A Complementary Approach for Adaptive and Adaptable Hypermedia: Intensional Hypertext

William W. Wadge\(^1\) and Monica M.C. Schraefel\(^2\)

\(^1\) Department of Computer Science
University of Victoria
Victoria, B.C., Canada
wwadge@csr.uvic.ca

\(^2\) Department of Computer Science, University of Toronto
Toronto, ON, Canada M5S 3G4
mc@dgp.toronto.edu

Abstract. We describe a methodology and an authoring/publishing tool for adaptable and/or adaptive Web documents. Our approach is based on intensional logic, the logic of assertions and expressions, which vary over a collection of contexts or possible worlds. In our approach the contexts are sets of values for parameters which specify the current user profile as supplied by the current Web page URL, and the latest user input. The author produces generic (multi-version) source in the form of HTML with extra markup delimiting parts that are sensitive (in various ways) to the parameters. This source (in what we call Intensional Markup Language) is translated into program in a Perl-like language called ISE (Intensional Sequential Evaluator). To generate the appropriately adapted individual pages, the server runs the ISE program in the appropriate context. The program produces HTML that, when displayed in the user’s browser, is rendered into the desired adaptation of the requested page. Although this intensional approach was originally designed to work without any explicit user model, we can extend it (and make the documents adaptive as well as adaptable) simply by incorporating a user model that monitors the user and computes some of the profile parameters.

1 General Adaptive/Adaptable Architecture and the Role of Parameters

The goal of adaptive and adaptable hypertext\(^3\) is in many respects to accessing the right information easier for users, but at the same time the mechanisms to

\(^3\) By adaptive hypermedia, we refer to hypermedia systems which primarily rely on a user model to support the delivery of user-specific content (see, for instance, [1]-[5]). By adaptable, we mean that users select from a variety of parameters to adapt the hypertext to their needs. A stateless version of adaptable hypertext can be created in client-side JavaScript, for instance, in which users can change the color of the background of a page dynamically. The server side approach described here can maintain the state of such a change across any other page selections made on that site.

© Springer-Verlag Berlin Heidelberg 2002
support this goal can greatly complicate the task for the content creators and publishers. In general there are many different possible states (configurations) of a user model - many different possible user profiles. In adaptable hypermedia, there are likewise multiple possible instances (or versions) of a domain object, such as the large color view of an image with English captions or the small, black and white one with French captions. The publishing system is therefore responsible in each case for delivering a whole family of variants of a document (one for each possible profile or domain instance), rather than one single monolithic document. It becomes necessary to develop a system that can render these families in an efficient way, which will avoid overloading either the system, or the content creators.

In the following sections of this paper, we describe a methodology and an authoring/publishing tool which can be understood as the result of using intensional logic to formalize these general observations about adaptive/adaptable architecture. We hope by this to provide a general front-end or author-based tool, which will facilitate the authoring of versionable, web-based hypermedia which can then be supported by either adaptive, adaptable or both kinds of inputs.

2 The Intensional Approach

Our approach is based on intensional logic [10], a natural choice for versioning, since it is the logic of assertions and expressions which vary over a collection of contexts or possible worlds. In our approach the contexts are sets of values for parameters which specify the user profile (if one exists), current page, and user input.

The intensional (possible-worlds) approach to versioning was originally developed by Plaice and Wadge [6] for use in configuring families of programs from families of components. Most software configuration tools work bottom-up, and allow the user to create a variant of the program by selecting different variants of the components. In intensional versioning, by contrast, each variant is described/determined by a “version expression” assigning values to parameters. For example, the expression

\[ \text{processor:ppc} + \text{OS:8} + \text{language:french} \]

might indicate the French language version for a PPC Macintosh running OS 8.

A particular version of a program is configured, as usual, by assembling particular versions of components. In intensional versioning, however, the component choices are not arbitrary; instead, they are determined by the expression for the version requested. For this to work it is necessary that each variant of each component be labeled with a version expression. In the simplest case, when configuring version V of a program, we use version V of each component. However in general we do not require that each component have a version whose label is exactly V. If there is no such component, we are allowed to choose a