is to come up with a sequence of visits, and the journeys between them, that promises to be an interesting overall experience without excessive use of travel time or leg-power. If there were a strict constraint that all the listed sites be visited, a 'travelling salesman' algorithm could be put to work in search of a time-efficient total solution – or, perhaps, the conclusion that there is no way to visit them all within a day. However, all but the most ardent tourist would probably take a more relaxed approach, trying a few alternative visit orders and transport options, and being willing to exclude sites that turn out to be inconvenient to include. Nonetheless, this would be a frustrating task in the absence of support for what we call tentative composition – meaning, in this case, being able to compose and compare a number of alternative itineraries, involving different sites and/or different visit orders.

We believe that the above three kinds of challenge can all be addressed by offering users access to Web resources through subjunctive interfaces: interfaces that allow a user to explore alternatives by setting up multiple application scenarios at the same time, viewing those scenarios side by side, and manipulating them in parallel. In this chapter we report our investigations into using subjunctive interfaces for Web access.

We begin by describing the RecipeSheet, the platform that we have been using for our implementations. In the three following sections we then introduce three usage examples that address challenges of the kinds given above, and for each example discuss briefly its applicability to common modes of use of today's Web. After these examples we address a potential downside to this work: that in seeking to make it easier to pursue an exploration that brings a range of information to a user's screen, we may be counter-productively increasing the burden on the user who then has to evaluate that information. We outline our plans for investigating this issue as part of our ongoing work.

SUPPORTING MULTI-SCENARIO WEB ACCESS

Our recent work has been based on implementing Web access mechanisms for the RecipeSheet [11][12], a spreadsheet-inspired environment that has built-in subjunctive-interface features and therefore supports parallel exploration of alternative calculations and their results. Like a spreadsheet, the RecipeSheet supports the setup of flow-like calculations in terms of dependencies between cells. The subjunctive-interface features mean that the cells providing inputs at the start of a flow can hold multiple values simultaneously, the user can set up alternative scenarios based on chosen combinations of those values, and the cells holding derived values will then show the results for all scenarios, colour-coded and/or spatially arranged to help the user understand which result arose from which scenario.

A RecipeSheet user defines inter-cell dependencies in terms of so-called recipes. There is a set of standard recipes, such as for extracting particular tagged elements from a chunk of XML, but users are also expected to create their own. Recipes can be programmed directly in Smalltalk, Open Object Rexx, or XQuery; recipes capturing behaviour from Web applications can be built using the mechanisms of C3W [4], and Web-service recipes can be created with the help of SOAP or REST. In addition, the setup of cells and recipes on a sheet can be saved as a composite recipe, that can then be used on other sheets.

Clearly, only the kinds of Web access that can be coded as recipes – taking one or more input values as parameters, and providing one or more results – can be used within the RecipeSheet. The inputs and the results can be simple textual or numerical values, or of richer types including XML and HTML. Our early demonstrations of C3W showed how simple HTML Web applications can be captured in a suitable form for this calculation model. We also explained that while some Web-application results can conveniently be delivered as string or numerical values, allowing easy follow-on processing, for applications that present their results in carefully crafted displays it may be preferable to clip and show those complex displays as they are. As shown in [12], a cell containing such a display can still be used to supply inputs for further cell dependencies, for example by letting the user select desired HTML sub-elements directly within the cell.

A further property of the RecipeSheet is that the processing for a recipe is itself an ingredient – in other words, an input – that can be specified in a cell. The RecipeSheet can therefore provide uniform handling of variation in both inputs and processing, which seems a natural requirement in some forms of Web access. For example, whereas one user may want to view the results of sending alternative keyword queries to a single search engine, another might want to send the same query to multiple engines. On a recipe sheet both forms of variation are straightforward, as is dynamically switching between the two.

Given an environment in which multiple Web-access scenarios can be supported, the potential benefits to be gained depend on how such scenarios are created and used. Each of the situations in the Introduction can be helped by a subjunctive interface being used in a different way: to support parallel retrieval, coordinated manipulation, and tentative composition respectively. The following implementation examples illustrate these three modes of use.

PARALLEL RETRIEVAL

Parallel retrieval refers to enabling the user of a retrieval-style application, such as a flight enquiry site, to specify not just a single retrieval but several alternatives, differing in arbitrary ways, at the same time. These retrievals are handled in parallel as separate scenarios, and their results displayed in such a way that the user can see them all simultaneously, and can see which retrievals delivered which results.

Fig. 1 shows a sheet that has been set up to find related research articles. The results in the *relatedPapers* cell are marked up, and then sorted, according to which scenarios they appear in. On the sheet that appears in the background, we can see that even though the three queries are all based on papers about the same Elastic Windows project, just three results were found by all three queries whereas several results were found by one query only. Thus a user who had arbitrarily chosen just one paper as the starting point would have missed out on many results that could be relevant.

Many researchers are investigating how the quality of search results eventually chosen by users is affected by the number and the order of the results that are presented. Keane et al. [5] found that users of Google are biassed towards choosing items near the top of the result lists, even though the chief measure determining the result order (link popularity) is not a direct reflection of item quality. Pandey et al. [14], seeking to counteract the Entrenchment Problem whereby new Web pages, even if of high quality, score low in search-engine rankings and are therefore denied the top positions that would get them noticed, discuss methods for randomly boosting some papers' ranks so that they have more of a chance of being seen.

We suggest that presenting the merged results from multiple searches is another way to work around the bias of any individual search. In [10] we demonstrated how augmenting a Google search with a set of additional searches narrowed by date (e.g., by adding '1990..1999' to the search phrase) could bring to light items that, though coming at the very top of the results for their particular era, were drowned out of the top entries in a standard (non-date-narrowed) search. We also suggest that allowing users to see why each item is being offered – for example, that it appeared high in a 1990s search but not in any other – will help them to judge the item's relevance. Muramatsu and Pratt [13] made a call for this kind of 'transparency' in search engine results, to help users of search engines to understand – and to take control over, if they wish – the transformations (such as stop word removal or suffix expansion) that are applied automatically to their queries. Such

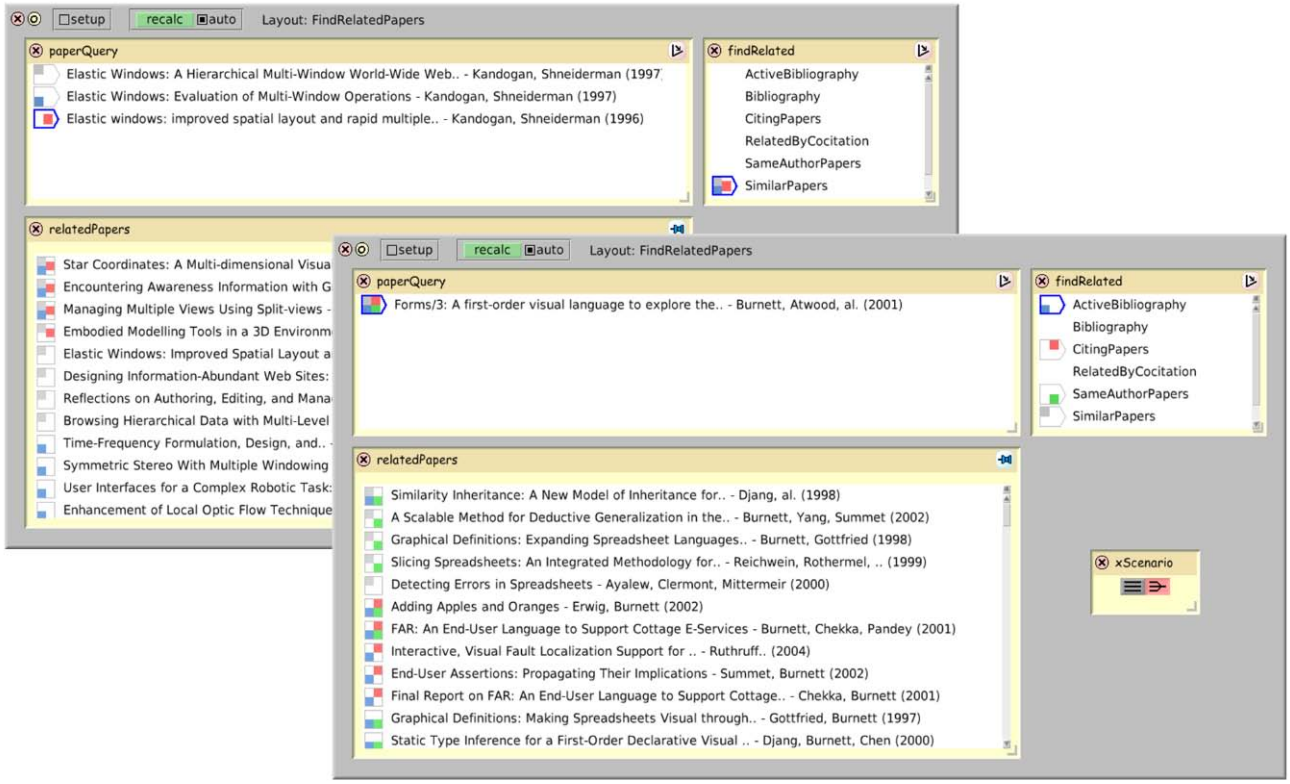

Fig. 1 Parallel Retrieval. Searching for academic articles, using mechanisms captured from the CiteSeer and DBLP Web sites. For an article specified in *paperQuery*, the sheet uses the recipe specified in *findRelated* to find related papers. The user has set up the sheet so that results from multiple scenarios are merged into a single list, with markup to show which scenarios each item appears in. In the sheet in the background the user has requested a `similar papers' retrieval for each of three articles from the same project; in the foreground, four alternative retrievals based on a single article.

transparency in result presentation has recently gained much attention; an extensive survey is found in [2].

However, it is far from clear how best to augment a search-result display to help the user understand where each result has come from. Dumais et al. [3], studying the impact of alternative formats for marking up results with automatically derived category information (e.g., distinguishing the various topics of pages retrieved by an ambiguous query such as “Jaguar”), found that users were much quicker at finding relevant items from lists divided according to category than from the complementary form of display in which category information was added to each item in a single list. For an application such as that shown in Fig. 1, where items typically belong to multiple scenarios (cf. a unique category), and where this multiple membership itself has meaning, the trade-off is likely to be less clear cut. In general we do not expect that any single presentation approach would be optimal for all parallel-retrieval situations; it depends too much on the nature of the information within each scenario, and the distinctions between scenarios. Our approach, therefore, is to give users the mechanisms they need to build multi-scenario interfaces for their own Web searches.

In any case, we believe that parallel retrievals are potentially valuable for a wide range of Web usage situations. In Kellar et al.’s [6] four-category classification of Web-based information-seeking tasks, we regard parallel retrieval as being relevant to at least Fact Finding and certain kinds of Transaction. Fact Finding is used to refer to short-lived tasks for locating specific pieces of information, while Transactions covers interaction with Web applications such as shopping sites, or email or blogging tools. The other two categories of information seeking – Information Gathering, and Browsing – are by their nature less structured, and therefore less likely to have the regularity that makes parallel retrieval practical.

The fact that some Transaction-style operations have side effects, such as making purchases, would set a context-specific boundary on the actions that most users would want to perform in parallel. Whereas it would be reasonable to enquire about room prices at several hotels for the same date, for example, it would be unusual then to proceed to book them all. On the other hand, if the user’s task happens to be to find a single hotel with a room available for each of several separate visits, proceeding to make a simultaneous booking for a set of enquiries (i.e., the various visits) might indeed make sense. Such an operation would fall within what we refer to as coordinated manipulation, as described in the next section.

COORDINATED MANIPULATION

By coordinated manipulation we mean having simultaneous control over several instances of an interactive application; in the introduction we gave the example of using this for browsing images. Within a subjunctive interface these application instances would typically reside within distinct scenarios created by the user.

Fig. 2 shows a RecipeSheet built for the European Union's integrated project ACGT, which is pursuing (among other things) the development of an ‘Oncosimulator’ that can reliably simulate

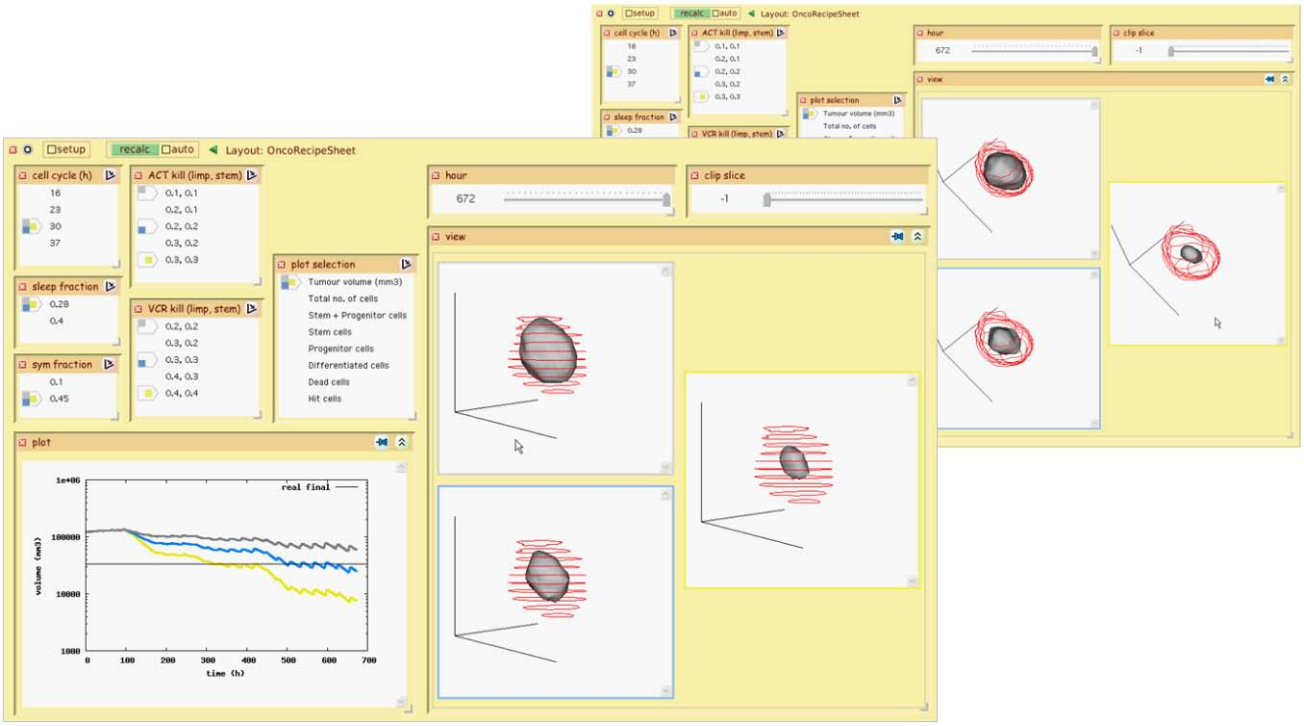

Fig. 2 Coordinated Manipulation. Two views of a sheet for exploring results from the ACGT Oncosimulator, a model for predicting the response of a patient-specific tumour to various forms of therapy. The five input cells at top left set the values for various simulation parameters. Here the user has set up three scenarios representing three levels of responsiveness to chemotherapy. In the large cell on the right, which shows an interactive 3D visualisation of the simulated tumour, user manipulation is mirrored across all scenarios; in the background we see the outcome of rotating about a horizontal axis.

cancer growth and treatment. The 3D visualisation on the right of the sheet supports a limited form of direct-manipulation interaction: by clicking and dragging with the mouse, a user can rotate a view about horizontal and vertical axes. When there are multiple scenarios, and hence multiple views, their orientations are synchronised by the RecipeSheet such that rotating any one view causes the others to rotate the same amount, as seen in the figure. Such synchronised interaction is a staple of recent developments in coordinated and multiple views [15], where it is recognised as a powerful technique for helping users to understand related data.

What is not readily apparent from the picture is that these views are in fact Web browsers, and the visualisations AJAX-enabled pages. This provides the scope for implementing coordination at various levels, potentially applicable to a wide range of applications. The simplest form of coordination involves mirroring operations at the level of individual mouse and keyboard events. This allows coordinated control of visualisations that, like the 3D view in the figure, give uniform responses for user actions at equivalent coordinates within the view. If one were to open a set of Google Maps pages on different locations, for example, the operations of panning, zooming and image selection could be mirrored at this level. Typing in a request for navigating from the map location to some other (common) location should also work, showing the different routes in the individual views. Where this simple approach would break down is if the user switches into a mode such as a city's Street View, where the interaction options available depend on one's precise location in the city.

A next level of coordination would be through identifying and mirroring logical events: abstracting combinations of mouse movements and clicks to make up events such as selecting a menu item, or highlighting the entity at some location within an HTML page's DOM tree. Going a level higher still, one could employ mechanisms such as those of Koala/Coscripter [9][7] to record and share operations in a way that would be robust even in the face of (some) differences in page layout.

Mirroring events at an abstract level therefore makes it possible to support not just manipulation of the objects within Web pages, but coordinated clicking of link anchors to navigate from one page to the next through matching regions of a Web site – for example, through standardised sets of pages relating to hotels on a travel site, or proteins on a bioinformatics site. Hence, as mentioned above, the possibility of querying a travel service to find a hotel that has a room available for each of several visits, then going through the booking procedure for all those visits together.

The above discussion shows how a given task can straddle the border between parallel retrieval and coordinated manipulation. Work by Teevan et al. [17] suggests that much directed search on the Web – that is, search for a target that is known in advance to exist – is carried out as a mixture of the basic elements that underlie the two. Teevan et al. distinguish between, on the one hand, teleporting, by which they mean jumping directly to a Web page found as the result of a query, and on the other hand orienteering, their term for localised, situated navigation that begins at a familiar starting point (such as a portal) and narrows in on the target. As noted before, the best we can do as interface designers is provide facilities for users to choose for themselves the mix of teleporting and orienteering, and the range of scenarios over which they wish to perform the two. For now we are investigating what facilities make sense for the user group developing and calibrating the ACGT Oncosimulator.

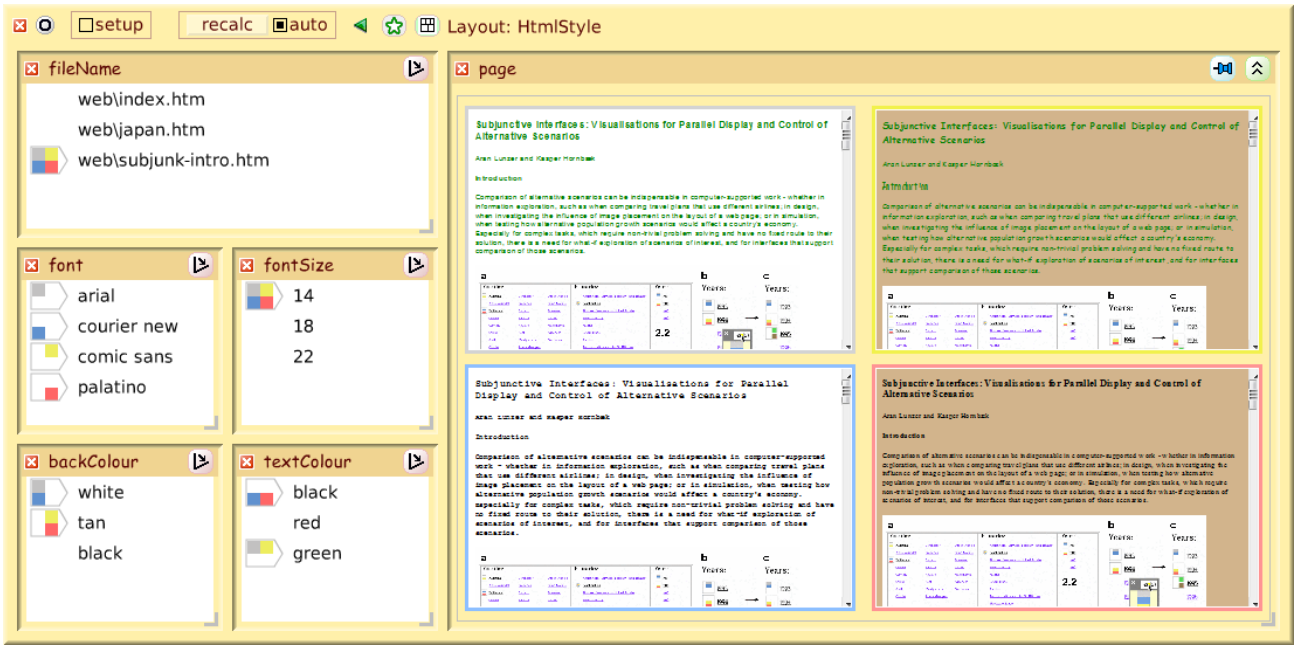

Fig. 3 Tentative Composition. In this case what is being composed is a rendered Web page, based on values supplied for the page content and for various style-defining parameters. The user has set up four alternative 'compositions', and can see at a glance differences between them, such as how the font style affects the amount of space needed to render a given paragraph.

TENTATIVE COMPOSITION

Some Web-based tasks can be characterised as the composition of multiple pieces of retrieved information, where it is the overall composed entity that serves the user's purpose, rather than the elements on their own. The example given in the Introduction, of building a sightseeing itinerary, is essentially a composition of the route recommendations returned by the navigation service in response to various point-to-point queries. Being able to experiment with and compare alternative compositions, such as in this case varying the sequence of locations to be visited, is what we refer to as tentative composition.

As with the preceding two modes of use, tentative composition covers a broad range of users' tasks. At the simple end of this range are tasks in which the 'elements' being combined are merely the values for placeholders within a structure that has been defined in advance: an everyday example would be the choice of starter, main course and dessert to make up a meal; using today's Web one might create an office party invitation by composing venue details, transport information, and a map. Cases such as these can be treated simply as parameterised retrievals, and therefore explored using parallel-retrieval mechanisms.

At the complex end of tentative-composition tasks are cases of general design with arbitrarily many degrees of freedom, such as the planning of a new building or of a multi-continent concert tour. For some of these complex domains there are already specialised applications that include support for exploring design alternatives, and we are not suggesting that generic mechanisms for handling multiple scenarios could provide a similar strength of support. We believe that subjunctive interfaces will make their mark on small-scale, ad hoc composition of Web resources.

Supporting tentative composition requires, first, providing a substrate on which compositions are built. Then there must be a convenient way for the user to specify alternatives, and supportive mechanisms for viewing the corresponding outcomes and understanding how they differ – either in terms of the final results, or the alternative specifications that led to them. The RecipeSheet, having been designed to work as a substrate for flow-based calculations based on values supplied in cells, is inherently suited to the simplest kinds of tentative composition which, as stated above, can be set up like parallel retrievals. Fig. 3 shows one such example, where the 'composition' being carried out is the building of a Web page, complete with style information.

Beyond these simple cases, the RecipeSheet's supportiveness depends on how the composition is defined as a calculation flow. The building of a sightseeing itinerary could be tackled in various ways: one possibility is to have a cell defining each sequential step in the itinerary (for example, one cell specifying the first visit address, a second cell specifying the second visit, and so on); another is to have a single cell in which the whole itinerary is specified as a list of addresses, and that lets the user specify different lists for different scenarios. The fact that the RecipeSheet makes it as easy to specify alternative processing as alternative parameter values would be useful in experimenting with alternative navigation services within these itineraries. However, we readily admit that both of the above approaches have potentially troublesome limitations: for example, the first would be highly inefficient for a user who wished to try adding or removing one visit from an existing itinerary, while the second would provide poor support for grasping rapidly how two or more itineraries differ.

While we are sure that the current design of the RecipeSheet is not the final answer in terms of supporting tentative composition in general, we believe its current level of support is sufficient to begin evaluation on exploratory tasks of this kind.

RISKS OF COGNITIVE OVERLOAD: THE PARADOX OF CHOICE

The Paradox of Choice is the title of a popular book by Barry Schwartz [16], in which he points out that although having some freedom to make choices in your life feels much better than having no choice at all, too much choice is a problem in its own right. People get stressed by the amount of mental effort involved in weighing up alternatives, by the worry that other, better alternatives are somewhere out there to be found, and, after making a choice, by the fear that on balance one of the rejected options might have been better.

Given that subjunctive interfaces are intended to improve the quality of information users receive by encouraging them to request and view more alternatives, Schwartz's studies undeniably suggest that we might be doing our users more harm than good. Especially given the vast amount of information available over the Web, it can be argued that what users desperately need is more filtering, not more retrievals.

However, we feel that the current popular approach to helping users make sense of the Web – namely, using some hidden ranking or other heuristics to deliver a small, possibly high-quality but necessarily biassed selection of results – is asking users to put too much trust in online systems. There is some evidence that users are alert to this: for example, Lin et al. [8] found, in a study of users' attitudes to question-answering systems, a tendency to feel uncomfortable accepting answers from systems that provided only the bare answer. The users wanted to see some context surrounding the answer, to help them confirm its legitimacy.

Nonetheless, there are also plenty of studies showing that giving users too much to do is counter-productive. Beaulieu and Jones [1] discuss the interrelationship between the visibility of system functions, the balance of control between user and system, and the user's cognitive loading. They found that a relevance-feedback retrieval interface that was designed to keep users in control of their queries, by revealing the details of the feedback-derived query terms and requiring the users to review and adjust those terms, in fact caused users to play a less active role; making the adjustments would have been just too much work. Muramatsu and Pratt [13], who concluded from their study of transparent queries (mentioned earlier) that perhaps the best style of interface would be a 'penetrable' interface – one that lets the user know what has been done, and also provides a chance to change it – made a point of adding the caveat that providing too much control could inadvertently overload the user.

Part of the issue, as Beaulieu and Jones note, is that users need to feel that the decisions available to them are relevant to their personal information needs, rather than being just artefacts of the interface. For our own goals of deploying end-user programming and customisation techniques that help users to express a range of directions to investigate, and then to make sense of the corresponding range of results, we must strive to ensure that users will perceive this effort as part of what they wanted to do anyway. If we can achieve that, there is hope that users will regard the ability to set up and work with multiple scenarios as a welcome level of choice, rather than an unwelcome source of stress.

STATUS AND PLANS

In this chapter draft we have outlined the role we believe subjunctive-interface mechanisms can play in supporting users' access to Web resources. In particular, we have identified three kinds of challenge in Web-based interaction for which subjunctive interfaces appear to be useful. We regard all three as lightweight instances of end user programming, given that the user is exploiting interface facilities to build a personal, customised view of available information.

Over the coming months we plan to run various studies to obtain evidence about the usability and effectiveness of the techniques described here; our findings will be added to the second draft, for inclusion in the final version. One study will be based on the use of the RecipeSheet in support of the ACGT Oncosimulator, where we hope to see users readily applying the scenario-management facilities to request and gather results of simulations under meaningful collections of alternative conditions, and also making appropriate use of the coordinated-manipulation facilities in exploring the corresponding results. In another study we shall investigate users' understanding of and preferences regarding the merged presentation of parallel retrievals, seeking to build on the context-presentation findings of [3] and follow-on work. Finally, we hope to be able to provide some quantification of the Paradox of Choice effects that the introduction of multi-scenario facilities seems likely to induce.

REFERENCES

[1] Beaulieu, M. and Jones, S. (1998). Interactive searching and interface issues in the Okapi best match probabilistic retrieval system. Interacting with Computers, 10(3), 237-248.

[2] Cramer, H., Evers, V., Van Someren, M., Ramlal, S., Rutledge, L., Stash, N., Aroyo, L. and Wielinga, B. (2008) The effects of transparency on trust and acceptance in interaction with a content-based art recommender. User Modeling and User-Adapted Interaction, 5.

[3] Dumais, S., Cutrell, E., and Chen, H. (2001). Optimizing search by showing results in context. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Seattle, Washington, United States). CHI '01. ACM, New York, NY, 277-284. DOI= http://doi.acm.org/10.1145/365024.365116

[4] Fujima, J., Lunzer, A., Hornbæk, K. and Tanaka, Y. (2004) Clip, Connect, Clone: Combining Application Elements to Build Custom Interfaces for Information Access. In Proceedings of ACM UIST 2004, October; Santa Fe, NM, 175-184.

[5] Keane, M. T., O'Brien, M., and Smyth, B. (2008). Are people biased in their use of search engines?. Commun. ACM 51, 2 (Feb. 2008), 49-52.

[6] Kellar, M., Watters, C. and Shepherd, M. (2007). A field study characterizing Web-based information-seeking tasks. J. Am. Soc. Inf. Sci. Technol. 58, 7 (May. 2007), 999-1018. DOI= http://dx.doi.org/10.1002/asi.v58:7

[7] Leshed, G., Haber, E. M., Matthews, T. and Lau, T. (2008). CoScripter: automating and sharing how-to knowledge in the enterprise. In Proceedings of the Twenty-Sixth Annual SIGCHI Conference on Human Factors in Computing Systems (Florence, Italy, April 05 - 10, 2008). CHI '08. ACM, New York, NY, 1719-1728. DOI= http://doi.acm.org/10.1145/1357054.1357323

[8] Lin, J., Quan, D., Sinha, V., Bakshi, K., Huynh, D., Katz, B. and Karger, D. R. (2003). What Makes a Good Answer? The Role of Context in Question Answering. Proceedings of the Ninth IFIP TC13 International Conference on Human-

Computer Interaction (INTERACT 2003), September 2003, Zurich, Switzerland, 25-32.

[9] Little, G., Lau, T. A., Cypher, A., Lin, J., Haber, E. M. and Kandogan, E. (2007). Koala: capture, share, automate, personalize business processes on the web. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (San Jose, California, USA, April 28 - May 03, 2007). CHI '07. ACM, New York, NY, 943-946. DOI= http://doi.acm.org/10.1145/1240624.1240767

[10] Lunzer, A. (in press). Using Subjunctive Interfaces to Put Delivered Information into Context. Knowledge Media Science: Preparing the Ground. K.P. Jantke, R. Kaschek, N. Spyratos and Y. Tanaka, Eds. Lecture Notes in Artificial Intelligence, vol. 4980, in press.

[11] Lunzer, A. and Hornbæk, K. (2006). An Enhanced Spreadsheet Supporting Calculation-Structure Variants, and its Application to Web-Based Processing. In K.-P. Jantke, A. Lunzer, N. Spyratos and Y. Tanaka (eds.) Proceedings of the Dagstuhl Workshop on Federation over the Web, Dagstuhl Castle, Germany, May 2005 (Lecture Notes in Artificial Intelligence, Vol. 3847 - 2006), 143-158.

[12] Lunzer, A. and Hornbæk, K. (2006). RecipeSheet: Creating, Combining and Controlling Information Processors. In Proceedings of the 19th Annual ACM Symposium on User interface Software and Technology (UIST '06), Montreux, Switzerland, Oct 2006, 145-153.

[13] Muramatsu, J. and Pratt, W. (2001). Transparent Queries: investigation users' mental models of search engines. In Proceedings of the 24th Annual international ACM SIGIR Conference on Research and Development in information Retrieval (New Orleans, Louisiana, United States). SIGIR '01. ACM, New York, NY, 217-224. DOI= http://doi.acm.org/10.1145/383952.383991

[14] Pandey, S., Roy, S., Olston, C., Cho, J. and Chakrabarti, S. (2005). Shuffling a stacked deck: the case for partially randomized ranking of search engine results. In Proceedings of the 31st international Conference on Very Large Data Bases (Trondheim, Norway, August 30 - September 02, 2005). Very Large Data Bases. VLDB Endowment, 781-792.

[15] Roberts, J. C. (2007). State of the Art: Coordinated \& Multiple Views in Exploratory Visualization. In Proceedings of the Fifth international Conference on Coordinated and Multiple Views in Exploratory Visualization (July 02 - 02, 2007). CMV. IEEE Computer Society, Washington, DC, 61-71. DOI= http://dx.doi.org/10.1109/CMV.2007.20

[16] Schwartz, B. (2004). The Paradox of Choice: Why More is Less. Harper Perennial. ISBN: 0-06-000569-6

[17] Teevan, J., Alvarado, C., Ackerman, M. S., and Karger, D. R. (2004). The perfect search engine is not enough: a study of orienteering behavior in directed search. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (Vienna, Austria, April 24 - 29, 2004). CHI '04. ACM, New York, NY, 415-422. DOI= http://doi.acm.org/10.1145/985692.985745

[18] Terry, M. and Mynatt, E.D. (2005). Enhancing general-purpose tools with multi-state previewing capabilities. Knowledge-Based Systems, 18, 2005. 415-425.