Teaching NL to FOL and FOL to CF Conversions

Ioannis Hatzilygeroudis

University of Patras, Dept of Computer Engin. & Informatics
26500 Patras, Hellas (Greece)
ihatz@ceid.upatars.gr

Abstract

In this paper, we present ways of teaching the NL to FOL and the FOL to CF conversions. NL to FOL conversion teaching is based on the use of a structured and interactive process of making the conversion, introduced here, which helps students, and the support of two software tools, one based on an implementation of that process and the other making a translation from a FOL formula to a NL-like sentence. FOL to CF conversion teaching is based on a software tool that implements the conversion process and allows the intermediate steps to be visible. This gives the students the capability of trying the conversion of any FOL formula to its CF. The first experience results are promising.

Introduction

Knowledge Representation & Reasoning (KR&R) is a fundamental topic of Artificial Intelligence (AI). A basic KR language is First-Order Logic (FOL), the main representative of logic-based representation languages, which is part of almost any introductory AI course and textbook (e.g. Russell and Norvig, 2003; Luger, 2004). In some proposals, emphasis is given to the propositional logic (PL) (Neller et al, 2006) as an easier way of understanding logic. However, PL is weaker than FOL, in that we cannot use variables. This makes PL less expressive and in some sense less interesting. Use of variables makes FOL capable of solving more interesting problems, e.g. answering questions of 'who' type, which our students find quite interesting. So, FOL is more expressive and seems to be more exciting in representing knowledge and reasoning with it, although more complicated.

To make automated inferences, Clause Form (CF), a special form of FOPC, is used. Teaching FOL as a knowledge representation and reasoning language includes many aspects. Two of them are (a) representing natural language (NL) sentences into FOL formulas and (b) transformimg complex FOL formulas into CF (In the following, for the sake of simplicity, we use the term "converting" or "conversion" in both cases). For the latter, there is a well-defined algorithmic process that has specific steps and can be automated via a computer program. That process can be found in almost all AI introductory textbooks. The former is an ad-hoc process; there is no specific algorithm that can be automated within a computer. This is mainly due to the fact that NL has no clear semantics as FOL does. Also, most of existing textbooks do not pay the required attention to that. They simply provide the syntax of FOL and definitions of the logical symbols and terms (e.g. Russell and Norvig, 2003; Luger, 2004). Even more specialized textbooks do the same (e.g. Brachman and Levesque, 2004). At best, they provide a kind of more extended explanations and examples (e.g. Genesereth and Nilsson, 1987). They do not provide any systematic guidance towards it.

We consider that the NL to FOL conversion is of great importance in teaching logic as a KR language. Understanding of when and for what to use a predicate, a function or a logical connective gives a more complete insight and understanding of FOL as a KR language. Furthermore, given that a knowledge engineer will often face the fact of converting NL sentences into FOL, while acquiring knowledge, exact knowledge of the factors of the conversion is very crucial.

On the other hand, the FOL to CF conversion is also necessary for better understanding both FOL and CF. Existing textbooks again do not pay as much attention as required to this process. Also, existing software does not make the process explicit. Rather, it leaves the process implicit, making such software unsuitable as a teaching tool.

There are several systems, like Plato and Logic Toolbox (see corresponding sites in References) that are characterized as logic educational software. However, most of them have been developed at Philosophy Departments and deal with how to construct formal proofs mainly using natural deduction rules, restricting themselves to PL.

In this paper, we introduce ways for effective teaching of both NL to FOL and FOL to CF conversions. For the former, we provide a structured and interactive process supported by corresponding software. For the latter, we provide a tool helping in teaching. The tools are part of a web-based sytem for teaching AI aspects (Hatzilygeroudis et al, 2004).

The structure of the paper is as follows. Next section deals with the NL to FOL conversion by presenting the proposed teaching process in some detail. The third section deals with corresponding aspects of FOL to CF conversion. In the fourth section, we present some evaluation results and finally last section concludes the paper.

The NL to FOL Conversion

The problem of the NL to FOL conversion concerns translation of a given NL sentence into a FOL formula. The main problem lies at the NL side. NL has no clear semantics, so NL sentences are not always absolutely clear, hence translation into FOL is not easy and may be not unique. Given that, the students should be advised to provide a clear interpretation of the NL sentence which their translated FOL formula corresponds to. In Figure 1, we present a skeleton of the proposed teaching process.

In step 1, an introduction to FOL syntax is performed. Then, some important aspects of NL to FOL conversion are presented and explained to the students via simple examples. In step 3, a structured and interactive process for the conversion, introduced in this paper, is presented and explained to the students via examples too. Afterwards, in step 4, the students are called to apply that process with the help of a special software tool, which implements the process, to predetermined examples. Finally, students are called to try NL to FOL conversion of unknown NL sentences having as only help another special software tool that converts FOL formulas into NL-like sentences. Thus, students can self-check their answers.

1. Introduction to FOL Syntax.
2. Presentation of interesting cases of NL to FOL conversion.
3. Introduction to the structured and interactive process for NL to FOL conversion.
4. Use of supporting software by the students on specific examples of NL to FOL conversion.
5. Use of FOLtoNL software by the students to try not predetermined examples.

Figure 1: NL to FOL Teaching Process

Given that FOL syntax is well-known and included in all introductory AI textbooks, we do not refer here to it. In the following subsections, we elaborate on the rest of the steps reported in Figure 1.

Interesting Cases

Some interesting cases of NL to FOL conversion, which if not paid attention to, may lead to errors are presented to the students via simple examples, after the introduction to FOL syntax. Some such cases are the following:

(a) Misuse of $\wedge$.
Example: "*All dogs love playing games.*"
Common error: *$(\forall x)$ dog(x) $\wedge$ loves(x, playing_games)*
(*It means that all members of the universe of discourse are dogs and love playing games, which is a wrong interpretation.*)
Right formula: *$(\forall x)$ dog(x) $\Rightarrow$ loves(x, playing_games)*

(b) The order of $\forall$ and $\exists$
Example: "*All love somebody.*"
Common error: *$(\exists y)$ $(\forall x)$ loves(x, y)*
(*It means that "somebody" is common to all, which is a wrong interpretation.*)
Right formula: *$(\forall x)$ $(\exists y)$ loves(x, y)*

(c) Use of a function
Example: "*Pluto loves its master.*"
Common error: *$(\forall x)$ master(x, pluto) $\Rightarrow$ loves(pluto, x)*
(*It supposes that Pluto has more than one master, which is a wrong interpretation. It would be the case for "Pluto loves its masters."*)
Right formula: *loves(pluto, master_of(pluto))*
(*A function represents an one-to-one relation.*)

(d) $A \Rightarrow (B \Rightarrow C) \equiv B \Rightarrow (A \Rightarrow C) \equiv (A \wedge B) \Rightarrow C$
(*It is easily proved via De Morgan's laws.*).
It is useful to have it in mind when converting NL to FOL (see next subsection for an example).

The above interesting cases are indicative. Some more could be added.

A Structured and Interactive Process

The difficulty in NL to FOL conversion comes not only from the unclear semantics of NL, but also from the lack of a systematic way of making that conversion. Most textbooks restrict themselves in giving some general guidelines (at best) for making the conversion, mainly via specific examples. We believe that a structured way of making that conversion helps students to overcome many of the difficulties and better understand both the working and the semantics of FOL.

We propose the structured and interactive process for making the NL to FOL conversion presented in Figure 2. To illustrate it, we give two example applications of the process.

1. Find the predicates and specify their arguments (number, type).
2. Construct corresponding atoms.
3. Divide atoms of the same level into groups.
4. Specify the connectives between atoms of each group and construct corresponding formulas.
5. Divide formulas and/or any of the remaining atoms of the same level into groups. If there are no such groups, go to step 7.
6. Specify the connectives between elements of each group, construct corresponding next level formulas and go to step 5.
7. Specify the quantifiers of the variables.
8. Construct the final FOL formula

Figure 2: Structured and Interactive Process for NL to FOL Conversion

Let "*All humans eat some food*" be the given NL sentence. Following the steps of the SIP, we have:

Predicates (step 1)
human/1 (variable: *x*)
food/1 (variable: *y*)
eats/2 (variables: *x*, *y*)

Atoms (step 2)
human(x)
food(y)
eats(x, y)

Groups-1 (step 3)
{*human(x), food(y)*} {*eats(x, y)*} or {*food(y), eats(x, y)*} {*human(x)*}

Formulas-1 (step 4)
$human(x) \wedge food(y)$, $eats(x, y)$ or $food(y) \Rightarrow eats(x, y)$, $human(x)$

Groups-2 (step 5)
$\{human(x) \wedge food(y), eats(x, y)\}$
or
$\{human(x), food(y) \Rightarrow eats(x, y)\}$

Formulas-2 (step 6)
$human(x) \wedge food(y) \Rightarrow eats(x, y)$
or
$human(x) \Rightarrow food(y) \Rightarrow eats(x, y)$

Quantifiers (step 7)
$x \rightarrow \forall$ (*because of "All"*)
$y \rightarrow \exists$ (*because of "some"*)

Final FOL formula (step 8)
$(\forall x)\ (\exists y)\ (human(x) \wedge food(y) \Rightarrow eats(x, y))$
\` or
$(\forall x)\ (\exists y)\ (human(x) \Rightarrow (food(y) \Rightarrow eats(x, y)))$

To get a better understanding of what we mean by "same level", we present another example. Let's suppose that the final FOL formula is the following: $(A \wedge (B \vee (C \wedge D)) \Rightarrow ((E \wedge F) \Rightarrow G)$, where A, B, C, D, E, F and G are atoms. Let us look now at the process of constructing that formula (we start from step 3, since we consider that steps 1 and 2 have already been done):

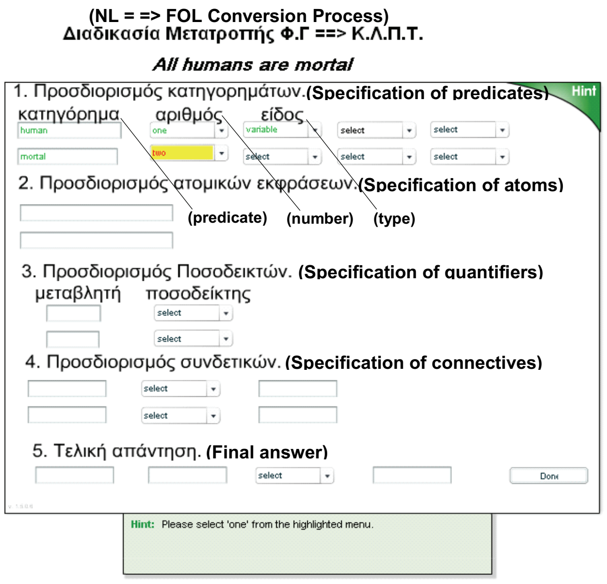
Step 3: {C, D}, {E, F} (two groups)
Step 4: $F1 \equiv C \wedge D$, $F2 \equiv E \wedge F$ (two formulas)
Step 5: {B, F1}, {F2, G} (two groups)
Step 6: $F3 \equiv B \vee F1$, $F4 \equiv F2 \Rightarrow G$ (two formulas)
Step 5: {A, F3} (one group)
Step 6: $F5 \equiv A \wedge F3$ (one formula)
Step 5: {F5, F4} (one group)
Step 6: $F \equiv F5 \Rightarrow F4$ (one formula)
Step 7: (not applicable to this example)
Step 8: $(A \wedge (B \vee (C \wedge D)) \Rightarrow ((E \wedge F) \Rightarrow G)$

Use of Supporting Software

In the last step of the process of Figure 1, the students are called to get experience in using the process of Figure 2 via special software. This software is designed to help students to follow the NL to FOL process of Figure 2. It consists of a database of examples that are presented and solved by the students in a way that implements the structured and interactive process of Figure 2.

In Figure 3, a snapshot of a session with the NLtoFOL software tool is depicted. At the moment a Greek based interface is used (English translation is given in parentheses), although the sentences are given in English.

The student is given a NL sentence (“*All humans are mortal*” in our case) and is asked to fill in a number of boxes with the right item. If it is the right one, then it gets a green color. If not, it becomes red and a red warning may be displayed in the hint display area. The student can also ask for a hint by clicking on the green triangle button at the top right position of the user interface. For example, in the snapshot of Figure 3, the selection of the student is wrong (the ‘mortal’ predicate has no two arguments), therefore it became red. The student then used the ‘hint’ button and a hint message has been displayed at the bottom, in the hint display area, saying that the student should select ‘one’. The hint does not always proposes the right item to be filled in, but may be some specific info to help the student to select the right item.


(NL = => FOL Conversion Process)
Διαδικασία Μετατροπής Φ.Γ ==> Κ.Λ.Π.Τ.

All humans are mortal

1. Προσδιορισμός κατηγορημάτων.(Specification of predicates)

κατηγόρημα	αριθμός	είδος		
human	one	variable	select	select
mortal	two	select	select	select

2. Προσδιορισμός ατομικών εκφράσεων.(Specification of atoms)

(predicate) (number) (type)

3. Προσδιορισμός Ποσοδεικτών. (Specification of quantifiers)

μεταβλητή	ποσοδείκτης
	select
	select

4. Προσδιορισμός συνδετικών. (Specification of connectives)

	select	
	select	

5. Τελική απάντηση. (Final answer)

		select		Done

Hint: Please select 'one' from the highlighted menu.

Figure 3: A snapshot of a session with the NLtoFOL Software

Apart from the NLtoFOL software, we have developed another special software that makes the inverse conversion: from FOL to NL. The FOLtoNL software is a rule based system implemented in Jess. Jess is a rule-based expert system shell, implemented in Java (Friedman-Hill, 2003). FOLtoNL can take any FOL formula as input, check its syntax and translate it into a NL-like sentence. An example run is displayed in Figure 4.

The FOLtoNL software gives the opportunity to the students to check their answers to any NL to FOL conversion. Because, as stated, the NL to FOL conversion cannot be automated, there is no means for a student to check his/her answer to an ‘unknown’ NL sentence. By using FOLtoNL, he/she can have a NL-like translation of his/her answer, which is much easier to compare with the initial NL sentence.

```
Jess> (clear)
TRUE

Jess> (batch FOLtoNL.clp)

First Order Logic to Natural Language

Give a sentence in First Order Predicate Calculus:

(all ?x) bird(?x) & ~flies(?x) & swims(?x) => penguin(?x)

INTERPRETATION : forall x if x is a bird and x does
not fly and x does swim then x is a penguin .
```

Figure 4: A run of the FOLtoNL tool

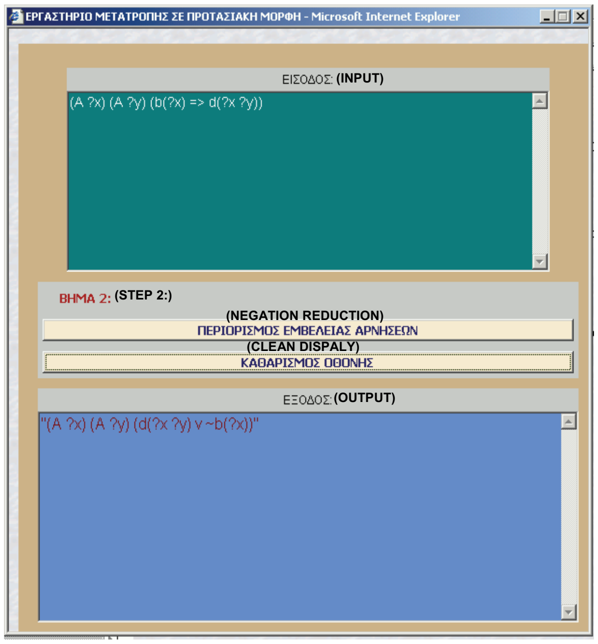
The FOL to CF Conversion

As stated above, the FOL to CF conversion is a well-defined process that can be automated within a computer. Most of existing systems make it in one step manner. That is, one cannot see all the steps of the conversion. Therefore, they are not suitable to be used in teaching. What is needed is a software tool that will allow the conversion steps to be visible.

We have constructed such a software tool and use it in teaching FOL to CF conversion (see Figure 5). The interface of the tool, which is Greek based (English translation is given in parentheses), includes three areas. The upper area is used for FOL formula input. The lower area is used to display each step’s result. Finally, the mid area specifies the step to be executed next. Conversion includes six steps:

1. implier elimination
2. negation reduction
3. variable renaming and transformation in PNF
4. Skolemisation and universal quantifiers removal
5. transformation into CNF
6. clause extraction and variable renaming.

The answer is constructed in a stepwise manner. In every step the system gives the student the right answer. For example, in Figure 5, step 1 has been performed and step 2 is the next one to be performed. So, he is able to check at every step if he has made some mistake. If he has, he is able to go on with the next step, using the correct answer given by the system. After completing the whole function, that is all steps, the final result, the correct clause in CF, appears.


ΕΡΓΑΣΤΗΡΙΟ ΜΕΤΑΤΡΟΠΗΣ ΣΕ ΠΡΟΤΑΣΙΑΚΗ ΜΟΡΦΗ - Microsoft Internet Explorer
ΕΙΣΟΔΟΣ: (INPUT)
ΒΗΜΑ 2: (STEP 2:)
(NEGATION REDUCTION)
ΠΕΡΙΟΡΙΣΜΟΣ ΕΜΒΕΛΕΙΑΣ ΑΡΝΗΣΕΩΝ
(CLEAN DISPALY)
ΚΑΘΑΡΙΣΜΟΣ ΟΘΟΝΗΣ
ΕΞΟΔΟΣ: (OUTPUT)

```
(A ?x) (A ?y) (b(?x) => d(?x ?y))
```

```
"(A ?x) (A ?y) (d(?x ?y) v ~b(?x))"
```

Figure 5: The FOLtoCF tool interface

A student can try to convert any FOL formula to its CF, in a stepwise manner. So, on the one hand, a student can check his/her ability in converting any formula and, on the other, he/she can practise by trying as many formulas as he/she wishes.

For typing convenience, we use the following symbols: '~' (negation), '&' (conjunction), 'V' (disjunction), '=>' (implication), 'all' (universal quantifier) and 'exists' (existential quantifier).

The FOLtoCF tool is actually part of ACT-P, a configurable automated theorem prover (Hatzilygeroudis and Reichgelt, 1994). ACT-P is implemented in Lisp, but its FOL to CF part has been transported in Java via a Java-based Lisp interpreter (Hatzilygeroudis et al, 2004).

Evaluation of Methods

We used the methods of teaching related aspects of FOL as a KR&R language presented in the previous sections during the AI introductory course at our department last year. We felt that the students were satisfied from using them. To validate our intuition, we made some kind of evaluation. We first distributed a questionnaire to a special group of students. There were a number of students that had started attending the AI course the year before the last one. For some reason they had stopped attending it and started attending it again last year. This happens sometimes, because the AI course is an elective course. Some students select it one year, but afterwards, because they realized they took more electives than could bear, abandon it and attend it next year. More than 80% of the students were more satisfied with the new method than the previous one that was based more or less on an ad hoc way of teaching without using the structured method and the corresponding software tools.

A second thing we did was to compare the exam results of the 2004-5 class, with which we did not use the presented teaching methods, and the 2005-6 class, where we used them. The comparison is based on the questions of the exam papers related to the system topics (see Table 1). It is clear that there was a remarkable improvement to the class average mark as far as NL to FOL conversion question is concerned, but a less remarkable improvement as far as FOL to NL conversion question is concerned. The first result is quite encouraging. It was actually anticipated, given that the NL to FOL process is a not well defined process and structuring it could greatly help. The second result is not as remarkable as the first, but it was also anticipated given that the average mark was already high. Although the number exam papers examined was relatively small (12 for 2005 and 16 for 2006) and the results were not statistically significant, they remain valuable.

Table 1. Exam Results

Question	Feb-2005 exam	Feb-2006 exam
NL to FOL (20 marks)	13.8	14.9
FOL to CF (15 marks)	12.8	13.2

Conclusions

In this paper, we present structured and interactive ways of teaching two basic conversion processes in teaching FOL as a KR&R language, the NL to FOL and the FOL to CF conversions. Both ways are supported by specialized software. We believe that contemporary AI education should not be based exclusively on theoretical class lectures, given the current capabilities of technology.

Provision of specific structure in a typically unstructured process, such as the NL to FOL process, gave a better understanding of various aspects of it. Students practicing with the supporting tools gained valuable hands on experience on both NL to FOL and FOL to CF conversions. Small scale evaluation results are encouraging.

There are some directions that this work can be improved. First, the tools used can be improved in some ways. For example, the FOLtoNL tool can be substantially improved. The way it translates the FOL sentences into NL ones is not satisfactory yet. It is actually the first step to real NL expressions. For example, in the case used in Figure 4, the desired output would be "penguins are birds that do not fly but can swim" or something close to that. This is a direction of further work in this aspect.

Second, a more detailed and more extensive evaluation can be done. This could reveal weaknesses of the methods and the tools whose elimination effort might lead to important changes. This is another direction for further work.

References

Brachman, R. J. and Levesque, H. J. 2004. *Knowledge Representation and Reasoning*. Elsevier.

Friedman-Hill, E. 2003. *Jess in Action: Rule-Based Systems in Java*. Manning Publishing.

Genesereth, M. R. and Nilsson, N. J. 1987. *Logical Foundations of AI*. Morgan Kaufmann, Palo Alto.

Hatzilygeroudis I. and Reichgelt H. 1994. ACT-P: A Configurable Theorem Prover. *Data & Knowledge Engineering* 12, 277-296.

Hatzilygeroudis I., Gianoulis Ch. and Koutsojannis C. 2004. A Web-Based Education System for Predicate Logic. *Proceedings of the IEEE ICALT-2004*, Joensuu, Finland, Aug 30-Sept 1, 2004, 106-110.

Logic Toolbox: http://philosophy.lander.edu/~jsaetti/ Welcome.html (access: November 2006).

Luger, G. F. 2004. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. 5th Edition. Addison-Wesley.

Neller, T., Markov, Z. and Russell, I. 2006. Clue Deduction: Professor Plum Teaches Logic. *Proceedings of the 19th International FLAIRS Conference (FLAIRS-2006)*, Melbourne Beach, Florida, May 11-13, 2006, pp. 214-219.

Plato: http://www.utexas.edu/courses/plato/info.html (access: November 2006).

Russell, S., and Norvig, P. 2003. *Artificial Intelligence: a modern approach*. 2nd Ed. Upper Saddle River, NJ, USA: Prentice Hall.