E-learning and Semantic Technologies: Tools and Concepts

Enrique Alonso

Dept. Lingüística, Lenguas Modernas, Lógica y Filosofía de la Ciencia
Facultad de Filosofía y Letras
UAM
enrique.alonso@uam.es

Abstract. The use of automated tools for teaching logic shows a set of stages on which it is worth reflecting. Each of them make a different interpretation of the task which is strongly determined by the resources available. At present the generalization of *Markup Languages* commonly used to compose the documents populating the Web can pose a new model for tools employed to teach logic. After a brief historical overview of the previous stages, I will analyze the architecture of this new model and will offer two case studies on which there has already been some work done.

Keywords: Tools for teaching logic, Semantic Web, Web ontologies, WYSIWYM editors.

1 Automated Deduction Applications

Tools originally used for teaching logic could be considered as automatic calculators mainly oriented to show the students strategies to solve problems. These were small applications designed to offer to the beginners assisted examples of techniques they were supposed to acquire as a fundamental part of their formal skills in logic. Their finest moment arrives in the 90's. During this period the main programs that played some roles in the teaching of logic were published and distributed. Nevertheless their roots can be found in developments obtained during the second half of the 80's. Contrary to what one might expect, the first initiatives did not occur in the field of logical calculi, but in a mixed area heavily inspired in model theory. *Tarski's World* began to be distributed with *The Language of First-Order Logic* by J. J. Barwise and Etchemendy in 1990 but it is possible that early releases were already available some time before.

Today CSLI still maintains an active site, *The Openproof Project*, devoted to the study and design of software useful for teaching logic. According to the organizers words it is an initiative born in the early 80's focused almost entirely on applications aimed at what they describe as:

> application of software to Problems in logic.

P. Blackburn et al. (Eds.): TICTTL 2011, LNAI 6680, pp. 9–23, 2011.

Tarski's World was able to evaluate a set of formulas interpreted on an elementary geometric universe[1] and solve the question of the satisfiability with respect to the given model. Later, in 1994, *Hyperproof* appears, also under Barwise and Etchemendy's address, whose main novelty is the introduction of a system of rules capable of guiding the student to obtain consequences from an initial situation. Probably these are the most commonly used tools in real experiences of teaching logic over the years.

Although this kind of automated calculator represent the oldest use of informatic tools in teaching logic, the truth is that nowdays we can still find several research programmes focused on the design of calculators for diverse logical calculi. Former editions of Tools for Teaching Logic Congress show a useful guide through the main results in this field.[2] We can find two main developments, one oriented to the implementation of environments based on *natural deduction* and another centered in *analytic tableau*. In the first group, we should mention the initiative led by W. Sieg since late 80's. This work is still open under the name of the *AProS Project -A*utomated *Pro*of *S*earch Project. In its last release it included new topics going far beyond the strict domain of clasical logic. This is the case of the so called *Strategic Thinking* to which Sieg's group has devoted its attention. It is also worth mentioning the project *Pandora* led by K. Broda. It is surely one of the best software tools designed to show our students typical strategies for solving exercises in natural deduction calculus. In the field of analytic tableau we should include *Tableau III* developed in Oxford in 2001 and *LoTrec*, created under an IRIT initiative at Lilac. While the first one focuses on sentential logic and first order logic, the second is noteworthy because it is focused on modal and descriptive logics which is certainly a novelty.

I do not intend to go into details because the impression we get from a brief tour of these tools points to the existence of a jungle of initiatives. The success of these initiatives depends most of the time on the potential of the institution in which they were developed, and the capacity of enthusiasm and dissemination of their leading researchers. Those places where dialogue among computer scientist and logicians was fluid have promoted, as it is easy to imagine, earlier and more durable initiatives. In the others, and here I include almost every university department or research center with an active line of work in logic, we can also find at one time or another, initiatives aimed at creating some kind of automatic calculator with a potential use in teaching logic. Many of these have ceased, as is evidenced by the large number of abandoned sites, and others have never reached fully operational releases. But, perhaps this is not what really interests me. The interpretation of logic as a tool for the conceptual analysis of discourse and therefore as a discipline to evaluate argumentation is deeply rooted in our community. I have no objection against this way of seeing things, but it is not the way I understand the role of logic. It is true that almost any introductory course in logic includes among its objectives the description of several formal systems, calculus,

[1] It consists of a grid where users can locate a number of bodies such as spheres, pyramids, tetrahedrons, etc. The formulas to be evaluated make reference to the size and relative position of these figures on the grid.

[2] Cite the previous monograph.

and the exposition of the corresponding skills to solve problems with them. But I think the aim of logic as a discipline goes far beyond these very limited targets. Nevertheless, the software available from the late 80's and 90's does not allow the imagination of very different implementations of those just discussed. Therefore we should not be too critical of these projects. The lest pleasant result of this view was to reinforce the conception of logic as a tool for analysis and assessment of argumentation, something which conflicts with the objections coming from informal logic and argumentation theory. The relative lack of applications focusing on translation from ordinary language to different formal languages has been identified, with some irony, as evidence against the practical value of these methods in real case studies. Another objection points to the fact that sometimes what the students had just learned was actually the management of a program rather than the formal tools this application was intended to teach. There were other cases in which on the contrary, the implementation required an extensive prior knowledge of logic to achieve an efficient use of the environment, so that the desired effect was not achieved in either case.

I do not think we can judge this model harshly because it responds to a sincere attempt to provide our students with a user-friendly access to technicalities that have always been considered complex and have caused some rejection among our students. It should also be noted that in most cases it was quite difficult to design such programs so that the work done was really good. Perhaps the most interesting thing in this period was the experience acquaired by many researchers in combining logic and software engineering to design tools that currently go far beyond a limited educational use.

2 Transition: From Presentations to E-learning Platforms

2.1 Presentations

Since the end of the 90's the use of presentations has become common, initially in Congresses and scientific meetings, and later in the classroom. It is true that this tool is not specific to logic, but it would be unfair not to consider its role in support of teaching and it is necessary for me to clarify the point I want to make. The presentations preserve a structure developed a long time ago from the use of slides or overhead projectors. In fact it could be said that the normal use of these resources is nothing but a translation into the digital age of what many wanted to do with an overhead projector but couldn't. From my point of view this tool is of fundamental importance because it opens the use of new technologies to aspects of teaching of logic that goes beyond the typical homework. For a time it seemed that the only possible use of software for teaching logic was the applications designed to solve problems. It was as if the logical nature of programming was the only point of contact with logical enquiry. The proof of a theorem, the explanation of the reasons or motivations of some fundamental result and its historical context, were matters that laid outside the scope of teaching tools specifically designed for logic. This is the kind of conception that I will criticize later.

The use of presentations in the classroom was not generally widespread until the early twenty-first century and the reason lies in the cost of the equipment needed. Professionals had previously acquired a lot of experience from its use at conferences and scientific meetings so that the advantages were obvious for every one. Even though these advantages actually came from its non-digital ancestor: the slides used in overhead projectors. One of the most important features of presentations is the ability to reuse materials created for a session at any time and in any other disposition: the *cut and paste* resources became to be used by everybody. The ability to reuse materials justified investments in time and effort that would otherwise be undesirable. Our way to practice and devise teaching would not be understood without widespread use of these resources.

One of the obvious disadvantages of some of these tools is their mishandling of special symbols, and in particular, of logical symbols. This fact oriented its employment to a very general use avoiding any kind of technical terminology. Only the explanation of very general ideas and facts could obtain some kind of gains from the use of presentations. The emergence of LaTeX classes specifically aimed at the production of presentations in late 90's[3] solved the problem but always with the specific costs of production which has the composition of documents in LaTeX.

Despite these problems, presentations are interesting because of their undeniable contribution to a much broader understanding of the use of new technologies in teaching. What is needed is not only software oriented to solving problems, but perhaps to the construction of explanations, test batteries, or to the description of more complex matters such as proofs of some fundamental theorems. Why not?

It is true that the inability to access the code of the software used by the most popular tool, *Microsoft PowerPoint*, did not allow the user to imagine the design of specialized tools that could contribute to an automatic handling of new aspects of teaching. But this was not the case for free software environments such as LaTeX in which the software itself was available to the community of developers. In fact, I think that the philosophy supporting LaTeX can contribute decisively to define new models of tools oriented to teach logic. It is true that these applications, presentations, beamers, etc, are considered more often as simple audio-visual resources. However to the extent that it is possible to conceive them as the result of some programming it is also possible to imagine more sophisticated interventions for which the programming plays a role. This will be discussed later.

2.2 E-learning Platforms

For some time, presentations were used as desktop apps running from the client side therefore rejecting any dynamic component[4] from its behaviour. Today, we

[3] The first distribution of the *Beamer* class, perhaps the most widespread at present, dates form 2003.

[4] *Dynamic* behaviour does not reduce to the presence of mobile components such as pop-ups, emerging windows, advertisements, banners and so on, but a real interaction between client and server sides.

live in what is probably the golden age of the so-called *e-learning platforms*. I think everyone has suffcient experience with these resources and so I will spare unnecessary descriptions. The great contribution of these tools is the integration of almost every relevant component of the traditional teaching in a single environment. The main consecuence of this movement towards an unified enviroment for teaching tools is the integration of all those applicactions into a common technical infrastruture. The teacher's task now consists of gathering texts and materials used in class presentations and explanations, tests, bibliographies, links to external materials, applications to solve exercises, and so on in a site having the aspecto of a single web page. All this has to be put together inside a common technical ground offered by the encoding of the e-learning platform.

At present all these materials are developed using very different techniques, some are presentations in one of the formats characteristic for this type of tool, others are mere text files, forms in HTML, videos, hyperlinks, etc. There is no kind of unity among them except that they are introduced by teacher to gather them into the same environment. This situation is increasingly strange because in practice all of them are related to each other in several ways. A teacher may have uploaded a text that the student should read at some point, in turn, that material is relevant to the explanation which is summarized in a presentation also displayed in the site. Finally to check the undestanding of a problem it is possible that the teacher has designed a test that students have try to answer, and which will be in some way evaluated. To address questions relating to this test, it is highly desirable to have a forum and perhaps, if the teacher has enough time, it is also very useful to produce some feedback with corrections and comments. As we have seen, there is a continuity between all these materials that, however, can never be transmitted through an e-learning platform adequately. The only way to do this is to gather all the material into the same virtual unit establishing in a fully manual way the mutual references. Teacher are supossed to express through appropiate instructions the relationship of each item with the rest of the material stored. So if a paper, book or video is neccesary to understand a presentation, the teacher has to inform them of that by means of the corresponding directions. The materials needed to answer a test or a question have also to be indicated in some way. But can this be done in a better way? To the extent that each of the contents included in on-line courses are built from various and incompatible resources it seems very difficult to move towards a greater integration of all these resources. Presentations, texts, forms and so on are at this moment resources not compatible from the point of view of software engineering .

E-learning platforms can be done in several ways, but the truth is that the technical infrastructure can be extremely simple if desired. The architecture of these tools does not differ substantially from that work on a blog or a social network: all these resources are examples of what we call today called Dynamic Web, or also Web 2.0. It is based on a combination of programming and markup languages that has proven extremely effective.

The base infrastructure is provided by the HTML, the language in which the web is weaved, the interaction between users and sites becomes possible thanks to a typical bundle of forms and WYSIWYG editors based on JavaScript, while the interaction between user and server runs thanks to the combined action of PHP and MySql. The importance of this model is based on its simplicity and its definition from open source and free software initiatives, all of them oriented to Web design. With these initial conditions, it is not difficult to imagine a greater integration of the different resources that are now part of an online course. To define new models of relation between human action and automated processes in teaching is not a matter of discovering new technologies. It has to do with our own philosophy and general ways of conceiving the interaction and possibilities of techonolgies which are present among us.

3 Mark-Up Technolgies

Applications oriented to problem solving skills have promoted the habit of designing our own software solutions. Programing an application was not thought to be a waste of time, nor a task independent of logical concerns. The popularization of presentations had the advantage of extending the use of new technologies to very general matters like explanations, general results, historical background, and so on. Finally, e-learning platforms allow intergration of almost every kind of resource into a single working environment. Moreover, the technological ground of e-learning platforms is the same that makes possible the communication through the Net, a fact of fundamental importance to understanding the next step.

3.1 Semantic Technologies and Tagged Text

In May 2001, Sir Tim Berners-Lee published an extraordinarily thought provoking paper which raised the possibility of pushing the network architecture in a new direction: the Semantic Web. Since then intensive work has been carried out on the design of new formalisms in which logic has played some role[5] obtanining relevant advances in the definition and implementation of conceptual maps, *ontologies*, for many different areas. Nevetheless, the progress of semantic techniques and the semantic web programm has been less successful than expected.

At present, semantic web seems to have become a sort of vague claim for more active and efficient techniques to connect human bahaviour with automatic enviroments in the Net not propertly substantiated by an unique research programm. To obtain this result I think we should improve our understanding of content tagging through metadata and the ways in which machines and humans interact. Sematic web is no more the particular and very concrete research

[5] Some of the most popular languages in Semantic Web initiative such as OWL, come from *descriptive logics*.

program started years ago, but a general philosophy I prefer to associate with the *technologies for tagging text* and metadata handling. And now the question is how can we make use of these techniques to reconsider the use of software in teaching and in particular in the teaching of logic?

Teaching is an activity which, despite pedagogy demands, can be divided into a very specific set of events or stages:

i. The construction of explanations for a number of issues previously taken as a target.
ii. The compilation of lists of support materials or literature.
iii. The development of tests and questions designed to check students understanding.
iv. The evaluation of these tests and possible explanation for the errors.

E-learning platforms are designed to gather all these stages with some efficiency. Nevertheless, this can only be achived through the manual assembly of a large number of very different materials and resources.

To compound this the teacher will take as a starting point texts that are either theirs, or have been cited previously in the literature or even have been created expressly for the occasion. Presentations are usually drawn up on existing material cut and pasted onto slides to apply a certain format, animation, transitions, etc. Questions included in tests are often constructed from texts containing statements that actually are their answers. The procedure just described, which is the most common, suggests a considerable investment of effort and what is worse, the repetition of work that, at least in its most creative stages, has already been done.

Tagging or markup technologies, generally understood, meet all the requirements which implement metadata structures on an existing content or allow the elaboration of a text including metadata on items typed during the same process. I think it is possible to use some of these resources to develop simultaneously or at least in a coordinated way, some of the materials that make up the different teaching stages described above. In particular, starting from an existing text, I think it is possible to imagine a *modus operandi* to obtain both an interactive explanation of its content and a test or collection of test incluiding some feeback with a minimun of teacher intervention. The new model of tools for teaching would be oriented to the use of a minimal effort to obtain a large amount of items that now have to be independently developed. To achive this goal we would make use of different metadata structures, taxonomies,[6] that could be recursively or simultaneously used on the same content thereby generating different outcomes.

I do not pretend to eliminate any creative aspect of the various stages of teaching, but to reduce the extra effort and repetitive tasks presently involved in e-learning standards. From this point of view traditional teaching still results a more creative and amazing practice for teachers who do not find it very exciting

[6] We could have made use of the term *ontology* to describe this structure of metadata, but there is some controversy with respect to scope of this term, so that we avoid its use -see Gruber.

to retype contents they have written on previous occassions. The technique I propose to solve this disappointing look at e-learning technologies is not new, nor does it require either great effort in the field of programming. Luckily, it is something much more modest: just a little change in the way some existing technologies can be used and applied to the making-up proccess of teaching materials.

3.2 Towards a New Model

The main tool for producing content in the digital age is the *word processor*. Almost any other activity having to do with teaching makes, at one time or another, use of a word processor. Originally word processors were unable to display text on screen consistent with the final printed page. The speed limits of computer processors did not allow that outcome. The text present on the display was a combination of data and tags used to apply some styles to plain text.[7] This situation was understood from the outset as a flaw to be overcome in some way thereby fixing the *WYSIWYG* -What You See Is What You Get- philosophy, majority today. For some time the only alternative to this editing text philosophy was the different environments for editing in LaTeXdetermined to keep in view all metadata structure that was applied to the text for further compilation. This strategy resulted in a degree of unpopularity and the fame to be a tool exclusively oriented to scientific or professional editing. But the truth is that time has changed many of our views on this edition model to the point of giving it a label that identifies its ideological position. Word processor showing the code used to add metadata layers, semantic, typographical or whatever, received the name of *WYSIWYM* -What You See Is What You Mean- editors in a clear gesture of complicity to the initiatives coming from Semantic Web. Another factor that has contributed to a new understanding of these editors is the extension and popularization of HTML and all its cognates. Web pages and sites are supported by a structure written in HTML whose code can be seen at all times. HTML is a markup language designed to facilitate the display of any kind of data. The web exists thanks to a philosophy strongly based on the distinction and independence of data and metadata structure so that the net itself can be seen as a vast collection of documents linked by metadata structures of increasing complexity.

The recurring tendency in recent years has been to assess very positively the separation of data and metadata structures in a document, but I think we still have not drawn all the consequences of this fact. Something a user might want to do is to create their own taxonomy so that it could be efficiently incorporated in a user-friendly text editor. Perhaps they might want something else, such as applying not one but, several taxonomies simultaneously on a text that is being typed. Some of these taxonomies could be theirs while others may have been taken from other environments. The reason there hs been so little progress in

[7] WordStar or WordPerfect, two classics in text editing, started that way.

this direction may be due to the fact that using a taxonomy, let alone its design, to tag some text is seen as a highly specialized process far from simple user skills, but perhaps this is what is changing.

To apply a taxonomy on a text, usually a metadata structure, is something we should reflect on for a moment. The simplest way to do this is by using a plain text editor and choosing a sufficiently general syntax for the taxonomy terms, one that could be recognized as a standard. At present that standard is offered by XML syntax widely adopted by a lot of net languages which try to follow its recomendations. An example of a more or less standard tag in a XML syntax style could look like this[8]:

```
<sentence id=1>
  <who>Gödel</who>
  <verb>proved</verb>
  <what>the incompleteness theorem for PA</what>
  <when>in 1931</when>
</sentence>
```

It seems a bit naïve to demand that users type manually the tags to be applied into their documents. This is still more evident if we consider that the model tries to make it possible to use a large set of taxonomies at the same time. It does not seem reasonable to ask users to make the extra effort of learning the exact meaning and management of the tags of every taxonomy they use. People who used to write LaTeXdocuments are well aware of the advantages of having asisted editing environments based on IDEs -Integrated Developement Enviroments- to facilitate the process of creation and typing of content. Until now, to develop a text editor capable of integrating in their structure mechanisms to tag text was a task of some complexity, not much, but enough to require a certain expertise in the design of desktop applications. Nevertheless, a few years ago some WYSIWYG editors were adapted to network environments becoming typical in webmail apps and some e-learning platforms, *Moodle* represents a good example. The decision taken by some developers to offer free versions of these word processors has allowed us to understand an extremely simple technology and to use it to obtain experimental prototypes of *tagging editors* that have exactly the required functionalities. These word proccesors are mounted on a typical network architecture incluiding HTML and JavaScript code that allows us to add toolbars containing all the buttons and commands needed to implement an taxonomy. Each button in one of these toolbars launches the corresponding tag or the corresponding complex structure obtained from combining several tags in a higher level object. These tags would be then applied to a text previoulsy highlighted through mouse actions, or would be fullfiled by typing text into their delimiters. Nothing prevents the user, among other things, to also apply typographical format in exactly the same way. In fact, for one of these tag editors typographical format is just another taxonomy which can be implemented on

[8] This example is an advance of the taxonomies we will use in one of our case studies.

a text to obtain the desired result. All metadata structures seem to share a common logical status that would be semantic in all cases, or in none, but I recognize this is a controversial issue that I do not want to deal with now.

To arrange an environment to launch taxonomies, and tag text without changing its basic structure is a fundamental part of the developement model I intend to propose, but it is obviously only a part. It implies a level of abstraction of the basic functions of text editors not fully understood until now in all its importance. Text editors would be then interfaces between a taxonomy loaded for the occasion and the computer keyboard. Without a taxonomy or library previously added, our editor could only produce plain text, nothing else. With a taxonomy, not to say a bundle of them, a text editor can compose almost every conceivable outcome based on the web infrastruture.

This new generation of tools allows us to imagine the free creation of taxonomies useful for teaching needs, although it is obvious that nothing would prevent its use in any other area. But our goal is not, as seems obvious, the mere typesetting of a tagged text but the manufacture of certain products aimed at teaching tasks and specific needs. It seems clear that a document formatted with extra metadata structure somehow contains all the instructions needed to build some other entity. Data would be the arguments and metadata would represent the functions whose definition we have to get to produce the final product. In the model I have proposed the result of applying some metadata to the data previously given should be a dynamic website. It seems obvious that it is so if our goal is, among other things, to obtain test batteries to be automatically evaluated or some interactive explanations. In more strict terms, the desired result of a tagged text composed from a teaching taxonomy should be a collection of .html, .php and .js documents grouped into a functional recognizable unit.

The transformation of a tagged document in any other entity is carried out through a compiling process[9] which can consist of several stages. In our case it is possible to identify at least three:

i. *parsing* of the original document
ii. distribution of the data in a datatable
iii. data recovery from databases and its use to build the resulting files.

It is clear that the taxonomy used to tag the text typed by a text editor forms a functional unit with the compiler needed to produce the site that itself is the result of the process. This forces us to think about the construction of complex packages that can be treated as autonomous units of tools that play a role in different parts of the process. It may seem of a little complexity, but it is actually a model typical of free software developements and clearly growing today. In fact, it is used by some of the most successful companies in the field of CMSs. This model allows individual developers to produce *modules*, *components* and *plugins*

[9] This interpretation of the process of *compiling* could be considered too wide by many, but is quite usual today.

simply by accepting a minimum of standards in the design of the aproppiate interfaces to connect new apps with the core of these environments.

The model I have proposed consists of a core and some *components* to be rejoined in the kernel and that can be changed when needed. The kernel consists of two different units, a *text editor* and a *compiler*. The components incorporate on the one hand the taxonomy load into the text editor as a toolbar, and on the other the items and directions the compiler should use to translate the text generated by the editor into a final product. The idea is simple and could be summarized in the scheme showed in fig.1:

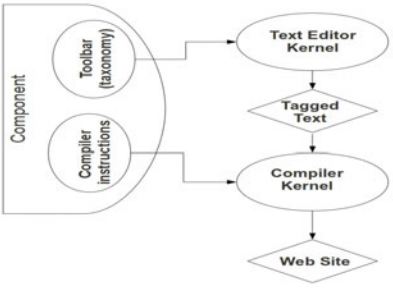

Fig. 1.

Although the work done so far can only be considered at an early stage, it is possible to analyze in more detail a couple of applications with a specific use in teaching and in particular in teaching logic.

3.3 Case Studies

As I have said throughout this brief presentation of the project, two of the most relevant stages in every teaching experience are the design of an exposition of some contents, and the construction of batteries of questions students must answer to check their comprehension of a subject. At present we already have some work done on the second of the two aspects and a series of assumptions about the first one. In the first place I will comment on the work that has already been done.

Components for designing test batteries. The target is to use an environment based on the architecture described above to automatically generate tests that students can answer receiving their qualifications and some comments about possible mistakes. The idea is to take as a starting point an already existing text and to add some format to generate questions. First, I will describe without details the taxonomy that can be used in the production of this type of content and a general example of the expected result.

An erotetic taxonomy (abstract)

- main element:

  `<sentence id=...></sentence>`

- erotetic tags:

  `<who></who>, <what></what>, <when></when>, <where></where>,`

  `<how></how>`

- sentence core:

  `<verb></verb>`

We now consider a very basic example. Let us suppose that we load into our text editor a text containing the following piece of information:

> " In 1931 Gödel probes undecidability of the First Order Peano Arithmetic frustating the formalistic program hopes sustained by Hilbert and his colleagues."

There are several ways to tag a statement like this and some of them may require some rewriting of the original text but I will avoid this aspect. A possible result is this:

```
<sentence id=1>
<when> In 1931 </when>
<who>Gödel</who>
<verb>proved</verb>
<what>the undecidability of First Orden Peano Arithmetic</what>
frustrating the hopes of formalist programm sustained by Hilbert
and his followers
</sentence>
```

We assume that the addition of tags has been achieved through the regular use of the mouse to highlight the affected data and then clicking on the tabs corresponding to the tag you want to launch. This tag is then inserted between the highlighted text introducing spaces, line breaks or other graphic resources to facilitate the proper display of the structure so generated. As can be seen in the example, not all the text has been tagged. We can make partial selections or we can also rewrite some parts to obtain a better grammatical coherence.

From this statement plus the metadata structure the system can generate at least the following questions:

i. When does Gödel prove the undecidability of ...?
ii. Who does prove the undecidability of ...?
iii. What does Gödel prove in 1931?

Depending on the grammatical structure of each language, the interface employed to generate question forms from the corresponding indicative sentences will be more or less complex. It is therefore important that this project remains open to component additions coming from independent groups interested in this type of technology. Really nothing prevents the existence of a number of taxonomies or compilers with different interpretations of the grammatical structure of erotetic logic even into the same language. What matters is the final result, and not the way to conceptualize content which may vary between authors.

The arrangment of questions in a test can adopt two possible structures: a multiple choice test, or open questions. In the first case students are suppossed to choose the right answer among a set of plausible alternatives. The system then suggests a qualification offering the correct answer and some evidence supporting it. In the second case, students can type an extended answer and the system only suggest the correct answer showing the corresponding evidence. The forms contained in fig. 2 is an example of typical multiple-choice test.

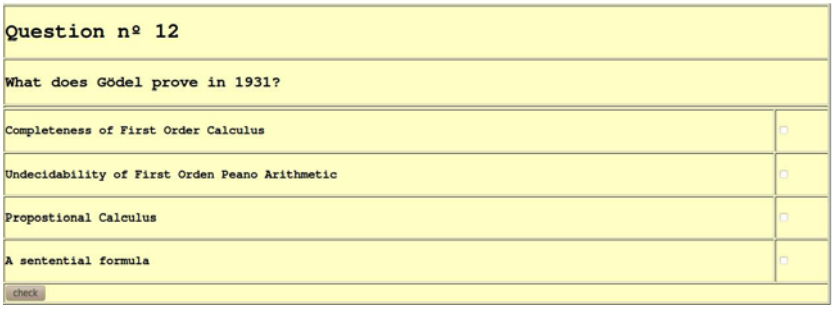

Fig. 2.

The generation of possible responses, including *inter alia* the right, is done through random selection of items within the same category the right answer belongs to. If the target of a question is of type *what*, as in the example, the other alternatives must also have been obtained from other items in the same field. This strategy ensures partial coherence among alternatives in a test, but does not offer a complete guarantee of it. The last two answers listed as alternatives in the example above, are partially inconsistent though perhaps not too much so as to be excluded. Even though this part of the process should be executed in a purely automated way, the truth is that the questions must pass a human filter to discard absurd tests of null interest for students. In order to produce batteries of questions with a suitable number of alternatives the system must possess a certain amount of data. If this critical level, calculated by the system, was not reached, it would be most feasible to propose only free-response questions in which this kind of problem does not arise.

Once the student responds by clicking the tab in the form, they should receive a reply containing at least the right answer and the evidence supporting it, as reflected in the fig. 3.

Fig. 3.

The dialogue the student establishes with the environment through their answers and their own assesment of them through the evidence supplied by the system is a really interesting field worth analyzing, but this is beyond the scope of this paper.

Components to design expositions. As I have said elsewhere in this same study, explanations or simple expositions are now provided by *presentations* built up with Microsoft *PowerPoint* or the beamer class of LaTeX. To imagine a taxonomy able to produce a bundle of dynamic web pages containing certain content is not difficult. To attach some of the dynamic behavior characteristic of presentations is trivial. To incorporate some feedback based on interactions between client and server side actions is now possible, thanks to the fact that our presentations are no different from any other dynamic web site. Its is pointless to say that this is something that is beyond the scope of the usual presentations. Now it is possible, for example, to suggest that the student answer a question, to find some data or content displayed in a short video and then follow the next step of our exposition. One advantage recently added to HTML web pages is the ability to compile text sequences encoded in LaTeXto show the corresponding result which entirely opens the possibility of using this procedure on typical contents of our discipline.

This approach only supposes some novelty with respect to the way we produce the file or final site, but I do not intend to present it as a relevant contribution in any way. It is quite evident that the latest moves in the generation of content on multiple e-learning platforms are now going in that direction. What really interests me is the application of the procedure to the presentation of contents that so far have not had an easy treatment in logic. I am thinking in particular about Theorems and other results that constitute the basic core of nonelementary courses of logic. The structure of expositions of these contents shows general patterns which are easy to obtain and translate into an appropriate taxonomy. The templates developed over the years by users of LaTeX either individually, or within specific groups,[10] show a useful example of what can be done in this

[10] The scientific publishers are the best example of this.

direction. Definitions, lemmas, theorems, corollaries form what would be some of the basic categories that should certainly be accommodated within a taxonomy oriented to presentations of logic contents. But we can go one step further and incorporate some of the basic skills needed to prove theorems in logic. I think of techniques such as *induction* and *reductio*. We all have enough experience in such techniques to know that it is usually possible to identify the basic components and display them in a reasonably organized way. For a better result, I suppose it would be nice to add the logical connectives and cuantors and also some argumentative particles usually employed in arguments, and proofs proper of purely formal contexts. This would ultimately make effective, not without some irony, the old dream of *metamathematics*, but on completely different basis.

On this occasion, I do not dare to advance a more detailed description of taxonomies that could incorporate the management of such kind of items. However the work is promising and provides not only a technology to improve teaching skills in logic, but also an elucidation of the formal devices that allow us to describe, I will not say to *formalize*, the structure of proofs of fundamental theorems in our discipline. Depending on the success of this method, it could also be considered the design of tag systems devoted to show our students typical strategies to prove theorems in logic, even in those cases in which some originality is present.

References

1. van Ditmarsch, H., Manzano, M.: Editorial 'Tools for Teaching Logic'. Logic Jnl IGPL 15(4), 289–292 (2007)
2. Broda, K., et al.: Pandora: A Reasoning Toolbox using Natural Deduction Style. Logic Jnl IGPL 15(4), 293–304
3. Sieg, W.: The AProS Project: Strategic Thinking and Computational Logic. Logic Jnl IGPL 15(4), 359–368
4. Pérez-Lancho, B., et al.: Software Tools in Logic Education: Some Examples. Logic Jnl IGPL 15(4), 347–357
5. Huertas, A.: Teaching and Learning Logic in a Virtual Learning Enviroment. Logic Jnl IGPL 15(4), 321–331
6. Berners-Lee, T., Fischetti, M.: Weaving the Web. Harper Collins Publisher, New York (1999)
7. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American (May 17, 2001)
8. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition 5(2), 199–220
9. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. International Journal of Human-Computer Studies 43(4-5), 907–928
10. Openproof Project, `http://ggww.stanford.edu/NGUS/Openproof/`
11. W3C Semantic Web Activity, `http://www.w3.org/2001/sw/`