An Example-Tracing Tutor for Teaching NL to FOL Conversion

Themistoklis Chronopoulos, Isidoros Perikos, Ioannis Hatzilygeroudis

An example-tracing tutor for teaching NL to FOL conversion

Themistoklis Chronopoulos, Isidoros Perikos, Ioannis Hatzilygeroudis

Department of Computer Engineering & Informatics, School of Engineering
University of Patras, Greece
txronop@yahoo.gr, {perikos,ihatz}@ceid.upatras.gr

Abstract. In this paper we present an Example-tracing Tutor for the conversion of a sentence written in natural language (NL) to a sentence written in first order logic (FOL), which is a basic knowledge representation language. The tutor is based on the scripting of the process of the NL to FOL conversion and it has been authored using the Cognitive Tutoring Authoring Tool (CTAT) in which we have implemented a completed student interface and we also have created a Behavior Recorder graph for the above process.

Keywords: cognitive tutor, example-tracing tutor, knowledge representation tutor.

1. 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 [1, 2]. To make automated inferences, Clause Form (CF), a special form of FOL, is used. Teaching FOL as a knowledge representation and reasoning language includes many aspects. One of them is translating natural language (NL) sentences into FOL formulas. It 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 [1, 2]. Even more specialized textbooks do the same [3]. At best, they provide a kind of more extended explanations and examples [4]. They do not provide any systematic guidance towards it. Given the above, students usually find difficulties in learning the task of formalizing NL sentences in FOL, which confronts to tutors' common experience.

In [5], we introduced a structured process for guiding students in translating a NL sentence into a FOL one, namely the SIP process. In [6], we presented a web-based system implementing the SIP process, i.e. helping students in learning how to convert NL sentences into FOL formulas. Having used the above system for some time, we resulted in the following findings: (a) At a first stage, students may not be necessary to work with sentences that produce formulas with more than three groups of atoms or

with more than one group of formulas. (b) Tutors would like to use a graphical way of describing the SIP steps for each formula and a way of massively inserting them. Also, they would like related hints or feedback messages to be presented to the users in case of errors. (c) A stand-alone version of the system would be useful.

Example-tracing tutors [7] are a recent type of tutors that perform model tracing, provide context-sensitive instruction, in the form of hints and error feedback messages, and are flexible to multiple possible solution strategies and paths. Authoring for this flexibility is based on the explicit demonstration of alternative paths in each problem. No programming is required. CTAT (Cognitive Tutor Authoring Tools) is an authoring tool for creating example-tracing tutors [8, 9, 10]. So, CTAT is suitable for implementing the SIP process and satisfying the above requirements.

In this paper, we present a new lighter implementation of the SIP process in the form of an example-tracing tutor.

2. Related Work

There various systems that are used or have been created for teaching or helping in teaching some kind of logic and logic-based reasoning. Logic Tutor [11] is an intelligent tutoring system (ITS) for learning formal proofs in PL based on natural deduction. As an intelligent system, it adapts to the needs of the students via keeping user models. So, it provides context-sensitive feedback and exercises tailored to the users. Logic-ITA [12] is actually an extension to Logic Tutor, where a tutor part has been added, used as a teaching assistant system. P-Logic Tutor [13] is also a kind of intelligent tutoring system aiming at teaching students fundamental aspects of PL and theorem proving. To this end, it provides an interactive web-based interface. All the above systems, although deal with learning and/or teaching logic, they are not concerned with how to use predicate logic as a KR&R language. They do not deal with how to formalize a NL sentence into FOL.

As far as we are aware of, there is only one system that claims doing the latter. It is called KRRT (Knowledge Representation and Reasoning Tutor) [14]. It is a web-based system that aims at helping students to learn FOL as a KR&R language. It deals with both knowledge representation in and reasoning with FOL. The translation form NL to FOL takes place in its KR part. However, the only help provided to the students is at syntactic and logical equivalence levels. The student gives his/her FOL proposal sentence and the system checks its syntax and whether it is the correct one (here equivalent sentences are acceptable). However, it does not provide any guidance about how to make that translation or even what is the kind of error made. The system in [11] does the same as the system presented here, but there are also significant differences that concern (a) the user interface, (b) the way it works internally for student interaction checking, (c) the way hints/help are/is structured and (d) the way new sentences are inserted.

3. A Structured and Interactive Process for NL to FOL Conversion

One problem in converting natural language into first order logic has to do with the unclear semantics that natural language has. Natural language has no clear semantics

as FOL has. However, the main difficulty comes from the lack of a systematic way of making the conversion. In a previous work [5], we introduced SIP (Structured and Interactive Process), a process that guides a student in translating/converting a NL sentence into a FOL one. SIP, which has been somewhat simplified here (to refer to simpler sentences, i.e. sentences that result in formulas with at most one group of formulas), is as follows:

1. Spot the verbs, the nouns and the adjectives in the sentence and specify the corresponding predicates or function symbols.
2. Specify the number, the types and the symbols of the arguments of the function symbols (first) and the predicates (next).
3. Specify the quantifiers of the variables.
4. Construct the atomic expressions (or atoms) corresponding to predicates.
5. Divide produced atoms in groups of the same level atoms.
6. Specify the connectives between atoms of each group and create corresponding logical formulas.
7. Form the group of formulas
8. Specify the connectives between formulas and create the next level formula.
9. Place quantifiers in the right points in the produced formula to create the final FOL formula

To demonstrate the steps of the above process, we present the conversion of the NL sentence "*All humans eat some food*" into a FOL formula in Fig. 1.

4. The Cognitive Tutor Authoring Tools (CTAT)

CTAT (Cognitive Tutor Authoring Tools) is an authoring tool for building tutors. Two types of tutors can be built using CTAT: example-tracing tutors and cognitive tutors [7, 8, 9, 10]. The first type is based on tracing specific pre-configured examples and requires no AI programming, whereas the second type requires AI programming and is based on a cognitive model, which is rule-based. Example-tracing tutors are easy to implement, but provide less flexibility, whereas cognitive tutors is quite more difficult to build, but can be quite more flexible. In this paper, we use CTAT for building an example-tracing tutor because, (a) it is easier to build, (b) we want to systematically analyze a large number of examples to extract possible cognitive patterns for building a cognitive tutor later, (c) we want to systematically analyze various types of hints or feedback needed.

Developing an example-tracing tutor in CTAT involves the following steps:

1. Creation of the graphical user interface (GUI).
2. Demonstration of alternative correct and incorrect solutions.
3. Annotation of the solutions steps with hint and feedback messages.

CTAT offers the *GUI Builder*, a tool for building the user interface of the tutor in the first step. GUI builder facilitates creating an interface in a graphical away without any programming, using a "recordable widget" palette added to Java NetBeans.

Additionally, CTAT offers *Behavior Recorder*, a tool for building "behavior graphs", which are graphs representing alternate correct and incorrect solutions to example problems, used in the second step above. For each problem, a corresponding behavior graph is created, which demonstrates student correct and incorrect problem solving behavior. Each such graph can be annotated with hints and feedback

messages, in step three. Hints concern correct links, whether feedback messages concern incorrect links. The steps on those graphs are associated with corresponding items of the user interface built in the first step.

Step 1. specify predicates/functions
"humans" → predicate: *human*
"food" → predicate: *food*
"eat" → predicate: *eats*

Step 2. Number, types and symbols of arguments

Predicate	Arity	Types	Symbols
human	1	*variable*	*x*
food	1	*variable*	*y*
eats	2	*variable, variable*	*x, y*

Step 3. Quantifiers
x → ∀
y → ∃

Step 4. Atoms
Atom 1: *human(x)*
Atom 2: *food(y)*
Atom 3: *eats(x,y)*

Step 5. Groups of atoms of the same level
Alternate 1
GroupAtom1: {*human(x), food(y)*}
GroupAtom2: {*eats(x, y)*}

Alternate 2
GroupAtom1: {*human(x)*}
GroupAtom2: {*food(y), eats(x, y)*}

Step 6. Connectives and formulas of groups
Alternate 1
GroupAtom1 → Form1: $human(x) \wedge food(y)$
GroupAtom2 → Form2: *eats(x, y)*

Alternate 2
GroupAtom1 → Form1: *human(x)*
GroupAtom2 → Form2: $food(y) \Rightarrow eats(x, y)$

Step 7. Connectives and formula of last group
Alternate 1
GroupForm1-1 → Form1-1:
$(human(x) \wedge food(y)) \Rightarrow eats(x, y)$

Alternate 2
GroupForm1-1 → Form1-1:
$human(x) \Rightarrow (food(y)) \Rightarrow eats(x, y))$

Step 8. Final formula
Alternate 1
$(\forall x)\ (\exists y)\ (human(x) \wedge food(y)) \Rightarrow eats(x, y)$

Alternate 2
$(\forall x)\ (\exists y)\ human(x) \Rightarrow (food(y)) \Rightarrow eats(x, y))$

Figure 1. Application of SIP process to "All humans eat some food"

At run time, CTAT's example-tracing engine implements the example-tracing function (or model tracing algorithm). This means that during the real-time use of the tutor, it maps the user problem solving behavior to the corresponding behavior graph and compares it with the one in the graph. Based on the results, the tutor provides either context-sensitive error feedback, when it matches an incorrect action link in the graph, or context-sensitive hints at student's request.

5. An Example-traced tutor for converting a sentence from NL to FOL

According to the process presented in Section 4, we first created the student interface of our system to reflect the NLtoFOL SIP process, as shown in Fig. 2. Actually a

separate student interface was implemented for each step or group of steps of the process. All those student interfaces were integrated into one interface as different step tabs, through which a student can try to convert a NL sentence into a FOL formula following the NLtoFOL SIP process. Each tab, except first, corresponds to a step of the NLtoFOL SIP process. The first tab ("Atoms") corresponds to steps 1-4 of the process. In each problem solving cycle the student follows the NLtoFOL SIP process selecting one tab at a time, selecting on it the interface elements to work on and performing a problem solving action.

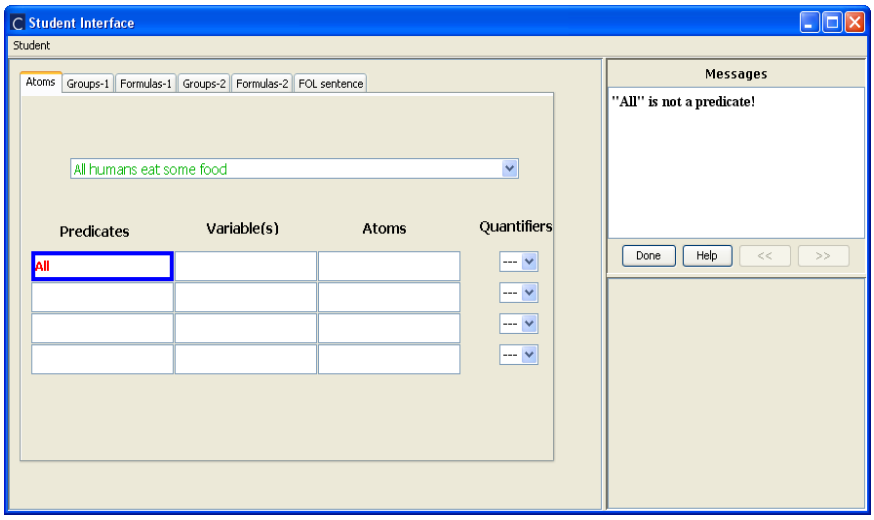

Figure 2. Student graphical interface

Then, we selected a number of NL sentences and for each of them we demonstrated both correct and incorrect problem-solving behavior, recorded by the Behavior Recorder, and as many behavior graphs as the sentences were created. Also, alternatives solution paths for the conversions of the same sentences, where applicable, were recorded as alternative solutions paths (see Fig. 3). These cases are used as the basis for Example-Tracing Tutors to provide guidance to students.

6. Feedback from the tutor

CTAT's Example-Tracing Engine uses the Behavior Graph to guide a student through a problem, comparing the student's problem-solving behavior against the graph. It provides positive feedback when the student's behavior matches steps in the graph, and negative feedback otherwise. If the student's input matches a link in the graph that was marked as an incorrect action, then any error feedback message attached to that link is presented to the student. When the student requests a hint, the hint messages attached to a link out of the current state in the graph are displayed. [15]

Tracing the student's step-by-step solution enables the tutor to provide individualized instruction in the problem solving context. Prototypically our tutor

provides immediate feedback on each problem solving action: recognizably correct actions are accepted and unrecognized actions are rejected.

The FOL tutor provides feedback on each problem solving action, by accepting correct actions, which is shown to the student by green color and tagging errors instead of accepting them, which is shown to the student by red color.

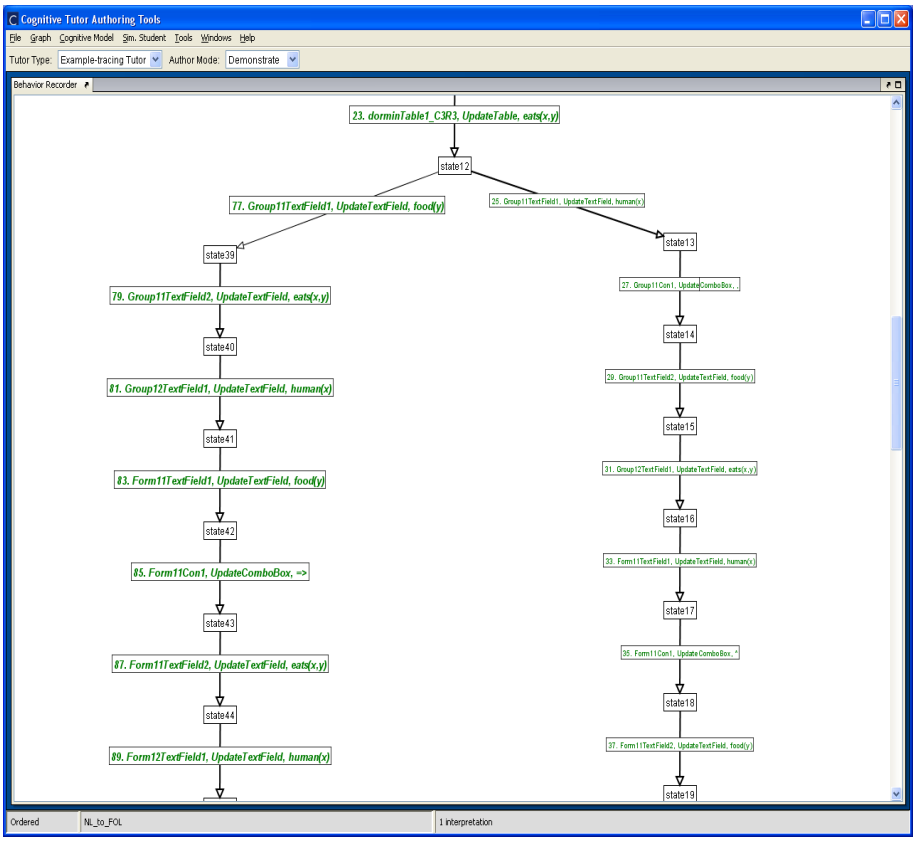

Figure 3. Behavior graph with two alternative conversions for the same sentence

7. Annotation of solution steps

7.1 Incorrect steps

In general any student input that is not recognized by the tutor is marked as incorrect, but *by defining incorrect steps* in the graph, the tutor will be able to provide a customized error feedback message for the specified input which. In each message we have included an example to demonstrate the correct use.

We focused to *common errors* that happen at the conversion of sentences from natural language to first order logic language, such as:

- Misuse of AND connective

- The order of quantifiers
- Use of function
- Grouping atoms of the same level
- Grouping formulas of the same level

We also demonstrated the tutor cases of *errors that are related to the sentence*. For example in the sentence "All humans eat some food", someone can characterize the "All" as predicates. In such errors the tutor gives feedback that is related to the theory of FOL language (see Fig. 2). The example-trace tutor accepts the answer in which the student fills *partially the right answer* e.g. fills the right predicate but in the wrong number ("humans" instead of "human").

7.2 Annotation of Hints

There are several factors that may affect the choice of a specific hint: tutoring topic, tutoring context, tutoring history, student's answer, and so on. First, to be pedagogically useful, a hint has to be related to the tutoring topic and be useful in helping the student find the expected answer. So the tutoring topic is important. [16]

The tutor also provides problem solving advice upon request of the student. We have implemented *four levels of advice* available for each problem solving action. *The first level* reminds or advises the student on the corresponding goal according to the SIP and a general description of how to achieve the goal. *The second level* provides a hint from the theoretical context of first order logic (definitions, syntactic etc) that is related to the corresponding step. The *third level* provides a hint specific to the case by providing a similar example. Finally, *the fourth level* provides concrete advice on solving the goal in the current context by suggesting the correct solution.

7.3 Knowledge labels

Once we completed the behavior graph, we added knowledge labels to links in the behavior graph, to represent the knowledge behind those problem-solving steps. This is a form of cognitive task analysis, since we determine how the overall problem-solving skill breaks down into smaller components. This process provided to us a way to copy hint messages from one step to a similar step. It also can be used by the PseudoTutor to do knowledge tracing whereby students' knowledge gaps can be assessed and the tutor can select subsequent activities to address those gaps. [9]

We have added the following knowledge labels to links in the behavior graph: *FindPredicate*, *FindArgument*, *SpecifyQuantifier*, *ConstructAtom*, *FormulateGroup*, *SpecifyConnective*, *ConstructFormula*. At the same time, it is a way of planning the cognitive model, since we intend in future work, to create production rules corresponding to each identified skill. [9]

8. Conclusions

We have implemented an example-trace tutor for the conversion of sentences in NL to sentences in FOL, according to the SIP. We also have created worked-out examples of conversions for several sentences of the same level with the Behavior

Recorder. The system has been preliminary used by a group of five students and the results are satisfactory. Four of them were satisfied in a large degree (>75%).

The examples can be used as the basis for Example-Tracing Tutors to provide guidance to students. The examples will also be used in future work, as planning cases and semi-automatic test cases for development of cognitive models, using the WME Editor and the Production Rule Editor, to create a production rule model.

References

1. Russell, S., Norvig, P.: Artificial Intelligence: a modern approach. 2nd Edition. Upper Saddle River, NJ, USA: Prentice Hall (2003)
2. Luger, G. F.: Artificial Intelligence: Structures and Strategies for Complex Problem Solving. 5th Edition, Addison-Wesley (2004)
3. Brachman, R. J., Levesque, H. J.: Knowledge Representation and Reasoning. Elsevier (2004)
4. Genesereth, M. R., Nilsson, N. J.: Logical Foundations of AI. Morgan Kaufmann, Palo Alto (1987)
5. Hatzilygeroudis I.: Teaching NL to FOL and FOL to CL Conversions. Proceedings of the 20th International FLAIRS Conf., Key West, FL, May 2007, AAAI Press, pp. 309-314 (2007)
6. Hatzilygeroudis I., Perikos I.: A Web-Based Interactive System for Learning NL to FOL Conversion. In: Ernesto Damiani, Jechang Jeong, Robert J. Howlett, and Lakhmi C. Jain (Eds.): New Directions in Intelligent Interactive Multimedia Systems and Services – 2, SCI 266, Springer-Verlag, pp. 297-307, (2009)
7. Aleven, V., McLaren, B. M., Sewall, J., Koedinger, K.R.: A new paradigm for intelligent tutoring systems: Example-tracing tutors. International Journal of Artificial Intelligence in Education, 19(2), pp. 105-154 (2009)
8. Aleven, V., McLaren, B. M., Sewall, J., Koedinger, K. R.: The Cognitive Tutor Authoring Tools (CTAT): Preliminary Evaluation of Efficiency Gains. In: the Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS-06), pp. 61-70 (2006)
9. Koedinger, K., Aleven, V., Heffernan, N., McLaren, B., Hockenberry, M.: Opening the door to non-programmers: authoring intelligent tutor behavior bydemonstration. Proceedings ITS-2004, pp. 162-174, Berlin: Springer (2004)
10. Koedinger, K. R., Aleven, V.: Toward a Rapid Development Environment for Cognitive Tutors. In U. Hoppe, F. Verdejo, J. Kay (Eds.), Proceedings of the 11th International Conference on Artificial Intelligence in Education, AI-ED 2003, pp. 455-457 (2003)
11. Abraham, D., Crawford, L., Lesta, L., Merceron, A., Yacef, K.: The Logic Tutor: A multimedia presentation. Electronic Journal of Computer–Enhanced Learning (2001)
12. Abraham, D. and Yacef, K.: Adaptation in the Web-Based Logic-ITA. In: De Bra, P., Brusilovsky, P., Conejo, R. (Eds). AH 2002, LNCS 2347, pp. 456–461 (2002)
13. Lukins, S., Levicki, A., Burg, J.: A tutorial program for propositional logic with human/computer interactive learning. In: SIGCSE 2002, pp. 381–385. ACM, NY (2002)
14. Alonso, J. A., Aranda, G. A., Martın.-Matceos, F. J.: KRRT: Knowledge Representation and Reasoning Tutor. In: Proceedings of EUROCAST 2007, LNCA 4739, pp. 400-407. Springer-Verlag, Berlin Heidelberg (2007)
15. Aleven, V., Sewall, J., McLaren, B. M., Koedinger, K. R.: Rapid Authoring of Intelligent Tutors for Real-World and Experimental Use. Proceedings of 6th IEEE international conference on advanced learning technologies (ICALT 2006), pp. 847-851 (2006).
16. Zhou, Y., Freedman, R., Glass, M., Michael, J. A., Rovick, A. A., Evens, M. W.: Delivering Hints in a Dialogue-Based Intelligent Tutoring System. Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), pp.128-134 (1999).