Protege2FloraTab: a plug-in for the interoperability between Protégé and FLORA2

Calabrese A.[1], Masecchia S.[2], Mele F.[1]

[1] Istituto di Cibernetica, Consiglio Nazionale delle Ricerche, Via dei Campi Flegrei 34 Pozzuoli NA
{a.calabrese, f.mele}@cib.na.cnr.it

[2] Università degli Studi di Napoli "Federico II" – Facoltà di Scienze MM.FF.NN
s.masecchia@studenti.unina.it

1. INTRODUCTION

In this paper we present the *Protege2FloraTab* plug-in, a tool that allows to translate in FLORA2 (and *vice versa*) an ontology defined in Protégé. This tool allows a Protégé user to construct its ontology through the Protégé graphical interface and convert the defined ontology in FLORA2. If necessary, he can also complete its knowledge base through the FLORA2 definition rules. A Protégé user has, even, the possibility to perform queries in FLORA2 in order to test its ontology and its associated inferential apparatus, and has also the possibility to access the FLORA2 code. Likewise, a FLORA2 user has the possibility to develop its ontology using Protégé graphical interface and its large number of plug-ins.

2. PROTÉGÉ AND FLORA2 INTEROPERABILITY

The conversion of an ontology from Protégé to FLORA2 and *vice versa* does not consist in a simple syntactical re-writing of rules. In fact, although Protégé and FLORA2 are both frame based, their formal apparatus is different and this causes problems when the problem to find precise correspondences between the two systems is approached. In this work we considered this important issue and we built a semantic interoperability (although partial) between Protégé and FLORA2 that is achieved through the following three phases:

1. selection of all the operations having the same meaning both in Protégé and in FLORA2;
2. selection among the exiting Protégé operations that do not have correspondent operations FLORA2 (and *vice versa*) of operation that can be implemented as possible functional extensions of Protégé or FLORA2;
3. identification of operations or structural entities that cannot be added as extensions in Protégé or FLORA2.

Protégé and FLORA2 operations for the creation of structural entities (a class, a slot, etc) can be considered as the application of a command (in Protégé through the graphical interface, in FLORA2 through command lines) that produces an internal representation. Protégé and FLORA2 have an high number of common structural operations having the same meaning (intrinsic interoperability). In fact they share the main concepts of frame based representations. The main shared operations are: creation of taxonomical relations among classes, definition of slots with the related types and creation of instances of classes. In the implementation of the *Protege2FloraTab* plug-in we preserved the natural correspondence of structural operations between Protégé and FLORA2. We also considered interoperability issues (not intrinsic interoperability) relating to those structural operations of Protégé that do not have a straight correspondence in FLORA2. In TABLE 1 we report the correspondences between the types of slot of Protégé that have a straight interpretation in FLORA2 (intrinsic interoperability), those that do not have a straight interpretation in FLORA2, but that can be made interoperable through an extension (not intrinsic interoperability) and, at last, those for which is not possible to establish a correspondence (not interoperable).

TABLE 1
ANALYSIS OF TYPES FOR THE INTEROPERABILITY BETWEEN PROTÉGÉ AND FLORA2

PROTÉGÉ TYPE	FLORA2 TYPE	RESULT OF THE ANALYSIS
Any	Not representable	Not interoperable
Boolean	Adding a new class: boolean::symbol. 'true':boolean. 'false':boolean.	Extension required
Class	Not representable	Not interoperable
Float	Built-in class Integer	Intrinsic interoperability
Instance	Class name	Intrinsic interoperability
Integer	Built-in class Integer	Intrinsic interoperability
String	Adding a new class and a new rule: string. _L:string :- is_charlist(_L)@prolog().	Extension required
Symbol	For each Protégé slot (named slotName) with values [val1, ..., valN] a new class will be added: slotNameSymbol::symbol. 'val1':slotNameSymbol. ..... 'valN':slotNameSymbol.	Extension required

The Integer and Float types of Protégé and FLORA2 are intrinsically interoperables given that they have the same interpretation in both systems. The same is true for the type Instance of Protégé in that is possible in FLORA2 to define a class as the type of a slot, and this means that the domain of values of that slot corresponds to the set of the instances of the referred class. On the other hand, for the types Any and Class does not exist a correspondent interpretation in FLORA2 and so is not possible to extend the system in order to include these typologies of slot. In the case of the type Any, this limitation is due to the different management of the overriding of the slot in the two systems. In Protégé, if a slot is declared of type Any, it is possible to associate a value to it only performing a type overriding. Instead in FLORA2 the structural inheritance is monotonic therefore for slot signatures is only permitted the overloading. In the case of the type Class, the limitation is simply associated to the impossibility to define meta-classes in F-Logic. Concerning the types Boolean and Symbol, they are not intrinsically interoperable. To solve this problem a new class, that specializes the predefined FLORA2 class symbol, must be added. Members of this new class correspond to the Boolean values specified in Protégé. The solution proposed for the interoperability of the type String, is instead more complex. In FLORA2 does not exist a predefined class String but there is the concept of string inherited from XSB Prolog. Practically, a string is a series of characters delimited by inverted commas (“ ”). Internally, a string is represented as a list that contains the ASCII codes of composing characters. As an example, the string “*foo*” is represented by the list [102, 111, 111]. Therefore using the XSB Prolog function *is_charlist/1*, which controls if a given list is a string, we have defined a class that allow to produce “Yes” answers in correspondence of FLORA2 queries: *?- “foo”:string*.

In addition to the declaration of the types Any and Class, other operations exist that are not interoperable and for which, it is not possible to define any extension (not interoperable) as the definition of slot hierarchies.

3. *PROTEGE2FLORATAB* INTERFACE

The *Protégé2Flora2Tab* interface (shown in fig. 1) is activated from the main panel of the Protégé interface. The interface allows to perform the following operations: a) compilation in FLORA2 of an ontology that has been created by means of the Protégé panels and activation of the FLORA2 inferential engine b) visualization of the complete compiled code and storage of this code in a text file. Before the compilation of the ontology, the visualization and save buttons (commands) are not active.

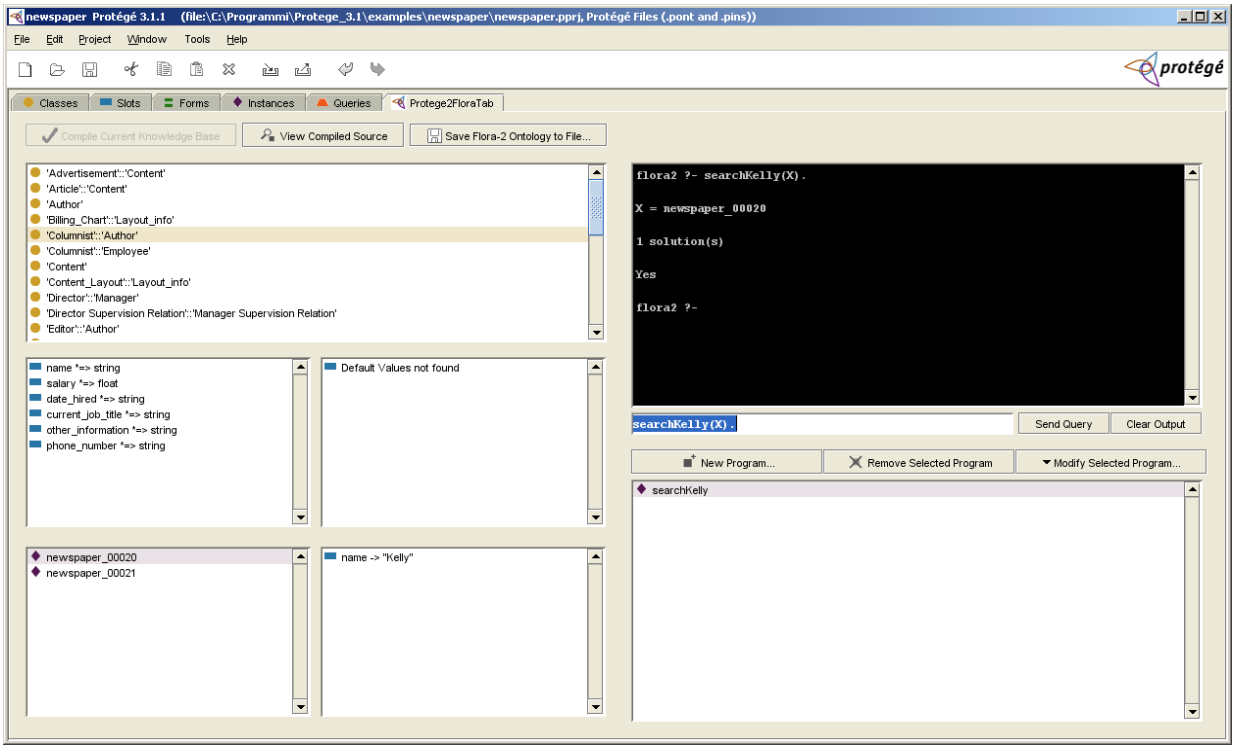

Figure 1

The five panels on the left side of the plug-in interface visualize the conversion of the ontology in FLORA2. These panels are active even before the compilation phase. An interesting *Protege2Flora2Tab* feature is the possibility to add FLORA2 rules that can be interpreted by the FLORA2 inferential apparatus . The button "*New Program*" activates a new panel (Figure 2) that allows to add new programs and to control their syntax before to save them. These programs will be compiled together with the other ontology entity definitions and therefore it is possible to recall all them through the panel of the FLORA2 queries.

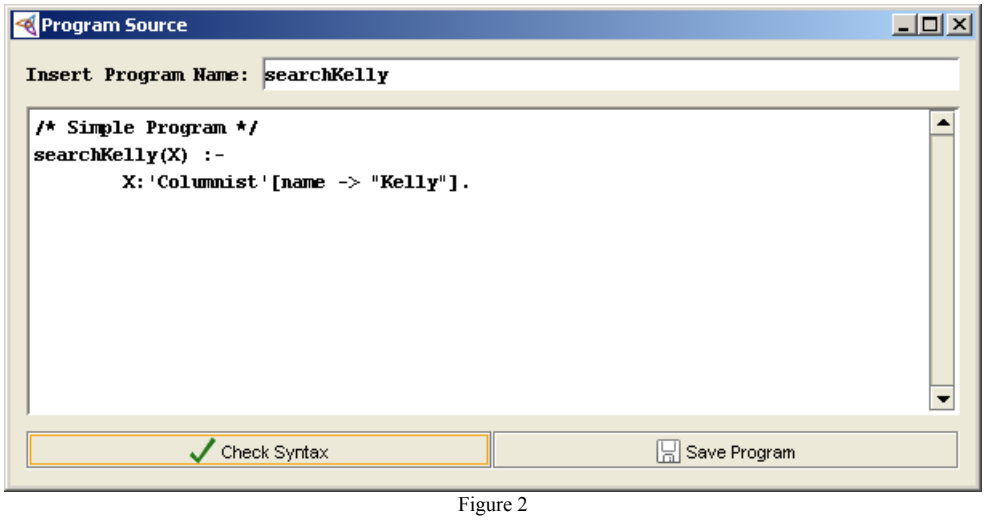

Figure 2

Protege2Flora2Tab allows to insert and to compile rules that will upgrade the inferential apparatus of the FLORA2 system and will extend the set of possible queries on the ontology entities. We emphasize that the programs will be saved as entities of Protégé, specifically as a Protégé meta-class (called *:FLORA-PROG*). This meta-class has two slots, *:PROG-BODY* and *:PROG-NAME*, which contain respectively the body and the name of the program.