An Intensional Tool Applied to French Language Educational Software*

Honglian (Elena) LiWilliam W. Wadge

Abstract

We present a Web-based French-language educational-software tool, French e-Flash Card (FFC), authored using our Intensional Markup Language (IML). The FFC site is interactive and dynamic and incorporates AI for French language grammar. These special features, lacking in most language educational tools (which are typically language-independent), are made possible by IML, which allows rule-based markup in an intensional (possible worlds) logic. The IML source files, relatively small, define whole families of pages indexed by parameters which specify (for example) the particular subject matter, the degree of difficulty, even the choice of vocabulary and agreement constraints on French example sentences generated on demand. We describe the FFC site, give an overview of IML, and describe our design methodology, which should be applicable to many similar projects.

1 Introduction

This article presents three related topics: the French language educational software known as French e-Flash Card (FFC), the design approach used in creating FFC, and the authoring tool employed, Intensional Markup Language.

The French e-Flash Card was designed and implemented by the authors. It is based in part on Wadge and Wadge’s original French Sentence Maker program [6]. The French Sentence Maker was programmed directly in ISE, a low-level Perl-like language with intensional features. FFC is a more ambitious project made possible by IML, which is not a programming language but rather a high-level markup language. IML is a rule-based extension of HTML which allows large chunks of ISE to be replaced by simple macro calls. It makes it much easier to produce the source for parameterized sentences and for tables, popup menus, search forms and the like.

FFC is in some ways a typical logic-based Web application; it has a knowledge base, a logic engine, user-selectable input parameters and a presentation layer for answers. It is an intelligent way to present the logical relations and rules underlying French grammar. It goes beyond most conventional Web-based software in that it is not just a visual interface to a database. Since the FFC rules embody knowledge of French grammar, it is capable of answering student questions and displaying example utterances that have not been explicitly entered in a database. Furthermore, the intensional logic behind IML supports an inheritance relation between contexts. This makes it easy to express the rules in terms of general principles and exceptions (and exceptions to exceptions, and so on). We provide a detailed description of the usage of the tool. We also investigate the difference between IML and the conventional Web-authoring tools.

We describe the problem and our motivation in §2; the design approach is explicated in §3; we explain the use of our intensional programming tool—IML in §4; intensional logic and intensional programming methodology as its foundation are also defined in that section; we illustrate the implementation of FFC in §5; we conclude our work in the last section.

2 Problem Definition

Current French class-based education mainly sees French-grammar knowledge as built up from small, simple facts; for example, the form a particular adjective takes in the feminine plural, or conjugated form of a particular verb in a particular person, number and tense. Students are typically taught and tested on their knowledge of these atomic facts. However, even simple French expressions use a surprisingly large number of these atomic facts. Therefore, beginning students of French often get frustrated when they attempt to use what they are learning in class. Some students cannot even make well-formed sentences in a simple conversation.

Students need training in the basic facts of grammar but this knowledge alone in practice is not enough to allow them to easily produce complete sentences. They need to see plenty of examples of complete sentences.

There are quite a few French-grammar Web sites or applications available nowadays. In reviewing them, we found that most of them still employ the class-based educational model. They usually move the contents of the French grammar textbooks online and include some exercises that are also based on the questions in particular grammatical units. These Web sites are just a simply electronic version of the grammar textbook. They are not really practical for students who try to use their learning.

These electronic textbooks have the same problem as their paper-based counterparts: they provide only a limited supply of examples of complete sentences, namely the ones explicitly written, one by one, by the authors. There may be hundreds of them, but a student who spends even one week in France, immersed in French, will hear or read thousands and thousands. The electronic versions of the textbooks often have AI, which automates the production of atomic grammatical facts, but not that of complete utterences or even complete phrases.

FFC is a modest first effort to remedy this deficiency. The (still relatively simple) AI in FFC automates the production of complete phrases and sentences, as well as that of atomic facts. French e-Flash Card tries to provide as many sentences as possible in each grammatical unit, in order to present typical uses of the grammar facts. For example, in the possessive adjective section, the author built a set of examples that cover all possible uses of possessive adjectives: singular, plural, masculine and feminine. Of special importance is the fact that the student can guide the generation of the example sentences by choosing the vocabulary employed, so that, for example, she can see the difference between saying (in French) “the house is big”, “the car is big”, “the houses are big”, “the cars are big”, and so on. The student will discover, for example, that the French word for “big” has a different form in each of these four examples.

In designing and implementing this sentence-making function, it is necessary to solve the following technical problems:

- How to make sure the user who has only limited knowledge of French grammar and vocabulary can make not only grammatically correct but also semantically meaningful sentences.
- How to provide for the customizability of the example sentence.
- How to provide dynamic examples to meet the user’s specific requirement and improve interaction between the user and the software.
- How to arrange the contents in a Web interface in order to contain such a large quantity of example sentences.

3 FFC Design Approach

FFC offers the student a choice for each of the major elements of a sentence, such as subjects, predicates and objects. The students choose appropriate vocabulary items which they want to use in the sentence, without knowing the exact form these items will take in the complete sentence. Then FFC itself will handle the grammar rules interwoven in the sentence and generate the sentence for the user. The user can choose different vocabulary items or move to different
grammatical sections to repeat the sentence-making process. The student can learn grammar by making sentences whose meaning she already knows but whose exact form she may not yet be able to produce. The FFC AI handles not only the rules for atomic facts but also those (e.g., for subject-verb agreement or word order) needed to combine the atoms into a complete sentence. FFC can therefore answer questions of the type “but how do you say . . . ” which normally require the presence of a native speaker.

In order to improve the customizability and interactivity, FFC uses the parametric method to provide the user with a customized version of a Web page. These sentence elements are listed in a table or in a menu. It allows the user to customize his or her sentences with appropriate vocabulary items. Each option in a menu or each item in a table is an intensional link which consists of a version dimension and its version value. Once it is selected by clicking the mouse, the current version expressions will be changed. These different customized versions of the pages are not pre-stored on the server side; instead, they are generated on demand by the same sort of parametric (intensional) logic that the IML implementation uses to generate the French sentences. Figure 1 shows the architecture of the FFC.

4 Intensional Tool

The Intensional Markup Language is the front-end authoring tool to implement FFC. It is one of the intensional programming languages which are developed on the basis of intensional logic by Wadge [5].

4.1 Intensional Logic

Intensional logic is the underpinning of intensional programming. “Intensional logic is therefore the logic of expressions in which the intension of sub-expressions (and not just extensions) have to be taken into account” [4]. It is very common in natural language. For instance, we cannot judge “It is raining” true or false unless we know the context such as when and where. Entities which vary according to context are called intensions, and each particular value determined by a particular context is called an extension.

Contexts are also called possible worlds, attributed to Leibniz by Chellas [1], Honderich [2] and others. Many variants of intensional logic have been described using possible worlds semantics, such as assignment of truth or false to a statement. For FFC, the set of possible worlds is a set of possible versions of a sentence, a whole page, or a whole site. In the simplest case, a version is a set of values for parameters. The logic used by FFC has a refinement relation $\sqsubseteq$.

The fact that $A \sqsubseteq B$ means that the version $B$ is a refinement of version $A$, which in turn (in the simplest case) means that $B$ refines or is consistent with $A$, but assigns values to extra parameters not specified by $A$. In the logic used by IML, $\sqsubseteq$ is a reverse inheritance relation. This means that version $B$ of a page defaults to version $A$ of that page, unless there the author provides explicit information about version $B$ which overrides this default.
4.2 Intensional Programming

Intensional programming is programming using intensional logic. The obvious characteristics of intensional programming are using intensional operators to manipulate different versions.

The first intensional programming language was Lucid, invented by W. W. Wadge and E. A. Ashcroft in 1974 [4]. Its possible worlds form a programmer-defined multidimensional space, where each dimension is defined as non-negative integer. It has several operators, allowing access to the value at the next index in a dimension, the previous index, all indices meeting some criteria. There is no direct connection between Lucid and IML; in particular, Lucid did not have a refinement relation between contexts. However they share the same fundamental system which is demand-driven multidimensional processing.

4.3 Intensional Markup Language (IML)

IML’s backbone is ISE, a Perl-like CGI language with run-time parameterization [5]. An IML package is a collection of Groff macro definitions. It conceals the complexity of ISE from the general Web authors by using Groff macro definitions instead of the intensional tags of ISE [3]. When all these tags are replaced by their definitions, the result is an ISE program.

It is ISE which implements the version space and refinement operation. In particular, ISE has a special case statement (the \texttt{vswitch}) whose alternatives are labelled with versions. On execution it chooses the alternative whose label is closest (in terms of refinement) to the current context. The \texttt{vswitch} with its \textit{best-fit} principle is the main feature which implements the default/override logic vital to the construction of both example sentences and customized pages.

![Diagram of IML Publishing Mode](image_url)
IML itself is a simple markup language that extends HTML. It is constructed on top of ISE [3]. The IML source is translated (once) into a corresponding ISE program which, when run with specific parameter settings, produces the HTML which renders to the desired version of the requested page [5]. Figure 2 shows the publishing process from IML to ISE and then HTML.

IML utilizes the parametric approach. It makes it possible for many members of a Web site community to share the pages by sharing parameters. Furthermore, it makes customization of Web pages possible by altering the parameters. For instance, if the user visits the noun section of FFC, then the URL looks like this:

http://i.csc.uvic.ca/elenali/FFC/new/ffc.ise<top:noun>

The former part is regular URL. The ending <top:noun> is the parameter expression which presents the user’s specific requirement of the noun section of FFC.

We make heavy use of the IML autolink construct to change the current values of parameters. As described before, different versions of a Web page family are presented and invoked with different version parameters. Therefore, automatic links can be used for inter-version swaps. The format of an autolink is:

```plaintext
.balink version_dimension:version_value
anchor_name
.ealink
```

The phrase in italic typeface is user-definable. An autolink consists of a pair of tags .balink (begin a link) and .ealink (end a link) as well as the anchor name. A version expression, which consists of a version dimension and its value, follows the tag .balink. Once the user clicks the autolink anchor, the version expression is sent with the original URL to the Web-server reasoning engine. Then the best-fit version of the Web page is calculated and sent to the user. It is called an autolink because it is a link to the same page, but to a different version.

The FFC knowledge base is represented mainly using the IML .gmod and .gcase macros, which allow the author to specify rules for altering or adding parameter values in the current context. Naturally, the rules are interpreted using best-fit logic.

```plaintext
.bgmod
  .gcase current_version:value set_version:value
  .egmod
```

If the current version dimension is null, then the set version dimension will be the vanilla version expression. Also, multiple gcase phrases can be nested.

The following code in the “possessive adjective” section of FFC computes the gender of the nouns. The first rule specifies the default gender as masculine, while the other three override this default for the words maison, chaise and voiture. For example, when the user selects the noun maison by clicking the link which changes the current version as nom:maison, this block of code returns the gender version expression as gen:e, which explicates that maison is a feminine word in French. Figure 3 is the implementation screenshot.

```plaintext
.bgmod
  .gcase "": gen:-
  .gcase nom:maison gen:e
  .gcase nom:chaise gen:e
  .gcase nom:voiture gen:e
  .egmod
```

FFC is a comprehensive result of the logic programming approach and the IML tool. It is an innovative test of the intensional programming approach and tool.
5 FFC Implementation

FFC, is a typical application of intensional programming. FFC contains a family of pages with thousands of versions, like the noun section which has gender, number and article sub-sections. Each of these grammatical sections is defined as one of several versions of Web pages of the noun section. Although, their grammatical parameters have different values as gender, number and article, they share the same source of graphics and other attributes of a page, such as top menus or footnotes. Therefore, the entire Web site can be considered as versions of a single Web page and related members can share source. The reasoning engine, as a version control system, allows code sharing between versions.

In FFC each composition of these elements of sentences is in fact a specific version of the Web-page family. Once the user changes one of these element choices, FFC generates a new version of the Web page which presents the new example for the user. Therefore, FFC is a typical multiversion Web application which has a concise layout and clear presentation.

5.1 Data Structure

The data structure of FFC is a “tree” structure. There are a total of four levels of files. The file in the top level is ffc.m. The second-level files include JavaScript menu files topic.menu.array.js, the layout files including topicM.i and topicR.i. The third-level files are the example sentence generators for each subtopic, e.g., enR.i file is used to create examples for pronoun en. The lowest level files are generic macrocodes, e.g., noun-analysis macro definition file announ.i.

5.2 Functions

5.2.1 Number-alternative menu

The options in a series of menus can alternate between singular form and plural form. The change is decided by the current context. For example, the possessive-adjective example generator employs the alternative menu. There are three menus: object menu, possessive-adjective menu and the number menu, which controls the object in its plural or singular format. Any of these three menus can interact with the others. For instance, if the user chooses the plural option from the third menu, the options in the object menu and possessive adjective menu will change into plural form.

5.2.2 Vocabulary-alternative menu

The options in vocabulary-alternative menus can be changed into completely different vocabularies according to the previous selections of the user.

For example, the example sentence generator of gender.m in the French noun topic employs the vocabulary-alternative menu. The user can choose verb and object from the verb menu and object menu respectively. Each verb matches with a different set of objects. Once the user selects one verb, then the corresponding set of options for object will be presented in the object menu. What is more, the verb menu will present the corresponding conjugated verb according to the selection of subject.

5.2.3 Verb Conjugation Search Engine

FFC allows the user to look up the conjugation of the verb in the present tense in the Verb Conjugation Search Engine. The user can input his or her inquiries into the blank, and FFC takes the inquiry and searches for the conjugation form.
The search engine works in the following way:

First, it catches the user’s input and stores the string as a version dimension.

Second, the verb analysis process is invoked to cut the string into single letters and store the last two to six letters into a word-endings array, and the other letters into word a root array.

Third, FFC analyzes the verb ending. It depends on the grammatical rules of French verb conjugation.

Fourth, FFC goes through the irregular verb conjugation database to check whether the verb is an irregular verb. The server side will use the best-fit algorithm to facilitate the search. If the verb falls into one of the irregular conjugation categories, it will be conjugated according to the specific rule. If the verb is just a regular verb, then the default (normal) conjugation rule will be applied.

Fifth, the conjugated verb ending will be added to the verb root in order to compose the conjugation verb.

The last step is that FFC returns the result.

Let us see two examples. First, the regular verb “aimer” will be separated into two parts, the verb root “aim” and the verb suffix “er”. After checking through the irregular verb database, there is no record for “aimer”. Therefore, it is automatically defined as a regular verb. The regular conjugation rule is applied to “aimer”. Finally, FFC returns the result to the user.

The second example shows the irregular verb “manger”. Verbs ending in “-ger” are irregular verbs which are defined in the irregular verb database. So FFC will conjugate the verb according to the specific rule and return the result to the user.

5.3 Comparison

Compared with current French-language educational Web applications, FFC has the following advantages:

- FFC breaks through the limitations of the traditional model of conventional language Web applications, which is Web interface plus database. In contrast, FFC allows students to immerse themselves in a huge supply of example sentences.
- FFC has dynamic contents and layouts. In addition, it supports customization.
- It has many more examples which can be produced with the different options selected by users themselves.
- The user can interact with the application.
- FFC employs all kinds of changeable menus, which can change the options according to the context. Moreover, it facilitates a semantic filtering out function. It is an artificial intelligence application.
- Because the sentences change according to different selections chosen by the student, the student is rarely able to go through all the examples at one time. So the longer the student uses the application, the more contents and examples they can exploit from the application. It allows students to generate new sentences and examples, thus keeps their interest longer. Therefore, FFC promotes an active learning curve.
- FFC is easy to maintain and be reused.

The methodology that we used to create this multiversion French educational application is an authoring formalism in which one can specify whole families of related pages with only modest additional effort to that required to specify a single page. After setting up the grammar rule for some cases, it can produce sentences after taking input from users. This is the AI built into FFC, which is the distinctive characteristic of FFC.
6 Conclusion

The objective of this paper is to propose an innovative application of intensional programming in educational language software. The objective is achieved through the development of the intensional multiversion Web application French e-Flash Card. We presented FFC in terms of prototype design, intensional Web engineering approach and implementation. In the process of describing the development of FFC, we also explained the use of the intensional programming tool Intensional Markup Language. Moreover, the advantages of IML tool were discussed through the comparison with other intensional programming tools and conventional Web authoring tools. What’s more, we demonstrated the methodology of developing multidimensional Web application through the developing of the application FFC.

References


