Int J Artif Intell Educ (2017) 27:475–514
DOI 10.1007/s40593-017-0139-y

ARTICLE

Assistance and Feedback Mechanism in an Intelligent Tutoring System for Teaching Conversion of Natural Language into Logic

Isidoros Perikos[1] · **Foteini Grivokostopoulou**[1] · **Ioannis Hatzilygeroudis**[1]

Published online: 24 February 2017

Abstract Logic as a knowledge representation and reasoning language is a fundamental topic of an Artificial Intelligence (AI) course and includes a number of sub-topics. One of them, which brings difficulties to students to deal with, is converting natural language (NL) sentences into first-order logic (FOL) formulas. To assist students to overcome those difficulties, we developed the NLtoFOL system and equipped it with a strong assistance and feedback mechanism. In this work, first, we present that feedback mechanism. The mechanism can provide assistance before an answer is submitted, if requested, but mainly it provides assistance after an answer is submitted. To that end, it characterizes the answer in terms of completeness and accuracy to determine the level of incorrectness, based on an answer categorization scheme, introduced in this paper. The automatically generated natural language feedback sequences grow from general to specific and can include statements on a student's metacognitive state. Feedback is provided as natural language sentences automatically generated through a template-based natural language generation mechanism. Second, we present an extensive evaluation of the effectiveness of the assistance and feedback mechanism on students' learning. The evaluation of feedback with students showed that full feedback sequences lead to greater learning gains than sequences consisting of only flag feedback and

✉ Isidoros Perikos
perikos@ceid.upatras.gr

Foteini Grivokostopoulou
grivokwst@ceid.upatras.gr

Ioannis Hatzilygeroudis
ihatz@ceid.upatras.gr

[1] Department of Computer Engineering & Informatics, School of Engineering, University of Patras, 26504 Patras, Hellas, Greece

bottom-out hints ($n = 226$), and that generic, template-based feedback sequences are comparable to the utility of problem-specific hints generated by human tutors ($n = 120$).

Keywords Feedback framework · Student assistance · Answer categorization · Feedback sequencing · Error categorization · Feedback effectiveness evaluation · Learning analytics

Introduction

A great variety of different learning systems exist that formulate what is called e-learning, such as Learning Management Systems (LMS), Computer Assisted Instruction (CAI) systems, Intelligent Tutoring Systems (ITS), Adaptive Educational Hypermedia Systems (AEHSs). ITSs constitute a popular type of educational systems that are becoming a fundamental means of education delivery, leading to impressive improvement in student learning (Aleven et al. 2009). Their main characteristics are that they provide instructions and feedback tailored to the learners and perform their main tasks based on Artificial Intelligence methods. The development of tutoring systems and personalized learning environments, especially those developed for the web, will be among the most important active research areas in the next decade (Narciss et al. 2014b). Nowadays, more and more universities and educational institutes develop and use ITSs to assist students in learning and tutors in teaching and managing courses, supervising students and mining students' performance data for the best advantage of them. ITSs have been used with great success in many challenging domains to offer individualized learning to students and give them the capability to learn in their own way and at their own pace (Mitrovic et al. 2007, 2009; VanLehn et al. 2005). A fundamental aspect of a tutoring system concerns its interaction with the learners and the ability to efficiently assist them during the learning process. To that end, it is necessary to provide personalized guidance and feedback tailored to the learner's needs.

As feedback is considered any message or display that a system presents to learners, after a request or a response (Wager and Wager 1985) and constitutes a key aspect of effective learning and instruction (Mory 2004). Feedback can vary in type, time, the amount of the provided information as well as the way it is offered (McKendree 1990). Giving semantically rich and personalized feedback to students requires high content and domain expertise, which is a challenging, time intensive and demanding task for tutors. So, in many cases it is not possible for tutors to provide detailed feedback and guidance to students on time (Black and Wiliam 1998).

A major advantage of ITSs is that they can provide students with automatically customized feedback. They integrate capabilities of generating and delivering various types of feedback, adapted to the answers of the students (Lopez 2009; van der Kleij et al. 2012). Formative feedback in educational systems has always been viewed as a critical factor for the improvement and the construction of knowledge and also for the acquisition of skills (Shute 2008). From a learner's perspective, when a student is developing his/her expertise on a topic, it is imperative that he/she is aware of his/her level of understanding (Berlanga et al. 2012). Also, feedback provides a way to help learners improve their understanding, recognize and fix their gaps and inconsistencies

in their acquired knowledge (Hattie and Timperley 2007) and be more deeply engaged with educational processes and materials.

ITSs have the ability of generating and delivering to students a wide variety of feedback types. Feedback in tutoring systems needs to be formulated, framed and delivered in such a way that attracts learners' active engagement and critical thinking (Havnes et al. 2012). Based on the level of the information that is conveyed in the feedback, the common formative feedback can be distinguished into *Knowledge of Result*, *Knowledge of Correct Response* and *Elaborated Feedback* (Dempsey et al. 1993; Shute 2008). The most common and basic kind of feedback is KR, which informs students whether their answers are correct or incorrect. KR can assist students to evaluate their success rate, but it doesn't provide principle-based explanations about the reasons of their answers' correctness or incorrectness (Moreno and Mayer 2007). So, *Knowledge of Result* feedback is in most cases coupled with other types of feedback, such as *Elaborated Feedback*. Feedback in tutoring systems should provide students with different levels of verification and elaboration in order to enhance its effectiveness (Kulhavy and Stock 1989). Verification concerns the confirmation of whether a student's answer is correct or not, while elaboration addresses analysis of the answer and related topics, discusses particular error(s) and guides the student towards the correct answer (Shute 2008). The level of elaboration and verification determines the amount of the provided information.

In addition to the type and the information of the feedback, the timing of the feedback delivery to the learners can affect its effectiveness (Corbett and Anderson 2001; Mathan and Koedinger 2003). Based on the delivery timing, feedback can be distinguished into *immediate* and *delayed*. Immediate feedback is typically provided immediately after submitting an answer, while delayed feedback is provided with some delay, after a block of items are completed, after a few minutes or even longer (Dempsey and Wager 1988). In general, immediate feedback can promote efficiency during training, while delayed feedback might lead to better retention and transfer performance (Mathan and Koedinger 2003). In the literature, there are many research studies on the effectiveness of the different types of feedback on learning (Mory 2004; Shute 2008; van der Kleij et al. 2012; Narciss et al. 2014b). The delivery of proper feedback is a very challenging task, because many educational, learner and situational parameters can facilitate or hinder the effect of feedback on learning (Narciss et al. 2014a).

In this paper, we present a sophisticated and strong feedback mechanism that effectively deals with the difficulties met by students in learning a difficult artificial intelligence process, namely conversion of natural language sentences into logic formulas. Through it, we actually present a general framework for modeling system assistance to students. This framework includes, first, a systematic and generic modeling of students' answers. Also, it includes a domain specific error categorization scheme, which is utilized for recognizing the types of errors made by the students and for guiding the feedback delivery mechanism. Additionally, an assistance generation scheme is introduced that models and delivers feedback in an incremental way, called *level-based* mode. Furthermore, a general approach for the generation of textual hints is presented. Finally, we explore the complex nature and sources of feedback and investigate the learning effect of different feedback forms in the context of a web-based

intelligent tutoring system for teaching the above mentioned artificial intelligence process.

The rest of the paper is structured as follows. "Background" section presents background knowledge on the conversion of natural language into logic. "Feedback Framework" section introduces the feedback framework utilized in the tutoring system and analyzes the parts it consists of. Initially, the framework's approach to model students' answer is illustrated and then the domain specific error categorization scheme is described. After that, the feedback sequencing hierarchy is illustrated. Next, the approach for generating, in an automated manner, the textual feedback messages to be delivered to the learner is presented. "Evaluation" section, presents the evaluation study conducted and the results gathered in real classroom conditions. Finally, "Method" section concludes the paper and provides directions for future work.

Background

Existing Status in Teaching Logic Formalization

Artificial Intelligence (AI) is considered to be an important domain in computer science, but also, according to our experience, a very hard domain for students to master. Many tutors acknowledge that Artificial Intelligence courses contain complex concepts, which are difficult for the students to grasp. Knowledge representation and reasoning is a fundamental topic of artificial intelligence. A basic knowledge representation 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. So, teaching FOL as a knowledge representation and reasoning language is a vital aspect, which includes teaching a number of concepts and processes. One of them is converting natural language (NL) sentences into FOL formulas, a process called *logic formalization* of natural language. It is an ad-hoc process, ie there is no specific algorithm that can be automated within a computer. This is mainly due to the fact that natural language has no clear semantics as FOL does. Most of existing textbooks do not pay the required attention to the above process. They simply provide the syntax of FOL and definitions of the logical symbols and terms (Russell and Norvig 2010). Even more specialized textbooks (Brachman and Levesque 2004) do the same. At best, they provide extended explanations and examples (Genesereth and Nilsson 1987). However, they do not provide any systematic guidance towards it. Given the above, students usually find difficulties in learning the process of formalizing NL sentences into FOL and rely on tutors' teaching experience. On the other hand, existing tutoring systems that deal with teaching logic either not deal with the formalization process (Sieg 2007; Yacef 2002, 2005) or do not offer a systematic teaching way and do not provide adapted feedback and guidance to the students (Alonso et al. 2007; Moreno and Budesca 2000).

In most cases, conversion [1] of NL sentences into FOL is taught by means of examples presented by the tutor. The students practice formalizations by solving

[1] We use the terms 'formalization', 'conversion' and 'translation' interchangeably with regards to the NL to FOL conversion process.

exercises on paper. However, it is impossible for a tutor to provide meaningful and personalized feedback to every student in the class, due to limited time. So, feedback in most cases is delayed and is provided to the students after hours or even days. In addition, exercises may have many correct answers that are equivalent and so, it is complex to monitor and specify students' solution paths while working on exercises and constructing their answers (Maestro-Prieto and Simon-Hurtado 2015).

To assist students in learning NL formalizations, ie how to manipulate sentences and convert them into logic, the process presented in (Hatzilygeroudis 2007) was proposed. It is called the NLtoFOL Structured and Interactive Process (SIP).[2] It guides students how to formalize a NL sentence in a step-based way. The conversion process is split into specific sub-processes/steps, where each step implements a specific part of the conversion. In this way, students have to implement a specific procedure of the translation process in every step and thus form the FOL formula in a constructive, part-based approach. Moreover, SIP can help students get a deeper understanding of the conversion process and also face errors and conversion misconceptions.

Afterwards, a web-based interactive system for learning natural language formalization (Hatzilygeroudis and Perikos 2009), based on the NLtoFOL SIP, was developed. The system offers guidance and assistance to the students during the learning sessions via a simple feedback mechanism, which adapts to each student's answers, performance and behavior in some degree. A small scale evaluation of the system showed satisfactory performance of the system. The feedback mechanism presented in this paper is a large scale extension of the one outlined in (Hatzilygeroudis and Perikos 2009), as developed through the past years. The system has been used in our department and a large scale evaluation has demonstrated very satisfactory performance concerning its tutoring capabilities and the learning construction and performance of students.

Challenges and Complexity of Formalizations

The conversion of NL sentences into the corresponding FOL formulas is acknowledged by many logic tutors to be a cognitively hard and complex procedure for the students to understand and successfully implement (Barker-Plummer et al. 2009; Hatzilygeroudis 2007). Moreover, studies on students' performance indicate formalization to be an error prone process resulting in low student performance (Barker-Plummer et al. 2008, 2011, 2012). The above pose two interlinked questions: (a) why formalizations are so hard for the students? and (b) what we can do to help students to gain a deeper understanding of this process?

To answer the first question, the NL sentence, the FOL formula and the conversion process have to be analyzed. There are many factors that have an impact on the formalization difficulty of NL sentences. A main factor is related to the fact that natural language tends to have no clear semantics, in contrast to FOL. In fact, written and, mainly, spoken NL sentences tend to be ungrammatical and informally composed, suffering from ambiguity, which adds difficulty to their formalization. So, determination of the elements constituting the corresponding FOL formula may not be explicit enough in a sentence.

[2] An overview of the NLtoFOL SIP conversion process is presented in Appendix A.

In some cases, sentences may refer to abstract entities, expressed by words like “someone” or “something” or similar, and thus determining the arity and the arguments of predicates in FOL may not be clear to students. Moreover, the quantification of the predicates denoting the entities and the objects of the sentence may not be clear and lapidarian. For example, in the sentences “All animals eat some plants” and “I like someone if he is vegetarian”, the similar words “some” and “someone” lead to different quantification, given that in the second sentence “someone” actually means “all” (vegetarians).

This also stands for the determination of the polarity of the predicates. Words denoting negative polarity may appear in a NL sentence. They cause additional difficulty and complexity for students to decide on (1) their appearance or not in the FOL formula and (2) their scope, ie whether a negation affects an atom, a group of atoms or even a whole formula. For example, the sentence “There are no mushrooms that are poisonous and purple” can be erroneously translated as “$(\forall x)$ mushroom(x) $\Rightarrow \neg$poisonous(x) $\wedge \neg$purple(x)” instead of “$(\forall x)$ mushroom(x) $\Rightarrow \neg$(poisonous(x) $\wedge$ purple(x))”. Determination of a negation's scope may be tricky for students.

Another case of ambiguity concerns determination of connectives. So, existence of an “and” in a sentence may not lead to the use of “$\wedge$” connective. For example, the FOL formula corresponding to the sentence “Apples and oranges are fruits” is often determined as “$(\forall x)$ (apple(x) $\wedge$ orange(x)) $\Rightarrow$ fruit(x)”, which is wrong, because nothing is both an apple and an orange.

The aforementioned remarks and examples indicate that an accurate and precise translation of a NL sentence into FOL may be complex and tricky, making it difficult for students to understand and implement. Additionally, the lack of a systematic way to do the translation makes formalization a conceptually difficulty and error prone process.

To successfully answer the second question above, we have to look deeper at the factors concerning the formalization process. First, something should be done to help students to understand the nature of the conversion process. This requires looking deeper at the conversion, modeling and dividing it in subprocesses, to make it conceptually easier and less complex for students to understand and implement. So, we modeled the process as a constructive step-based procedure, where each step implements a clear and specific part of the conversion, and proposed a structured and interactive process (SIP) (Hatzilygeroudis 2007), which is presented in Appendix A. In addition, from the tutor’s perspective, for the innate natural language aspects, he/she cannot do anything to help students apart from providing some explanations about the sentence semantics. However, during the learning sessions, the tutor can provide valuable feedback to the students facilitating and cultivating an efficient learning. For example, the tutor can explain to each student the errors made and why his/her answer is not correct and also provide directions about how to correct it.

Based on the above, we developed and used the NLtoFOL tutoring system to assist students in learning formalizations by using the NLtoFOL SIP process and additionally by offering guidance and meaningful feedback, through a sophisticated mechanism, to the students during the learning sessions.

Feedback Framework

Feedback is provided to users via two components: Error Detection unit (ED) and Feedback Generation unit (FG), as presented in Fig. 1. The error detection unit is used to analyze a student's answer and recognize the types of the errors made. Initially, it characterizes an answer in terms of accuracy and completeness. Then, in case the answer is not correct, it recognizes and analyzes the errors made, based on a domain specific error categorization schema. The feedback generation unit is used consequently to automatically generate and deliver various types of feedback messages based on the error(s) made by the learner. It uses a hierarchy of assistance levels, where each one is associated with specific types of feedback and hints. The feedback sequencing follows an incremental assistance delivery approach.

Feedback Delivery Cycle

The feedback delivery cycle is as follows (processes like error detection, answer categorization and feedback generation are presented in detail in the following sections):

1. As soon as an answer is submitted, forward it to the error detection (ED) unit to analyze and categorize it in one of the answer categories.
   - 1.1 If the answer is correct (complete and accurate), proceed to the next step
   - 1.2 Otherwise, forward the answer to error categorization mechanism to determine the type(s) of error(s) made.
2. Based on the feedback assistance scheme and the student's answer specify the type and the parameters of the feedback to be provided.
3. Forward the student's answer, the recognized error type(s), the proper feedback type and its parameters to the feedback generation (FG) unit, to generate the proper feedback message(s) as follows:

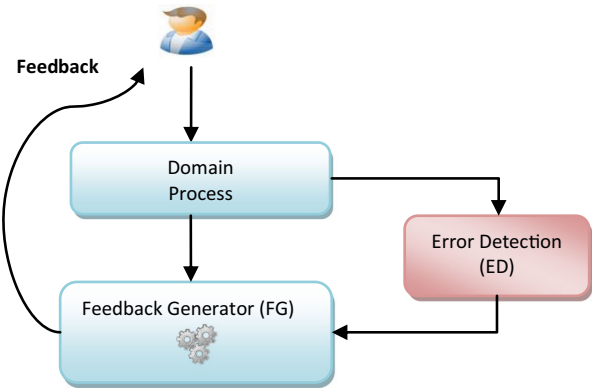

Fig. 1 An overview of the feedback framework

- For each error type, analyze the exercise to define the cause(s) of the error and create the proper explanation(s).
- For each element that is correct, generate the proper positive feedback message.

4. Produce the appropriate textual messages based on the template-based approach.
5. Deliver the feedback messages to the student according to the feedback hierarchy and in the proper way (ie immediately or on demand).
6. Allow student to provide a new answer and go to step 1.

Types of Feedback in the System

There is extensive research on the types of feedback and the functionality of each type in tutoring systems (Hattie and Timperley 2007; Mandernach 2005). Findings indicate that different types of feedback have different effects on the learning process of a student (Koedinger and Aleven 2007; Mory 2004; Vasilyeva et al. 2008). Providing feedback and hints in every step of a multi-step procedure is an essential feature of a tutoring system (Gheorghiu and VanLehn 2008). In general, feedback offered to students, in order to be effective, has to be error specific and tailored to the student's needs in terms of performance and knowledge gaps. The feedback messages provided by the system to a student's answers and actions can vary in type, in the information that they contain, in timing and also in the way they are presented. A categorization of the feedback types that can be provided to the students has been developed. The categories and the types of feedback have been determined combining widely used classifications of feedback (Narciss 2008; VanLehn 2006). The main types of feedback offered by the feedback framework are the following:

Minimal feedback. The system informs the student whether the answer is correct or not. Giving just a correct or incorrect notice can dramatically improve the efficiency of learning (Wang and Wu 2008).

Flag feedback. Initially, after a student has submitted an answer to a step, the system informs him/her whether the answer is correct (colored green) or incorrect (colored red). Flag feedback (Anderson et al. 1995) provides a binary "flag" rather than further information and is considered to be a special kind of minimal feedback.

Positive feedback. The system can inform the student about the correct parts of the answer and provide corresponding justifications on them. Providing positive feedback can help the students know that they are on the right way and also help them reinforce the knowledge they have.

Knowledge about concepts. The system provides hints and explanations on terms and processes and also hints and explanations of the appropriate conceptual context.

Procedural feedback. The system tries to help the students how to proceed towards the solution by providing hints on what has to do for the sentence's formalization and also what to do next, suggesting the concepts to be used or rules to be applied.

Error-specific feedback. When a student's answer is incorrect the mechanism recognizes the errors made and provides the proper feedback based on the errors, taking into account the formalization and the student model.
Bottom-out hints. The system can decide to give to the student the correct answer or part of it. This can be done after a student's request or after constant failures and special learning circumstances.
Knowledge on metacognition. The system analyzes students' behavior during the learning sessions and provide metacognitive guiding and hints.

Error Detection

The error detection (ED) unit is responsible to detect and recognize the errors made by a student. Just after the student submits an answer, the error detection unit is activated to handle and analyze the student's answer. The process of error detection is a fundamental task of an intelligent tutoring system. The pedagogical assumption is that the motivation for understanding and learning is getting stronger when an error is detected (Hirashima et al. 1998). The error detection unit performs two basic operations. First, it characterizes the student's answer, in terms of completeness and accuracy, and categorizes it in the proper answer type category. Then, in case that the answer submitted by the student is not correct, determines the number and the type of errors made, utilizing a domain specific error categorization scheme.

Answer Categorization

The analysis of a student's answer is quite important for assessing it as accurately as possible and also assists the generation of the proper feedback message(s). Therefore, a generic answer categorization was developed to help the analysis. In this context, a student's answer is characterized in terms of completeness and accuracy (Fiedler and Tsovaltzi 2003). An answer is characterized as *complete* if all the *elements* of the correct answer appear in the student's answer. In contrast, when the student did not specify all the elements of the correct answer, his/her answer is characterized as *incomplete*. The student's answer is characterized as *accurate* when all the elements of the student's answer are correct, while is characterized as *inaccurate* when an element (or more) of the student's answer is not defined correctly. The categorization scheme developed is also enriched in terms of superfluity (Gouli et al. 2006). The superfluity is used to characterize a student's answer that contains all the required elements plus one or more elements that are unnecessary. The term "element" is used to represent any individual piece of information that the student has to specify in an answer. So, a student's answer can be characterized as:

- *Incomplete – Accurate*: An answer with missing elements. However, the existing elements are correct.
- *Incomplete – Inaccurate*: An answer with missing elements, where one or more of the existing elements are incorrect.
- *Complete – Inaccurate*: An answer that includes all needed elements of a correct answer. However, one or more of them are incorrect.

- *Complete - Accurate* (correct answer): An answer that includes all the elements of a correct answer and all are correct.
- *Superfluous*: An answer that includes more elements than needed and one or more elements appear to be incorrect. This can be further classified as:

  - *Superfluous – Accurate*: the required elements are defined correctly, however the presence of the extra element(s) make the answer incorrect.
  - *Superfluous – Inaccurate*: one or more of the required elements are incorrect.

In case that a student's answer is characterized as complete and accurate, the student has specified the correct answer or a correct answer, in case that more than one answers are correct. In such a case, the current step of the formalization process has been completed successfully and the student can proceed to the next one. In all other cases, the student's answer is considered as incorrect and the student has to work towards the correct one. The student's answer is further analyzed by the error detection unit, in order to recognize the types of the errors made based on a domain specific error categorization scheme.

In the context of the NLtoFOL system and based on the step of the NLtoFOL SIP process, an *element* may be a word representing a predicate (e.g. city, dog-catcher), a variable (e.g. x, y), an atomic expression (e.g. city(x)), a connective (e.g. $\wedge$, $\Rightarrow$) etc. For example, in the first step of the SIP process, elements of the student's answer are the words specified by the student to represent the basic predicates and functions of the sentence.

Error Categorization

In order the error detection (ED) unit be able to specify the errors made in an incorrect answer, we have designed a domain specific error categorization scheme. This error categorization scheme has been developed based on (a) an extended analysis and mining of the errors made by students and (b) general rules and constraints coming from principles of the domain. Based on the above, the types of possible errors are classified according to the scheme presented in Table 1.

The possible error types are organized in six major categories. Each category includes a collection of different error types. More than one type of errors may be involved in a student's answer at each step of the process.

An Example Case in NLtoFOL System

As an example, we present handling of some indicative students' answers during the process of converting the exercise-sentence "Every city has a dog-catcher that has been bitten by every dog living in the city" into FOL via the NLtoFOL system.

In the first step of the process (see Appendix A), students must specify the predicates and the functions in a sentence. For the example sentence, let's suppose that a student determines three predicates: "city", "dog" and "lives-in". The determined predicates are correct, but the student has not specified all the predicates of the sentence. Thus, the system characterizes the student's answer as *Incomplete-Accurate.* Suppose that another student determines six predicates which are: "city", "dog-catcher", "dog", "lives-in", "has" and "is-bitten". The student's answer is superfluous, because the student has

Table 1 Error categorization of the natural language formalization process

Error category	Error type	Error explanation
Predicate error	Number Error	Plural number of a noun instead of singular is used. E.g. “men” instead of “man”.
	Person Error	Another person instead of singular third-person is used. E.g. “eat” instead of “eats”.
	Semantics Error	A wrong word (representing e.g. a function or a constant etc.) is used as predicate.
Term error	Type Error	A wrong type is specified for an argument of an atom. E.g. “constant” is specified instead of “variable”.
	Semantics Error	A word denoting a predicate is used as a function symbol (e.g. “brother-of” is used as a function symbol) or a constant etc.
Argument error	Cardinality Error	The number of the arguments of an atom is wrong.
	Order Error	The order of the arguments of an atom is wrong.
Connective error	Type Error	One connective (e.g. “∧”) is used instead of another (e.g. “⇒”) to connect the atoms of a formula or a negation is missing.
	Scope Error	The scope of a connective (e.g. of a negation) is wrong.
Quantifier error	Type Error	The type of a quantifier is wrong.
	Scope Error	The scope of a quantifier does not cover all occurrences of its variable.
	Order Error	The order of quantifiers is wrong.
Group error	Atoms Error	The atoms denoted to form a group are wrong.
	Order Error	The order of the atoms denoted to form a group is wrong.
	Cardinality Error	The number of the atoms forming a group is wrong.
	Number Error	The number of the created groups is wrong.

specified one more predicate than required. Additionally, it is inaccurate since the predicates “has” and “bitten” are specified in an incorrect way. Thus, the answer is characterized as *Superfluous-Inaccurate*.

In the second step of the process, students have to specify the number, the type and the symbol of each argument of the function symbols and of the predicates. Let’s suppose that a student has given the answer presented in Table 2. The system characterizes the answer of the student as *Complete-Inaccurate. Complete* because the student has specified all the elements of the correct answer and *Inaccurate* because two elements of the answer are not correct. Then, the answer is analyzed by the error detection mechanism to recognize the errors made. The mechanism recognizes that the student has defined the predicate “dog” incorrectly. More specifically, the student has made an *argument error* and even more specifically a *cardinality error*, having defined two arguments instead of one. Also, the student has made an *order error* for the predicate “lives-in”, by having determined a reverse order for its variable arguments.

Let’s see an example student answer for the third step, where the quantifiers of the variables are specified: “x → ∃”, “y → ∃”, “z → ∀”. The student’s answer is complete

Table 2 Student answer and correct answer for step 2 of process

Predicate/Function	Arity	Argument types	Symbols	
Student answer (SA)				
city	1	variable	x	
dog-catcher	2	variable, variable	y, x	
dog	2	variable, variable	x, z	
lives-in	2	variable, variable	x, z	
has-bitten	2	variable, variable	y, z	System's response
Correct answer (CA)				Correct ✓
city	1	variable	x	Correct ✓
dog-catcher	2	variable, variable	y, x	Incorrect ×
dog	1	variable	z	Incorrect ×
lives-in	2	variable, variable	z, x	Correct ✓
has-bitten	2	variable, variable	y, z	

because all the elements of the correct answer are included, but it is inaccurate since the quantifier of variable "x" has been defined incorrectly. The student has defined "x → ∃" instead of "x → ∀", which is the correct one. Thus, the answer is characterized as *Complete-Inaccurate*. The error detection mechanism recognizes that the student made a *quantifier error*, more specifically a quantifier *type error*. In the sixth step of the process, the student must specify the connectives between atoms of each group and create the corresponding logical formulas. A student's answer has been the following:

AtomGroup 1 → Form 1: *city(x),* AtomGroup 2 → Form 2: *dog-catcher(y, x)*
AtomGroup 3 → Form 3: *dog(z)* ∨ *lives-in(z, x)*

The system characterizes the answer as *Incomplete-Inaccurate.* The student has failed to determine all the atom groups of the sentence and thus his/her answer has been incomplete. The answer is also inaccurate because an atom group has been identified incorrectly. The error detection mechanism recognizes a *connective error*, because the connective "∨" specified by the student for the AtomGroup3 is incorrect. Furthermore, the answer is incomplete, because the student hasn't specified the necessary AtomGroup4: "*has-bitten(z, y)*". Finally, consider as an example answer, in Step 10, the following:

"*(∀x)City(x) ⇒ ((∃y) dog-catcher(y, x) ∧ (∀z) (dog(z) ∧ lives-in(z, x)) ⇒ has-bitten(z, y))*"

The system categorizes the answer as *Complete-Accurate.* The answer is characterized as complete, because the student has defined all the elements of the correct FOL formula, and 'accurate', because all defined elements are correct. So, this answer is a correct one.

Feedback Sequencing

The system's assistance is structured to consist of *levels* of assistance, each one giving increasingly more specific advices. Each assistance level provides students with specific type(s) of feedback. The system is designed to implement an incremental assistance and feedback delivery policy. Initially, it starts by delivering feedback associated with the first level of assistance, scaling up its assistance afterwards to the final level, where the correct answer is offered. The feedback sequence has been designed based on feedback methodologies and hinting strategies (Fiedler and Tsovaltzi 2003; VanLehn 2006; Zhou et al. 1999). In addition, to enhance the system's assistance, a deeper analysis of the tutor's behavior during the learning sessions of the formalization process in real classroom conditions has been performed. More specifically, the system analyzes an answer and specifies the feedback to provide, trying to model and simulate the way a tutor does it. A human tutor is considered to be extremely effective at providing hints and guiding students learning. A key aspect of the tutor's behavior is that he/she does not reveal the solution directly, but tries to engage students in more active learning and thinking. Furthermore, research studies underpin that computer generated hints can effectively assist students in a similar degree to a human tutor (Muñoz-Merino et al. 2011; VanLehn 2011). The system's assistance can be provided in stages, (i) before the student submits an answer to a formalization step of an exercise and (ii) after the student submits an answer that is not correct.

Assistance Before Answer Submission

A student can request and get assistance for the current formalization exercise's step, while working on it and before he/she submits an answer. The assistance schema before the student submits an answer includes two levels. Each level is associated with specific types of feedback, each one of which is associated with specific hints/explanations. The assistance levels, the feedback types and the hints are presented in Table 3.

The first assistance level provides hints implementing *knowledge about concept* feedback. These hints can address the characteristics of the involved theory topics and aim to remind learner about the theoretical context (definitions, syntax etc.) related to the current step. For example, hints may present to the student specific parts of theory such as "Predicates representing nouns of the natural language should be in single person while predicates representing verbs should be in third person", "When using a quantifier, it must be associated with a variable and every district variable should be quantified by the proper quantifier".

Table 3 Assistance schema before answer submission

Assistance level	Feedback type	Explanation
First level	Knowledge about Concept	Hints indicating the involved relevant theory topics.
		Hints/explanations of concept definition.
Second level	Procedural feedback	Hints how to work towards the sentence formalization.
		Hints what to do first/next.
		Present similar exercises that have completed before.

At the second level of assistance, *procedural* feedback is provided to the student. The hints at this level aim to remind the students of the corresponding learning goal and provide specific guidance on how to achieve it. For example, hints may present to the student, the main directions of how to work and what to do first on his answer and hints may denote that "In order to specify the functions and the predicates, you need to specify first the word representing the functions and then the words representing the predicates". Also, they aim to focus student's attention on specific and peculiar elements of the exercise's current step process that may be tricky, leading to high error rates and failing attempts. Furthermore, the system at this level can provide the student with similar exercises that has successfully completed before. The hint messages are tailored to the characteristics of the current exercise's formalization step and are associated with the hint button. They are displayed after a student's request for assistance, ie they are on-demand help.

As an example, let's consider again the sentence "Every city has a dog-catcher that has been bitten by every dog living in the city". A first level assistance during the first step of the process is a *knowledge about concept* feedback message that reminds student of some general principles and relevant theory involved in current SIP step, as presented in Fig. 2.

Assistance After an Incorrect Answer Submission

The second stage of the system's assistance is offered after the student submits an answer that is not correct. After an error occurrence and its recognition by the error detection unit, the feedback generation unit takes over and generates the proper feedback messages. The assistance schema consists of four levels of assistance, which are presented in Table 4.

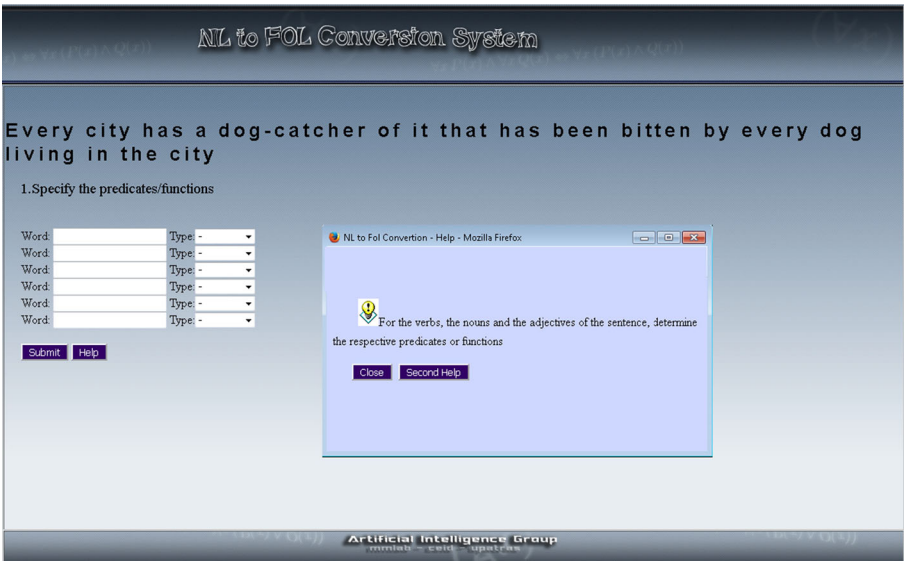

Fig. 2 An example hint for the first step of an exercise

Table 4 Assistance schema after an incorrect answer

Assistance level	Feedback type	Explanation
First level	Flag feedback	Correctness notice for one element of the answer each time.
		Correctness notices for all the parts of the answer.
Second level	Positive feedback	Analysis and justification on correct answers.
	Error-specific feedback	Answer analysis and error explanations.
Third level	Procedural feedback	Hints how to work towards the sentence formalization.
		What to do next.
		Present similar exercises that have completed before.
Fourth level	Bottom-out	Reveal one element of the correct answer each time.
		Reveal the correct answer.

The system, at the first level of assistance, informs the student about the correctness of his/her answer by coloring green the correct and red the incorrect elements of the answer. This type of feedback (*flag feedback*) can be offered to the student in two different ways. First, after the student's answer is analyzed and the correct and incorrect elements are determined, the system can provide incremental flag feedback by coloring one by one the answer elements by the proper color denoting its correctness or incorrectness. So, the student can incrementally request for additional correctness notice for the next element of his/her answer. In this way, the amount of the feedback delivered is handled by the student and can be just as much as the student needs. In the second way, flag feedback can be offered all at once, by coloring all the elements of the student's answer with the proper color. In this way, the student gets the full functionality of the flag feedback, getting the correctness notices for each element of the submitted answer. The flag feedback is offered part-by-part in case the student's answer consists of many parts and two or more of its parts are incorrect or missing.

The second level of assistance mainly offers specific justifications for the correct elements of a student's answer and proper explanations for the errors made. Initially, hints are presented based on the type of the student's answer. More specifically, the system can inform the student about the type of the answer provided in terms of completeness and superfluity. For example, hints may denote that "Some predicates are not denoted/missing" or that "You have determined more function symbols than needed". After that, the system offers specific hints about each error. The student has already been informed about the correctness of each element of his/her answer by the assistance of the previous level and now the system provides hints justifying the correct elements and explaining each one of the errors made. Indeed, student's newly acquired knowledge may have high probability of uncertainty and hence could benefit from confirmation through positive feedback (Mitrovic et al. 2013).

The third level of assistance provides the student with procedural hints. First, the system's third level hints can remind student of the step's goal and provide hints of how to correct his/her answer. For example, hints can address specific errors made such as "Word love should be in third person", "Word dogs should become a predicate". In addition, the system based on the student's exercise history may present similar exercises that the student has completed successfully along with the corresponding correct answers. For example, a hint may denote that "A similar exercise that you have

completed successfully was the sentence [exercise remark] where the correct answer to the current step was [exercise step answer]". The aim of this kind of hints are to provide the proper similar and analogous exercises and provide clues of how to fix their answers and determine the correct answer.

The fourth assistance level provides the student with the correct answer or a part of it. The correct answer can be offered in two ways. First, after a student's request for assistance, the system can reveal an element of the correct answer. After having seen some elements of the correct answer, the student can work towards the correct answer. Second, the system can offer all the elements of the correct answer at once. The correct answer is offered to the student part-by-part in case that the answer consists of many parts and two or more of the parts of the student's answer are incorrect or missing. In addition, the student receives proper explanations and then he/she can proceed to the next step of the sentence's formalization process.

For example, for the first step of the conversion process of the sentence "Maria loves all dogs that love their master", the student has specified three predicates ("loves", "dogs", "love") and one function ("master-of"). The system has colored green the correct parts ("loves", "master-of") of the answer and red the incorrect ("dogs", "love"). The student can request additional assistance. In Fig. 3, a second level assistance message providing error-specific feedback is presented.

Metacognitive Assistance

In most tutoring systems, the learner holds the control of assistance. Thus, the learner must decide how to use the system's assistance, something that requires that the learner is in position to make good judgments about his/her own knowledge and that is also

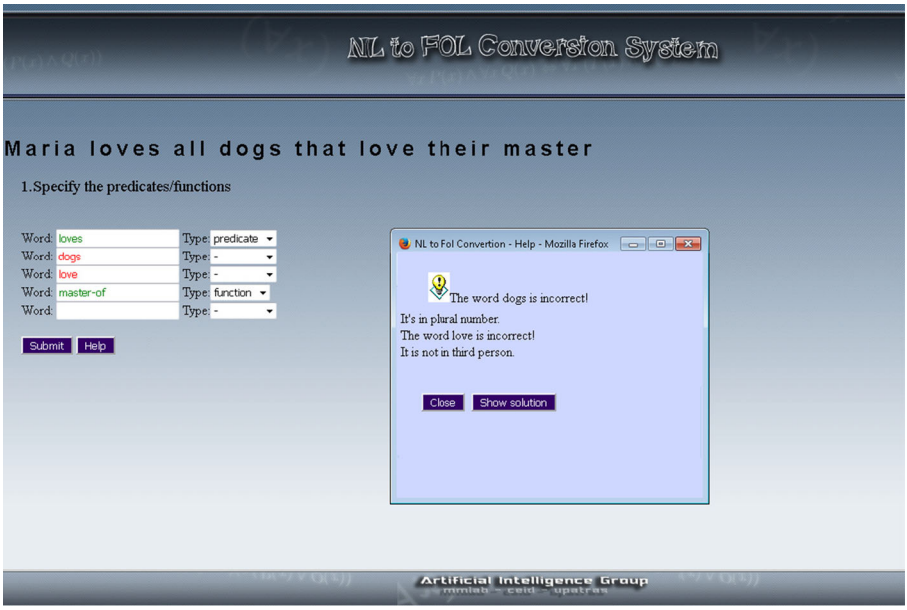

Fig. 3 Flag feedback and error analysis for the student's answer during step 1

able to judge when he/she can benefit from assistance (Aleven and Koedinger 2000). Effective help seeking behavior can result in better learning in educational systems (Aleven et al. 2016; Goldin et al. 2013; Roll et al. 2011). However, in many cases, a learner may not use the assistance of tutoring systems in the proper way, something that can diminish the learning capabilities of the tutoring process (Aleven et al. 2003, 2006a, b). So, a special part of the system's interaction with the student is based on the student's metacognitive learning behavior. Modeling students' metacognitive skills and using these models to scaffold students learning, can lead to more effective and efficient learning (Mathan and Koedinger 2005; Roll et al. 2014). During learning sessions, the system records all the students' actions, analyzes them and tries to specify each student's metacognitive behavior. The metacognitive behavior concerns modeling the student's metacognitive help using and seeking strategies, which can provide the system with meaningful information regarding the student's learning characteristics. The metacognitive model specifies how a student uses the help facilities of the system. Based on it, the system can intervene and propose to the student to request system's assistance or not. In Table 5, basic cases of the student's metacognitive behavior are presented.

The system can make recommendations to the learner giving suggestions to request assistance or even immediately provide the student with the proper hint. For example, when the learner repeatedly fails to specify a specific element of the answer, the system can make a suggestion to him/her to request the system's assistance. In such cases, a pop up will appear stating the system's recommendation. However, if the learner neglects the recommendation, the system can intervene and provide the learner with the proper hint messages immediately after his/her answer.

The suggestions to make and the way to deliver them in terms of timing and content are specified by a rule-based expert system. All students' actions in the system are recorded and parameters such as the number of submissions on an answer, submissions time, help requests and meta-parameters concerning the analysis of the errors made are specified. Student-related parameters can provide meaningful information regarding aspects of student's help seeking behavior. The expert system consists of two main types of rules. *Profile assignment rules* specify and update aspects of a student's help seeking behavior and indicate whether a student is active or inactive while working on an exercise. *Recommendation rules* are used to make help suggestions. Below, three example rules are presented (one of profile assignment type and the others of recommendation type). For the sake of readability, they are expressed in natural language.

Table 5 Example feedback based on student's behavior

Student behavior	Feedback
Student is taking too much time to submit an answer.	Try to take a hint.
Student is constantly requesting hints before trying to submit an answer. (help abuse)	Try to submit an answer.
Student isn't requesting any help and is striving to specify the correct answer. (help refusal)	Try to take a hint.

R1: IF (student_creates_events is high)
THEN (student_is_working_on_the_exercise is yes)

R2: IF (time_between_submission is high)
and (answer_score is low)
and (hints_delivered are low)
and (student_is_working_on_the_exercise is yes)
THEN (take_hint_recommendation).

R3: IF (submitted_answers is low)
and (answer_score is low)
and (hints_delivered are many)
THEN (do_not_take_hint_recommendation)
and (submit_answer_recommendation).

The development of the rules was based on historical data on students learning in combination with the experience and the guidelines of experienced tutors. Student characteristic parameters are constantly updated during the student's learning sessions based on their interactions with the system.

Textual Feedback Generation Approach

The Feedback Generation (FG) mechanism is used to generate and provide students with various types of feedback according to their actions. The aim is to model and automate the process of feedback messages generation. The automatic generation of feedback messages is an essential part of the assistance and feedback delivery within educational systems (Gerdes et al. 2016). The mechanism takes as input the exercise and the type(s) of error(s) made and provides the appropriate feedback message(s). The generation of feedback messages is made using a template-based approach. More specifically, the error types and the system hints are associated with a suitable feedback *template*. A template consists of three main components: (i) the error category associated with the current error, (ii) the type of the error made and (iii) the message texture pattern to bear the specific textual message. In Table 6, some examples of feedback texture templates are illustrated.

The instantiation of a pattern concerns filling in the template with the proper information regarding the elements of the student's answer. Example student errors and the corresponding feedback messages during the translation process of the sentence "*Maria loves all dogs that love their master*", are presented in Fig. 3.

Table 6 Example feedback templates

Error category	Error type	Pattern
Predicate error	Number Error	[element of the student answer] is not correct because [explanation]. You should [remedial action].
Predicate error	Person Error	[element of the student answer] is not correct. It is not in third person.
Quantifier error	Type Error	Quantifier [element of the student answer] is not correct because [explanation] is associated with the word [explanation].

As an example consider the first step of the process, where a common error concerns the specification of the proper predicates. Let's suppose that a student has specified as predicates the words "love", "dogs" and the word "master-of" as a function symbol. The error detection recognizes the errors and informs the feedback generation unit about the types of errors the student has made. The sentence is analyzed and the word "love" is recognized as a verb and that it is not in the third person in the student's answer. So, it is incorrect. Also, it is recognized that the word "dogs" is a noun, it is in plural number and thus, it is incorrect. Then, an instance of the feedback framework will be created by filling in the gap in the feedback message pattern with the word "love". The feedback generation mechanism instantiates the templates:

[element of the student answer] is not correct because [explanation].

[element of the student answer] is not correct. It is not in third person.

By filling in the proper parts, the feedback messages to be given to the student, as illustrated in Fig. 3, produced:

The word dogs is not correct because *it is in plural number.*

The word love is not correct. It isn't in third person.

As a second example, errors during the third step of the process, where the student specifies the proper quantification for the variables that represents the predicates determined, are presented. A common error concerns the incorrect quantification of the variables. Let's suppose that a student in his/her answer has specified for the variable "x", which he/she has denoted to represent the entity "dog", the existential quantifier "∃", which is incorrect. The feedback generation mechanism based on this answer analysis and the diagnosis conducted, can instantiate as a second level, error-specific feedback message, the following template:

Quantifier [element of the student answer] is not correct because [explanation] is associated with the word [explanation].

By filling in the pattern with the proper information, the feedback that is given to the student is the following:

Quantifier *specified for the variable* "*x*" *which represents* "*dogs*" is not correct because *the entity dogs* is associated with the word "*all*".

The textual messages mainly concern the generation of the hints associated with the second and the third level of assistance.

Evaluation

Experimental studies were conducted to evaluate the NLtoFOL system assistance and feedback mechanism during learning. To that end, two different experiments were

designed and implemented. Participants of the experiments were students of the Artificial Intelligence class at the computer engineering and informatics department of the University of Patras. All students were in the 4th year of their study and their age ranged from 21 to 23 years. Also, they had attended the AI course lectures on logic as a knowledge representation and reasoning language and had the necessary knowledge on background topics. The main objective of the experimental studies was to obtain an assessment of the effectiveness of the feedback framework developed for the NLtoFOL system, by comparing the effects of different feedback scenarios on learning. In the experiments, we followed a pre-test/post-test approach.

First Experiment

Method

The participants in the first experiment were 226 undergraduate students (male and female). All students participated in a session of 1 h and 15 min consisted of two activities: (i) introduction and explanation of how to use the NLtoFOL system (15 min), (ii) pre-test (1 h). The experiment included four stages: pre-test, learning phase, post-test, questionnaire (see Fig. 4).

The students were randomly divided into two equal size groups, of 113 students each, namely Group A and Group B. The rates of girls and boys in Group A was almost the same as those in Group B. Initially, all participants took a pre-test aiming to assess students' domain relevant prior knowledge on converting NL sentences into logic. The pre-test consisted of ten exercises of various difficulty levels, ranging from very easy to advanced (Perikos et al. 2011, 2016). More specifically, the test consisted of one very

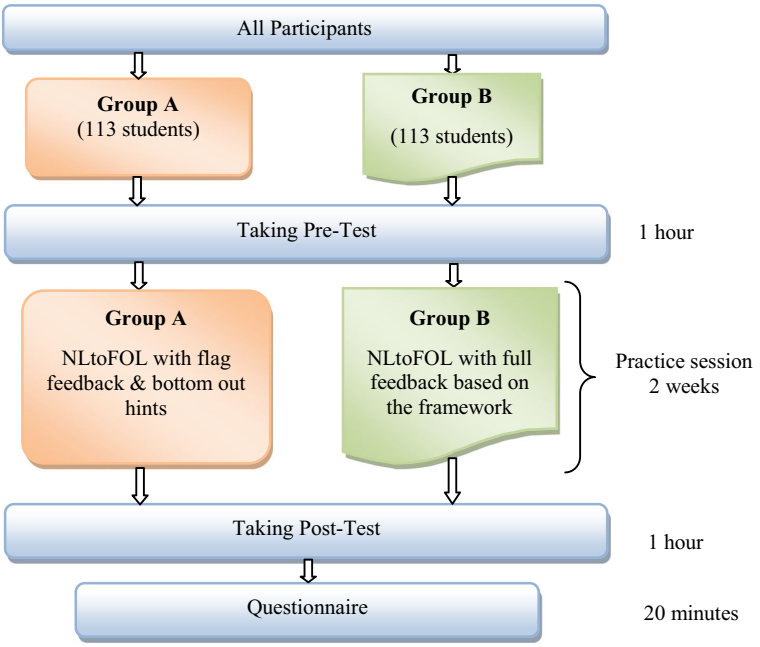

Fig. 4 Structure of the first experiment

easy exercise, two easy, four medium, two difficult and one advanced exercise. Each correct conversion was assigned one point and the maximum pre-test score was 10 points. The duration of the pre-test was 1 h and it was conducted in the computer room area of the department.

After the pre-test, access was given to all participants to register and use the NLtoFOL system for learning the formalization process. During registration, the students had to fill in their personal and demographic information such as, name, gender, age, year of study etc. The students used the system for 2 weeks, to study how to convert natural language sentences into FOL using SIP, as described in the background section. Group A used the system with a scenario where flag feedback and bottom out hints only were provided in case of an incorrect answer. Group B interacted with the system with a scenario where the system provided the full assistance and feedback framework, as presented in the previous section. The students of both groups had the ability to take hints before they give an answer. During the 2 weeks' period, both groups had access to the exact same set of exercises which consisted of 80 exercises ranging from very easy to advanced difficulty level. Also, both groups were asked to aim at daily sessions of at least 30 min and also to experiment with all the exercises that were available in the system. During the learning sessions, students' actions were recorded by the system.

After the 2 weeks learning phase, all the participant students took a post-test. The post-test consisted of 10 formalization exercises of the same difficulty levels as the pre-test and the students were given 1 h to do the formalizations and submit their answers. The exercises were presented to the students in an incremental difficulty level way, starting from exercise 1 (very easy) to exercise 10 (advanced). The exercises of post-test were in accordance with those in pre-test. More specifically, exercises of the same difficulty level were content equivalent, covering the same aspects of the formalization process and necessitating the same steps of the SIP process. Moreover, the answers of exercises of the same difficulty level were either slightly different in (logical) structure or logically equivalent. However, students couldn't be aware of or trace those similarities, due to the different NL expressions. In addition, all exercises were selected not to suffer from ambiguities and ellipses so that the textual feedback messages could be created accurately by the feedback generation mechanism. For each exercise of the post-test, the students were given the NL sentence and had to write the corresponding FOL formula without the help of the system. So, a student's submitted answer to the post-test consisted of 10 FOL formulas, one for each exercise (sentence). After a test submission, answers (FOL formulas) were automatically graded using the system's automatic assessment mechanism (Perikos et al. 2012) in a consistent way. The automatic assessment achieves high accuracy in grading students' exercises, in line with the tutor's marking behavior. The assessment results and the analysis of the errors made were available to tutors and to students almost immediately after the tests' submission.

In addition, the students of both groups were asked to fill in a questionnaire. The questionnaire was designed by the tutors of the course for evaluating the usefulness of assistance and feedback, asking for the students' experiences and opinions about the learning impact of the NLtoFOL system. The questionnaire of Group A included 12 questions and the questionnaire of Group B 18, where the six additional questions were related to students' stance towards aspects of the textual characteristics of the hints.

From the 12 common questions, 9 questions required answers based on a five point Likert scale (1- strongly disagree to 5-strongly agree) and 3 were open ended questions. Some example Likert scale questions are the following: "I enjoyed learning with the NLtoFOL system", "The feedback offered was accurate", "The feedback helped me understand my errors", "When an error was made, the assistance helped me to fix my answer", "For the items that I answered incorrectly, I examined the feedback", "The system intervened to give me suggestions that were accurate and needed". The open ended questions were provided at the end of the questionnaire to allow students to write their comments about the different types of feedback and the NLtoFOL system. An example open question: "What do you feel are the strong and the weak points of the offered feedback?". After analyzing the students' responses to the questionnaires, the reliability of the questionnaire was checked using the Cronbach's alpha (Cronbach 1951) metric. Reliabilities of the scales were good for the two groups with internal consistency coefficients $\alpha = 0.73$ for Group A and $\alpha = 0.82$ for Group B.

Results and Discussion

By the analysis of the learning phase in the first experimental study, one of our aims was to verify and assess the learning that took place between the pre-test and post-test and also look for differences in learning between different learning conditions. A one-way, between-subjects analysis of variance (ANOVA) was conducted, to evaluate the hypothesis that the two groups of undergraduate students (Group A and Group B) did not significantly differ as far as prior knowledge on natural language formalization at pre-test is concerned. The participants of Group A had a mean pre-test score of 3.12 (SD = 0.66) and those of Group B a mean score of 2.97 (SD = 0.65). Indeed, there were no significant initial differences between the group means ($F = 2.99 > 1$, $p = 0.085 > 0.05$). The test had statistics power 40.53% and its effect size (Cohen 1988) was small ($d = 0.23$). Hence, the groups were roughly of the same knowledge level before starting learning about the formalization process.

Another objective of this study was to examine the effectiveness of feedback on improving the learning achievements of the students. To determine the impact of feedback, we conducted an Analysis of Covariance (ANCOVA) to extract the differences between the two groups using the pre-test scores as the covariate and the post-test scores as dependent variables. The results indicate that after controlling for initial quantitative ability, the differences in post-test scores are statistically significantly different between the two groups: $F(1222) = 14.98$, $p < 0.001$, partial eta squared = 0.063.

Table 7 presents the ANCOVA results, in which the adjusted mean values of the post-test scores were 5.72 for Group A (flag feedback & bottom out hints), and 7.50 for Group B. Moreover, a significant difference was found between the two groups with

Table 7 ANCOVA results

Groups	N	Mean	SD	Adjusted mean
Group A (Flag Feedback & Bottom out hints)	113	5.72	0.57	5.68
Group B (Full Feedback)	113	7.50	0.90	7.53

$F(1222) = 14.98$ and $p = 0.00 < 0.05$, implying that the system had significantly positive effects on the learning achievements of Group B, where assistance and feedback based on the full developed framework was provided to the students during learning the formalization process in the NLtoFOL system. A descriptive analysis of the results was conducted. In Fig. 5, the boxplot of leaning scores of Group A and Group B before the interaction of the students with the system (pre-test) and after the interaction (post-test) are presented.

The results show an increase in the score, indicating that the students have learned in both cases. Students of both groups performed better in the post-test, achieving greater marks than in pre-test. However, the students of Group B performed much better than those of Group A. In the post-test, they made fewer mistakes and got a deeper understanding of the formalization process.

In addition, in the context of the first experimental study, the analysis of the Groups questionnaires showed that 80.5% of Group A and 84% of Group B enjoyed the experience of using the NLtoFOL system. Also, 81.5% of the students of Group A and 91% of Group B answered that they were helped in learning formalizations. Regarding the accuracy of the feedback, 93% of the students of Group A and 80.5% of Group B answered that feedback was accurate. The difference is probably due to the use of the simple type of feedback (flag and bottom out feedback) only for the students of group A, which, because of its nature,

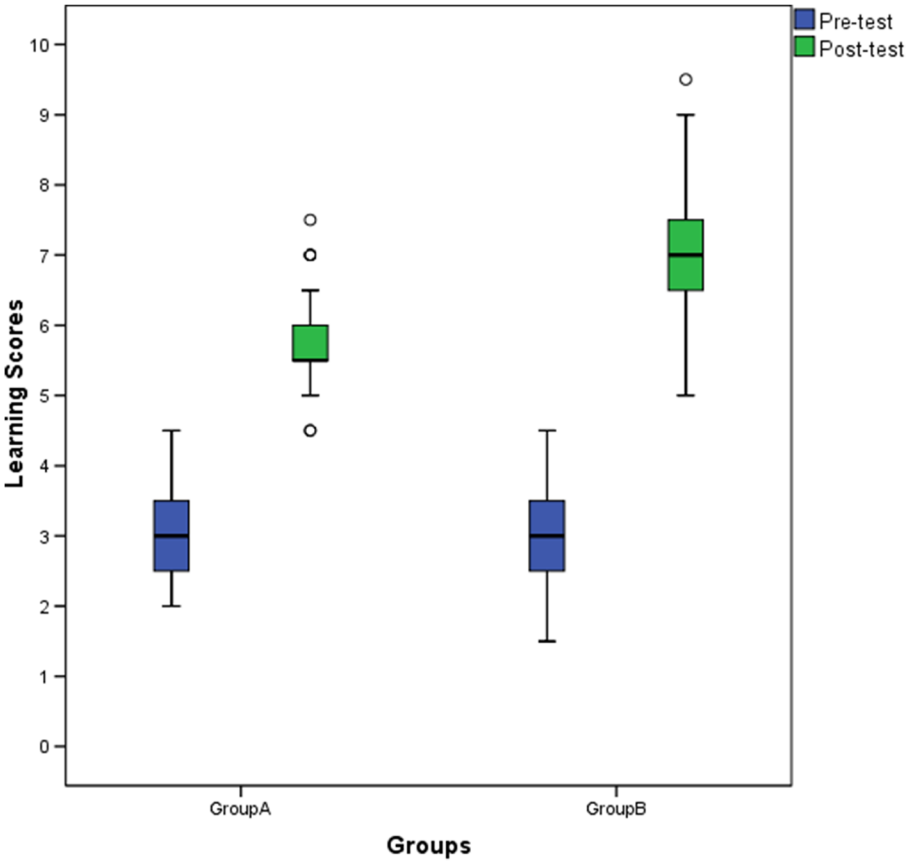

Fig. 5 Learning gains boxplots of Group A vs Group B

always produces precise hints. As far as facing errors is concerned, 35.5% of Group A and 86.5% of group B indicated that the feedback helped them in understanding their errors. Additionally, 76% of Group A and 83% of Group B declared that feedback assisted them fix their answers. Furthermore, students of Group B found textual message to be of high quality (80.5%), grammatically correct and comprehensive (76%), with appropriate length (84%). They also expressed that the system's interventions, suggesting help-related actions, were accurate and needed (79.5%). Finally, most of the students (87.5% of Group A and 95.5% of Group B) suggested integrating the system into the course curriculum and be given to next year's students.

Mining Students Learning Actions

In this part of the analysis, we examine the learning data recorded during the learning phase of the students of both groups. The analysis was designed to provide a deeper and more complete insight of the students' learning process, their performance and their knowledge construction. Also, it aimed to shed light on the efficiency of the feedback schema and examine how well the schema worked in students' learning progress.

Initially, based on the recorded data, 25 exercises of the system that all students of both groups had practiced with were selected. The exercises covered all of the different difficulty levels. Students' actions during the learning sessions of those exercises were retrieved and undergone deeper analysis. From that analysis, we were able to calculate metrics regarding the students' learning actions, indicating how students behaved during their dealing with those exercises. So, for those 25 exercises, we calculated (i) the total attempts and answers submitted by the students of both groups, (ii) the feedback messages provided by the system, (iii) the bottom out hints provided by the system, (iv) the performance of the students after the delivery of different hints. The calculated metrics are presented in Table 8. Regarding the submitted answers from students of Group A, an average of 7.2 answers submitted per exercise's step. This indicates that the students submitted 6.2 answers, on average, after a first incorrect answer. In the learning scenario of Group A (and of Group B), the system after the first erroneous answer, immediately provided flag feedback indicating the correct and the incorrect elements of the submitted answer. However, students of Group A submitted an average of 5.2 erroneous answers, until they were able to fix their answer and submit the correct or a correct one. On the other hand, the students of Group B submitted, on average, 3.9 answers per step of sentence conversion process. Subsequently, 2.9 answers were submitted after a first incorrect one. Furthermore, it is calculated that the students needed to make another 1.9 submission attempts before fixing their answer and submitting a correct answer. A metric of great importance concerns the calculation of the system's

Table 8 Student behavior in the learning phase

	Group A	Group B
Average attempts (per step)	7.2	3.9
Hints provided (except flag feedback)	51%	76%
Bottom out hints (% per step completed)	51%	12%

assistance provided to the students of both groups. As system's assistance, we consider the feedback messages given to the student during the steps of the process and after the student submits an answer. So, in the learning scenario of Group A, this assistance concerns the delivery of bottom-out hints, which provided to students with an element or the entire correct answer. In the learning scenario of Group B, the system's assistance concerns the delivery of feedback based on the full feedback framework. The results indicate that students of Group A completed 51% of the formalization steps with the assistance of the system, offered in the form of bottom out hints. On the other hand, the students of Group B completed 76% of the steps of the exercises based on the system's second level and third level assistance. The delivery of bottom out hints, provided to both groups, has been also analyzed. The students of Group B completed 12% of the process steps based on the assistance of bottom out hints, in contrast to Group A, which completed 51% of the formalizations steps based on bottom out hints.

Further results from the analysis indicate that the system's second and third level assistance helped the students of Group B to complete 64% of the formalization steps without bottom out hints, out of the 76% of steps, for which students requested assistance. In the second and third level of assistance, the system provides students with positive, error-specific and procedure feedback and the results denote that these types of feedback helped students to fix their errors and submit a correct answer. Moreover, it can be also seen that the students of Group A were in need of true help, since they submitted many incorrect answers, much more answers than the students of Group B, and also completed half (51%) of the formalizations steps with the delivery of a part of (or the entire) correct answer. What is more, students of Group A after the delivery of flag feedback, in 51% of the exercises' steps, were not able to specify the correct answer and requested for bottom out help. That indicates that the previous offered assistance (flag feedback) wasn't so effective in helping them to correct their answers. In contrast, Group B students were able to complete 64% of the steps with hints of the second and third level of assistance and completed only 12% of the steps with the assistance of bottom out hints.

Another interesting aspect to examine concerns the analysis of the students' performance, after the system's assistance, in relation to different assistance levels. In cases that students had submitted an answer that was not correct, hints of different assistance levels were delivered based on the assistance schema. We examined the degree that those hints assisted students to fix errors and specify the correct answer. For this purpose, students' answers that were submitted just after students had got some kind of hints from the system were analyzed. In Table 9, the percentages of correct answers submitted after the delivery of hints for each assistance level are presented.

The results indicate that the delivery of flag feedback assists students to immediately specify the correct answer in 17% of the cases for Group A and in 20% of the cases for

Table 9 Correct answers percentage per assistance level given after hints provision

Correct answer rates	Group A	Group B
Flag feedback	17%	20%
Second level	–	64%
Third level	–	34%
Fourth level	81%	92%

Group B. The second level assistance of the system and the delivery of positive feedback and error specific hints assisted students of Group B to immediately fix the errors made and submit a correct answer in 64% of the cases. The third level assistance, when requested and delivered to the students, assisted them to correct their answer in almost one third of the cases (34%). Finally, bottom out hints assisted students to fix their answers and submit a correct one in 81% of the cases of Group A and in 92% of the cases of Group B. For the analysis of the fourth assistance level, cases where bottom out hint offered part (and not the entire) of the correct answer to the students were collected. Those cases mainly concerned inaccurate answers, where bottom out hints corrected one element that was incorrectly specified, or incomplete answers, where hints provided a missing element. The difference of the effectiveness of the bottom-out hints in assisting students of both groups to submit a correct answer is mainly due to the fact that students of Group B, when were given bottom out hints, they had, in general, less errors and thus it was easier for them to correct their answers.

In addition, the analysis revealed some interesting situations that seem to confuse students. For example, an interesting situation was reported when students submitted a superfluous–accurate answer. The results of the analysis revealed the submission of many incorrect answers, after the delivery of flag feedback in a superfluous–accurate answer. This situation was not expected, since in such a case, a student had submitted an accurate answer and had defined all the required elements correctly. Analyzing the students' answers that were submitted after a superfluous-incorrect answer, we found that in most cases (approximately 81%) the students tried to fix the extra element of the answer, which was colored red. It seems that flag feedback created the misconception to the students that the red element should be fixed rather than removed. The operational behavior of the flag feedback tends to direct the students to work on fixing the red colored element rather than to remove it. Consequently, it seems that the delivery of flag feedback in case of superfluous - accurate answers is not a good choice.

Furthermore, a third analysis of the first experiment's data was conducted, regarding a deeper analysis of the students' post-test data. More specifically, we analyzed the performance of the students of the two groups in the exercises of the post-test and concentrated on the errors made. The number of the errors made is particularly important as the outcome of the diagnosis had an impact on the adopted pedagogical strategy. The post-test consisted of 10 exercises and their difficulty levels were as follows: exercise 1 was very easy, 2–3 were easy, 4–7 were medium, 8–9 were difficult and exercise 10 was of advanced level. For each exercise, the answers provided by the students (FOL formulas) of both groups were retrieved and the number of errors made at each exercise was calculated. In Fig. 6, the average number of errors made by students of both groups is presented.

The results show that students of Group A made in the post-test more errors than students of Group B. The average number of the errors made per question by Group A is approximated between 5 and 9, while average errors made by Group B range between 2 and 7 errors per question. In addition to the score achieved, errors analysis indicates that the full feedback framework helped students of Group B in getting and retaining a better understanding of the formalization process.

Moreover, an additional study was conducted with the aim to analyze the system's metacognitive level of feedback and its effect on students learning. For this purpose, students' learning actions after the provision of metacognitive hints, were analyzed. A first aim was to examine the degree that students accepted and followed the system's

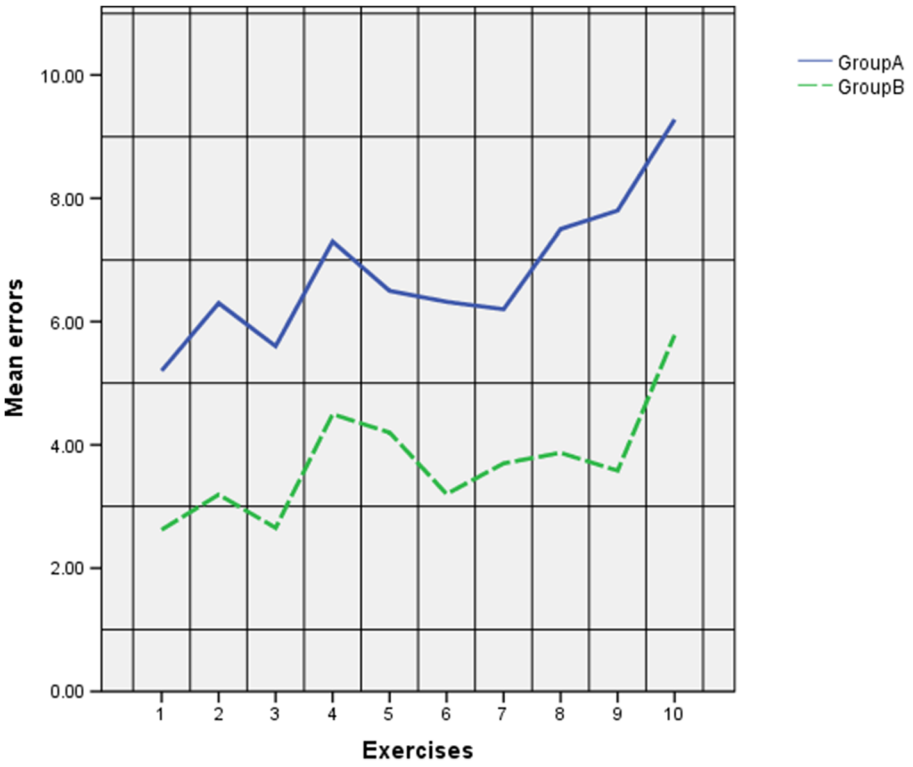

Fig. 6 Average errors made by the students

metacognitive assistance and the corresponding suggestion made. In Table 10 the acceptance rates of the metacognitive recommendations are presented.

The results reveal that the majority of students follow the system's recommendations. Students of GroupA followed system's recommendation to request for assistance in 81% of the cases offered and students of GroupB in 89% of the cases. In addition, the students' performance in those cases was deeper analyzed. More specifically, after the students' requests for assistance and the delivery of the corresponding hints messages, the degree that those hints assisted students to fix their answers was examined. The students' answers that were submitted just after they had got the hints were analyzed and the percentages of correct answers are illustrated in Table 11.

The analysis showed that the effectiveness of hints is quite good and greater compared to the results of Table 9. The difference of the effectiveness of hint messages

Table 10 Students acceptance of metacognitive recommendation

Hint type	Acceptance	
	Group A	GroupB
Request assistance recommendation	81%	89%
Submit answer recommendation	–	72%

Table 11 Hints effectiveness after recommendations for request assistance

Correct answer rates	Group A	Group B
Flag feedback	–	–
Second level	–	67%
Third level	–	39%
Fourth level	84%	94%

can derive from the fact that the recommendations for requesting assistance were offered to students when they have worked on the current exercise and have examined possible solutions with only system's flag feedback.

Second Experiment

Method

A second experiment was conducted with the aim to examine the degree that the feedback provided by the system assisted students' learning compared to the feedback specified by the tutor. In this experiment, 120 students were selected and randomly divided into two groups. The first group, namely Group FF, consisted of 60 students and used the NLtoFOL system with a scenario, where full assistance and feedback were provided by the system. The second group, namely Group TF, consisted of 60 students and used the NLtoFOL system with a scenario, where tutor feedback was provided to the students. More specifically, students of this group were asked to study specific exercises for which the feedback messages where inserted by the tutor of the course. The tutor had encoded the feedback messages as text templates in a similar way to that used by cognitive tutors (Aleven et al. 2009). Tutor had examined the exercises that were available to students and had associated specific exercises, error types and constraint violation cases with proper feedback templates to generate the hint messages or even with exact hints messages to be offered to students. The concept was the tutor to specify appropriate feedback template messages and hints to exercises, solution stages and error types freely according to the assistance plan that he/she follows in real class conditions. For example, some cases of feedback template messages were "Predicate third person violation. Predicate [word] is not correct.", "The number of the arguments specified for the predicate [predicate] is incorrect.". Feedback messages cover generic and specific cases and were offered to students often enough according to the tutor's assistance plan. Hint messages were provided to learners in a similar manner like in the framework and more specifically, flag feedback was provided instantly on students' answers and tutor's hints were offered on learner's demand for assistance. The structure of the second experiment is illustrated in Fig. 7.

Initially, both groups took a pre-test, which consisted of 10 exercises of various difficulty levels. Each correct conversion was assigned one point and the maximum pre-test score was 10 points. The duration of the pre-test was 1 h and it was conducted in the computer room area of the department. After that, both groups were given access to use the NLtoFOL system to study formalization process for 1 week. The system during the learning sessions provided the students of Group FF with feedback according

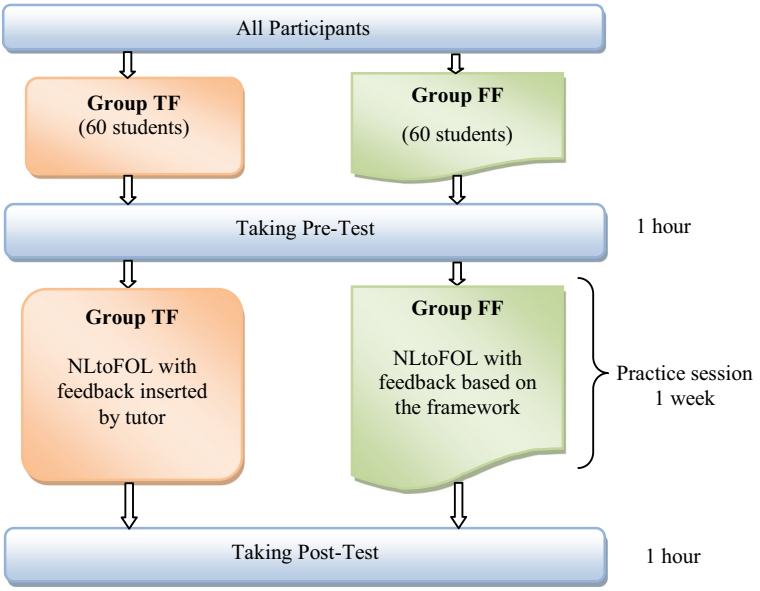

Fig. 7 Structure of the second experiment

to the full feedback developed schema and to the students of Group TF feedback based on the messages inserted by the tutor. After the two groups studied for 1 week, the students took a post test. The post-test consisted of ten formalization exercises of various difficulty levels and the students were given 1 h to do the formalizations and submit their answers. The post-test was conducted in the department's computer area.

Results and Discussion

A preliminary analysis of the pre-test scores was performed using one-way analysis of variance (ANOVA) to investigate the hypothesis that the two groups of undergraduate students (TF and FF) did not significantly differ on prior knowledge on formalizations at the pre-test. The participants of group TF had a mean pre-test score of 3.39 (SD = 0.83) and those of group FF a mean pre-test score of 3.20 (SD = 0.87). The results show that there were no significant initial differences between the pretest group means ($F = 1.52 > 1$, $p = 0.22 > 0.05$). The test had statistics power 33.7% and its effect size was small ($d = 0.23$).

An Analysis of Covariance (ANCOVA) was conducted to extract the difference between the groups using the pre-test scores as the covariate and the post-test scores as dependent variables. Table 12 summarizes the ANCOVA results, in which the adjusted

Table 12 ANCOVA results

Groups	N	Mean	SD	Adjusted mean
Group TF (Tutor feedback)	60	7.03	1.10	6.95
Group FF (Full Feedback)	60	6.90	0.10	6.96

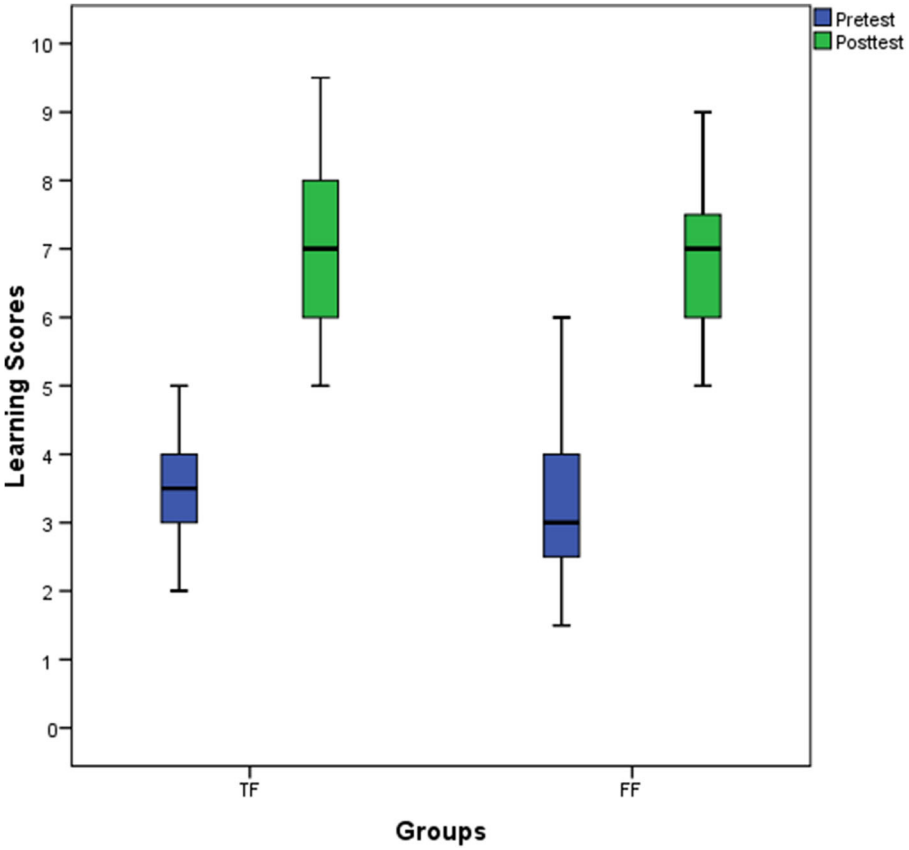

Fig. 8 Learning gains of the two groups

mean values of the post-test scores are 7.03 for Group TF and 6.90 for the FF. Moreover, no significant differences between the two student groups (TF and FF) was found ($F(1116) = 1.39$ and $p = 0.24 > 0.05$, $\eta_p^2 = 0.012$), the effect size was 0.11 and the power of the test 0.22, implying that the groups learned and performed in the same degree. In Fig. 8, the learning gains of the two groups are presented.

Results show an increase in the score indicating that the students have learned in both cases. Also, both groups returned a p-value, which was less than 0.001 and it can be concluded that the student's score increased significantly by using the NLtoFOL system in both learning scenarios. Also, it can be seen that both tutor inserted hints and the textual hints provided by the system based on the feedback framework assisted the students to the same degree.

Conclusions and Future Work

In this paper, we introduce a general framework for modeling system assistance to students in an intelligent tutoring system. This framework includes, first, a systematic and generic modeling of students' answers. Also, it includes an assistance generation scheme that models feedback types into levels of assistance and delivers feedback in an

incremental, level-based way. Furthermore, a general template-based approach for the generation of natural language hints is presented. The above framework is validated through its instantiation in a specific intelligent tutoring system, called the NLtoFOL system, which deals with teaching/learning a specific AI topic, namely the conversion of natural language sentences into logic formulas. For the utilization of the feedback framework in the tutoring system, the domain's error categorization scheme was formulated, which is used for recognizing the types of errors made by students and for guiding the feedback providing mechanism.

Moreover, we study the complex nature of feedback and we investigate the learning effects of different feedback types on the NLtoFOL tutoring system. Extended experimental studies were conducted to evaluate the effectiveness of the framework in the tutoring system. The analyses indicate that the developed feedback framework effectively guides students during their learning sessions and assists them at a similar level as human tutors do.

The framework is quite generic and for its utilization in educational activities in various domains and tutoring systems, an error categorization of the domain and the formulation of the textual feedback templates of the errors would be required to be designed. An aim of the framework is to provide a systematic way to model students' answers, model the feedback types into levels of assistance and also provide a blueprint of how textual feedback messages could be generated automatically. The feedback hierarchy is generic and can be adopted as presented. The modeling of students' answers introduced is quite suitable for educational activities, where students' answers can be decomposed into parts. In contrast, it cannot be directly applied as it is to open answer questions.

As future work, we plan to make a larger scale analysis of the students' educational data and study the feedback effectiveness on additional aspects, like for example in relation to the difficulty of the exercises and the students' level and performance. Furthermore, we aim to examine the possible utilization of the feedback framework in educational systems mainly in the context of symbolic logic and mathematics, where it seems to be suitable due to the characteristics of the skills taught, the nature of the exercises and the students' answers to them. Exploring this direction is a key aspect of our future work.

Acknowledgments The authors would like to thank the anonymous reviewers for their valuable and insightful comments which greatly assisted in improving this paper.

Appendix A - NLtoFOL SIP Conversion Process

The NLtoFOL SIP conversion process was introduced (Hatzilygeroudis 2007) to assist students in learning and implementing the translation of NL sentences into logic. The main purpose of SIP is to help the students face the cognitive complexity of the translation by modelling it as a step-based process and offering to the students a structured way to implement it. SIP process consists of ten steps. At each step the student has to do a specific task for converting to and constructing the FOL formula. The steps are the following:

1. Spot the verbs, the nouns and the adjectives in the sentence and specify 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. Divide produced formulas in groups of the same level formulas.
8. If only one group of formulas is produced, create the next level formula and go to step 10.
9. Specify the connectives between formulas of each group, create the next level formulas and go to step 7.
10. Place quantifiers at the right places in the produced formula to create the final FOL formula.

The steps of the process that are needed to be implemented in order to convert a NL sentence into FOL depend on the sentence characteristics and semantics and so different sentences may require the implementation of different steps of the process. Below, the NLtoFOL SIP process is illustrated through an example conversion, that of the sentence:

Every city has a dog-catcher that has been bitten by every dog living in the city

Step 1: Spot the verbs, the nouns and the adjectives in the sentence and specify corresponding predicates or function symbols.
There are five such items here:

city → predicate: *city*
has a dog-catcher → predicate: *dog-catcher*
has been bitten → predicate: *has-bitten*
dog → predicate: *dog*
living in → predicate: *lives-in*

Predicates must be expressed in singular number (in cases of nouns or adjectives) and in third person (in cases of verbs). Also, auxiliary verbs (like "has") are not represented by a predicate in the FOL formula. They are usually absorbed by related nouns, adjectives or regular verbs. For example, the "has a dog-catcher" case above.

Step 2: Specify the number, the types and the symbols of the arguments of the function symbols (first) and the predicates (next).
We do that in the following table: (Table 13).

Table 13 Sentence's Step 2 analysis

Predicate/Function	Arity	Argument types	Symbols
city	1	variable	x
dog-catcher	2	variable, variable	y, x
dog	1	variable	z
lives-in	2	variable, variable	z, x
has-bitten	2	variable, variable	y, z

The arity of a predicate is the number of entities that the relation represented by the predicate involves. For example, "city" has arity "1", because it is a relation of an entity with itself, ie represents an attribute of an entity. Similarly, "lives-in" has arity "2", because it relates two entities, e.g. a dog and the place where he/she lives. The number of arguments of a predicate is equal to its arity. An argument of a predicate can be either a constant (if it refers to an individual entity, e.g. John) or a variable (if it refers to a set of entities) or a function (if it refers to entities defined via introduced functions). What is the type of an argument depends on the NL sentence. For example, in the given sentence there is no reference to any individual/specific entity (like a dog name, a city name etc.) and also there are no functions defined, so all arguments of all predicates are variables. The symbols used for variables usually belong to a pre-specified set of symbols (e.g. x, y, z, w, s, r, t) possibly with indexes. Another issue here is which arguments of which predicates will be the same variable(s). This again is based on the semantics of the NL sentence. For example, "lives-in" and "dog" have a common variable ("z"), because according to the sentence it refers to dogs living in the city.

Step 3: Specify the quantifiers of the variables.

$x \rightarrow \forall$ (due to "every"), $y \rightarrow \exists$ (due to "has a"), $z \rightarrow \forall$ (due to "every")

A variable is assigned a universal quantifier ("∀") if, according to the sentence, represents a whole category of entities (e.g. variable "x" that represents all cities). It is assigned an existential quantifier if it represents some unknown member(s) of a category of entities (e.g. "y" that represents 'some' dog-catcher of a city, not all).

Step 4: Construct the atomic expressions (or atoms) corresponding to predicates.

We construct as many atoms as the predicates:

Atom 1: *city(x),* Atom 2: *dog-catcher(y, x),* Atom 3: *dog(z),*
Atom 4: *lives-in(z, x),* Atom 5: *has-bitten(z, y)*

This is straight forward, if we know the predicates and their arguments, and is based on the syntax of an atomic formula in FOL.

Step 5: Divide produced atoms in groups of the same level atoms.

This mainly refers to grouping atoms that should be connected with each other with some connective:

AtomGroup 1: *{city(x)},* AtomGroup 2: *{dog-catcher(y, x)},*
AtomGroup 3: *{dog(z), lives-in(z, x)},* AtomGroup 4: *{has-bitten(z, y)}*

The criterion for finding atoms that belong to the same group is to have their predicates in a compact, semi-autonomous part of the NL sentence, like a subordinate clause or a sub-clause that is the subject or the object of another predicate in the sentence. For example, in our case, "dog" and "lives-in" belong to the sub-clause "every dog living in the city", which is the subject of the predicate "has bitten" or the object of the predicate "has been bitten"). So, they will be in the same group.

Step 6: Specify the connectives between atoms of each group and create corresponding logical formulas.

We form the formulas corresponding to the groups of step 5.

AtomGroup 1 → Form 1: *city(x),* AtomGroup 2 → Form 2: *dog-catcher(y, x)*, AtomGroup 3 → Form 3: *dog(z) ∧ lives-in(z, x)*, AtomGroup 4 → Form 4: *has-bitten(z, y)*

Which connective will be used to join two atoms totally depends on the semantics of the corresponding part of the NL sentence.

Step 7: Divide produced formulas in groups of the same level formulas.

This usually corresponds to specifying the left and right parts of an implication:

FormGroup1-1: *{city(x)}*, FormGroup1-2: *{dog-catcher(y, x)}*, FormGroup1-3: *{dog(z) ∧ lives-in(z, x), has-bitten(z, y)}*

The guidelines here are similar to those of Step 5, but for groups of atoms instead of single atoms, ie at an upper level.

Step 9: Specify the connectives between formulas of each group, create the next level formulas and go to step 7.

FormGroup1-1 → Form1-1: *city(x)*, FormGroup1-2 → Form1-2: *dog-catcher(y, x)*,
FormGroup1-3 → Form1-3: *(dog(z) ∧ lives-in(z, x)) ⇒ has-bitten(z, y)*

Also, it solely depends on the NL sentence semantics.

Step 7: Divide produced formulas in groups of the same level formulas.

FormGroup 2–1: *{city(x)}*
FormGroup 2–2: *{dog-catcher(y, x), (dog(z) ∧ lives-in(z, x)) ⇒ has-bitten(z, y)}*

Step 9: Specify the connectives between formulas of each group, create the next level formulas and go to step 7

FormGroup 2–1 → Form2-1: *city(x)*
FormGroup 2–2 → Form2-2: *dog-catcher(y, x)∧ (dog(z)∧ lives-in(z, x)) ⇒ has-bitten(z, y)}*

Step 7: Divide produced formulas in groups of the same level formulas.

FormGroup 3–1: *{city(x), dog-catcher(y, x) ∧ (dog(z) ∧ lives-in(z, x)) ⇒ has-bitten(z, y)}*

Step 8: If only one group of formulas is produced, specify the connectives between formulas of the group, create the next level formula and go to step 10.

FormGroup 3–1→ Form 2–3: *city(x) ⇒ (dog-catcher(y, x) ∧ (dog(z) ∧ lives-in(z, x)) ⇒ has-bitten(z, y))*

Step 10: Place quantifiers in the right places in the produced formula to create the final FOL formula.

The produced final FOL sentence is as follows:

"*(∀x) city(x) ⇒ ((∃y) dog-catcher(y, x) ∧ (∀z) (dog(z) ∧ lives-in(z, x)) ⇒ has-bitten(z, y))*"

Appendix B - Natural Language Analysis for Generating Error Explanations

A major purpose of the system's assistance is to help students understand their errors and misconceptions. So, the feedback mechanism automatically generates proper justifications and explanations in order to cultivate an effective learning process. The error explanations can help the learner get a clear understanding of why his/her answer is erroneous and can have a crucial impact on knowledge revision and construction (Bitchener 2008). When a student submits an erroneous answer, the system performs a deep analysis of the errors in order to trace their origin regarding the structure and the semantics of the exercise (natural language sentence). Specifically, the analysis of the errors concerns the identification of the exercise's elements and the semantics that are translated incorrectly as well as the constraints that may be violated during the translation. So, alongside the student's error analysis, the NL sentence is also analyzed and the errors made by the student are semantically associated with the proper elements of the sentence.

The system analyzes the structure of each exercise-sentence with the use of natural language processing tools, which are integrated into the system. The natural language processing tools that the system utilizes are the Stanford parser (De Marneffe et al. 2006) and the Stanford NER (Finkel et al. 2005) tools. Initially, the Stanford parser, which is a very popular morphosyntactic analysis tool, is used to determine the grammatical structure of a sentence and specify for each word its base form (lemma) and its grammatical role in the sentence. Also, it specifies the relationships between the sentence's words and determines the corresponding dependencies, which provide remarkable assistance in sentence analysis. The dependency tree represents the grammatical relations between the sentence's words in a tree based approach. Those relationships are presented as triplets consisting of the name of the relation, the governor and the dependent respectively. Dependencies indicate the way that words are connected and interact with each other. When the sentence morphosyntactic analysis is completed and the dependency tree is created, special parts of the dependency tree and specific words are further analyzed. The dependency tree is analyzed and the relationships and types of interactions/connections between the sentence words are examined.

First, the words that specify entities are specified. These words can represent general classes of entities such as humans, animals etc. and their determination is made based on the words' grammatical role in the sentence. Words that represent entities constitute

the predicates of the sentence's corresponding FOL formula. After entity words are recognized, the system performs a deeper analysis regarding their role in the sentence and the type of their relationships/connections with other words. More specifically, it tries to analyze special types of relationships that may appear with quantification words. These relationships are recognized as modification - 'mod' dependencies in the dependency tree developed. So, these dependencies, connecting entity words with quantification words, are analyzed and a quantification word's impact defines the way that the entity has to be quantified in the corresponding FOL formula. Following, the system specifies the existence of named entities in the sentence. The determination of named entities is made with the utilization of Stanford Named Entity Recognizer (NER) tool. The tool labels sequences of words in a sentence which are the names of things, such as person names and company names with the proper category label. Examples of named entities can be a person (e.g. Maria), locations such as a country (e.g. Greece), a city (e.g. Athens) etc. Named entities constitute the constants of the sentence's corresponding FOL formula and the recognition of the named entities in a sentence is of extreme importance. For example, the dependency tree of the exercise-sentence "Maria loves all dogs that love their master" as specified by the parser tool is depicted in Fig. 9.

In addition, the named entity recognizer tool has specified that the word "Maria" represents a person. The nodes of the dependency tree are the sentence's words and the edges specify the existing relationships between the words. For each word, its grammatical role (depicted just under the word) in the sentence and the way it interacts with other words are specified. The interaction between two words is denoted by the existence of an edge and the exact type of interaction is denoted by the edge's name. Basic information concerns the determination of the predicates/functions, which are the verbs, the nouns and the adjectives of the sentence. The grammatical role of each word designates that predicates and functions are formulated by the words "loves", "dogs", "love" and "master", since these are the nouns and the verbs of the sentence. The recognition of the word "Maria" as a person, designates that it will be represented by a constant in the FOL formula. The relationship "det" between the words "dogs" and "all" designates that the quantifier of the

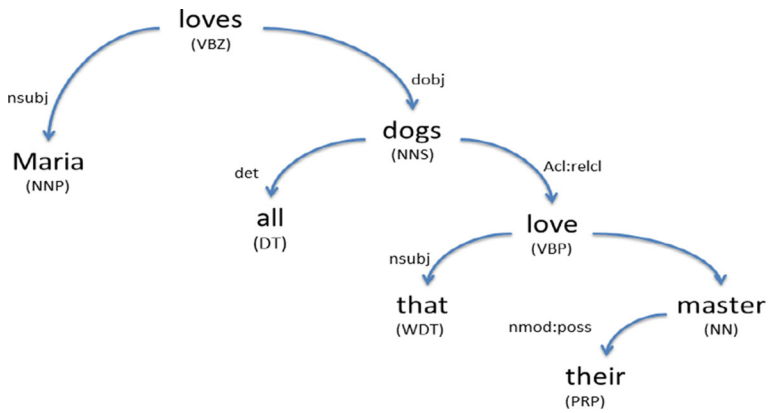

Fig. 9 Dependencies of the sentence "Maria loves all dogs that love their master"

variable representing "dogs" will be the universal one. The relationships of the word "loves" (nsubj with the word "Maria" and dobj with the word "dogs") designates that the predicate "loves" has two arguments, which are the constant "Maria" and the variable representing "dogs".

The analysis of the sentence-exercise provides substantive information regarding the sentence characteristics and can assist the feedback generation mechanism to provide meaningful assistance to the learner in different ways. The elements of the student's answer are associated with the proper words of the NL sentence and the information of each word provides valuable data in order to explain the reason.

The mechanism, in case of errors, can automatically identify the semantics of the error and provide proper explanations denoting the reason why an element of the answer is incorrect. Also, can provide semantic justifications why an element of the answer is correct and make suggestions on how to proceed with the formalization process denoting what to do to fix an incorrect element of the answer, what to do next and why. The hints provided to the student are generated as simple, short NL sentences by the feedback generation mechanism.

References

Aleven, V., & Koedinger, K. R. (2000). Limitations of student control: Do students know when they need help?. *In Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 292–303). Springer Berlin Heidelberg.

Aleven, V., Stahl, E., Schworm, S., Fischer, F., & Wallace, R. (2003). Help seeking and help design in interactive learning environments. *Review of Educational Research, 73*(3), 277–320.

Aleven, V., Mclaren, B., Roll, I., & Koedinger, K. (2006a). Toward meta-cognitive tutoring: a model of help seeking with a cognitive tutor. *International Journal of Artificial Intelligence in Education, 16*(2), 101–128.

Aleven, V., McLaren, B. M., Sewall, J., Koedinger, K. R. (2006). The cognitive tutor authoring tools (CTAT): preliminary evaluation of efficiency gains. In *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 61–70). Springer Berlin Heidelberg.

Aleven, V., Mclaren, B. M., Sewall, J., & Koedinger, K. R. (2009). A new paradigm for intelligent tutoring systems: example-tracing tutors. *International Journal of Artificial Intelligence in Education, 19*(2), 105–154.

Aleven, V., Roll, I., McLaren, B. M., & Koedinger, K. R. (2016). Help helps, but only so much: research on help seeking with intelligent tutoring systems. *International Journal of Artificial Intelligence in Education, 26*(1), 205–223.

Alonso, J. A., Aranda, G. A., Martn–Mateos, F. J. (2007). KRRT: Knowledge representation and reasoning tutor system. In *Computer Aided Systems Theory–EUROCAST 2007* (pp. 400–407). Springer Berlin Heidelberg.

Anderson, J. R., Corbett, A. T., Koedinger, K. R., & Pelletier, R. (1995). Cognitive tutors: lessons learned. *The Journal of the Learning Sciences, 4*(2), 167–207.

Barker-Plummer, D., Cox, R., Dale, R., Etchemendy, J. (2008). An empirical study of errors in translating natural language into logic. In *Proceedings of the 30th Annual Meeting of the Cognitive Science Society/ CogSci* (pp. 505–510).

Barker-Plummer, D., Cox, R., Dale, R. (2009). Dimensions of difficulty in translating natural language into first order logic. *International Working Group on Educational Data Mining*.

Barker-Plummer, D., Cox, R., Dale, R. (2011). Student translations of natural language into logic: The Grade Grinder corpus release 1.0. In *Proceedings of the 4th international conference on educational data mining* (pp. 51–60).

Barker-Plummer, D., Dale, R., Cox, R. (2012). Using edit distance to analyse errors in a natural language to logic translation corpus. *In Proceedings of the International Conference on Educational Data Mining,* 134.

Berlanga, A. J., Van Rosmalen, P., Boshuizen, H. P., & Sloep, P. B. (2012). Exploring formative feedback on textual assignments with the help of automatically created visual representations. *Journal of Computer Assisted Learning, 28*(2), 146–160.

Bitchener, J. (2008). Evidence in support of written corrective feedback. *Journal of Second Language Writing, 17*(2), 102–118.

Black, P., & Wiliam, D. (1998). Assessment and classroom learning. *Assessment in Education, 5*(1), 7–74.

Brachman, R. J., Levesque, H. J. (2004) Knowledge representation and reasoning. Morgan Kaufmann Publishers, Massachusetts, US, 9, 9.

Cohen, J. (1988). *Statistical power analysis for the behavioral sciences* (2nd ed.). New Jersey: Hillsdale.

Corbett, A. T., & Anderson, J. R. (2001). Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 245–252). ACM.

Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika, 16*(3), 297–334.

De Marneffe, M. C., MacCartney, B., & Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. *Proceedings of LREC, 6*, 449–454.

Dempsey, J. V., & Wager, S. U. (1988). A taxonomy for the timing of feedback in computer-based instruction. *Educational Technology, 28*(10), 20–25.

Dempsey, J. V., Driscoll, M. P., Swindell, L. K. (1993). Text-based feedback. Interactive instruction and feedback, 21–54.

Fiedler, A., & Tsovaltzi, D. (2003). Automating hinting in an intelligent tutorial dialog system for mathematics. *Knowledge Representation and Automated Reasoning for E-Learning Systems*, 23.

Finkel, J. R., Grenager, T., Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics* (pp. 363–370). Association for Computational Linguistics.

Genesereth, M. R., & Nilsson, N. J. (1987). Logical foundations of artificial Intelligence. *Morgan Kaufmann*.

Gerdes, A., Heeren, B., Jeuring, J., van Binsbergen, L. T. (2016). Ask-Elle: an adaptable programming tutor for Haskell giving automated feedback. *International Journal of Artificial Intelligence in Education*, 1–36.

Gheorghiu, R., & VanLehn, K. (2008). XTutor: an intelligent tutor system for science and math based on excel. In *Proceedings of the International Conference on Intelligent Tutoring Systems* (pp. 749–751). Springer Berlin Heidelberg.

Goldin, I. M., Koedinger, K. R., Aleven, V. (2013). Hints: You can't have just one. In S. K. D'Mello, R. A. Calvo, A. Olney (Eds.), In *Proceedings of the 6th International Conference on Educational Data Mining (EDM 2013)* (pp. 232–235).

Gouli, E., Gogoulou, A., Papanikolaou, K. A., Grigoriadou, M. (2006). An adaptive feedback framework to support reflection, guiding and tutoring. *Advances in web-based education: Personalized learning environments*, 178–202.

Hattie, J., & Timperley, H. (2007). The power of feedback. *Review of Educational Research, 77*(1), 81–112.

Hatzilygeroudis, I. (2007). Teaching NL to FOL and FOL to CF conversions. In *FLAIRS Conference* (pp. 309–314).

Hatzilygeroudis, I., & Perikos, I. (2009). A web-based interactive system for learning NL to FOL conversion. In *New Directions in Intelligent Interactive Multimedia Systems and Services-2* (pp. 297–307). Springer Berlin Heidelberg.

Havnes, A., Smith, K., Dysthe, O., & Ludvigsen, K. (2012). Formative assessment and feedback: making learning visible. *Studies in Educational Evaluation, 38*(1), 21–27.

Hirashima, T., Horiguchi, T., Kashihara, A., & Toyoda, J. (1998). Error-based simulation for error-visualization and its management. *International Journal of Artificial Intelligence in Education, 9*(1–2), 17–31.

Koedinger, K. R., & Aleven, V. (2007). Exploring the assistance dilemma in experiments with cognitive tutors. *Educational Psychology Review, 19*(3), 239–264.

Kulhavy, R. W., & Stock, W. A. (1989). Feedback in written instruction: the place of response certitude. *Educational Psychology Review, 1*(4), 279–308.

Lopez, L. (2009). *Effects of delayed and immediate feedback in the computer-based testing environment*. ProQuest.

Maestro-Prieto, J. A., Simon-Hurtado, A. (2015). Learner-Adaptive Pedagogical Model in SIAL, an Open-Ended Intelligent Tutoring System for First Order Logic. In *Proceedings of the International Conference on Artificial Intelligence in Education* (pp. 702–705). Springer International Publishing.

Mandernach, B. J. (2005). Relative effectiveness of computer-based and human feedback for enhancing student learning. *The Journal of Educators Online*, *2*(1).

Mathan, S., & Koedinger, K. R. (2003). Recasting the feedback debate: Benefits of tutoring error detection and correction skills. In *Proceedings of the International Conference on Artificial Intelligence in Education* (pp. 13–20).

Mathan, S. A., & Koedinger, K. R. (2005). Fostering the intelligent novice: learning from errors with metacognitive tutoring. *Educational Psychologist, 40*(4), 257–265.

McKendree, J. (1990). Effective feedback content for tutoring complex skills. *Human-Computer Interaction, 5*(4), 381–413.

Mitrovic, A., Martin, B., & Suraweera, P. (2007). Intelligent tutors for all: the constraint-based approach. *IEEE Intelligent Systems, 4*, 38–45.

Mitrovic, A., Martin, B., Suraweera, P., Zakharov, K., Milik, N., Holland, J., & McGuigan, N. (2009). ASPIRE: an authoring system and deployment environment for constraint-based tutors. *International Journal of Artificial Intelligence in Education, 19*(2), 155–188.

Mitrovic, A., Ohlsson, S., & Barrow, D. K. (2013). The effect of positive feedback in a constraint-based intelligent tutoring system. *Computers & Education, 60*(1), 264–272.

Moreno, A., & Budesca, N. (2000). Mathematical logic tutor-propositional calculus. In *Proceedings of the First International Congress on Tools for Teaching Logic* (pp. 99–106).

Moreno, R., & Mayer, R. (2007). Interactive multimodal learning environments. *Educational Psychology Review, 19*(3), 309–326.

Mory, E. H. (2004). Feedback research revisited. *Handbook of research on educational communications and technology*, *2*, 745–783.

Muñoz-Merino, P. J., Kloos, C. D., & Muñoz-Organero, M. (2011). Enhancement of student learning through the use of a hinting computer e-learning system and comparison with human teachers. *IEEE Transactions on Education, 54*(1), 164–167.

Narciss, S. (2008). Feedback strategies for interactive learning tasks. *Handbook of research on educational communications and technology*, *3*, 125–144.

Narciss, S., Sosnovsky, S., Andres, E. (2014a). Adapting tutoring feedback strategies to motivation. In *Open Learning and Teaching in Educational Communities* (pp. 288–301). Springer International Publishing.

Narciss, S., Sosnovsky, S., Schnaubert, L., Andrès, E., Eichelmann, A., Goguadze, G., & Melis, E. (2014b). Exploring feedback and student characteristics relevant for personalizing feedback strategies. *Computers & Education, 71*, 56–76.

Perikos, I., Grivokostopoulou, F., Hatzilygeroudis, I., Kovas, K. (2011). Difficulty estimator for converting natural language into first order logic. In *Proceedings of the International Conference on Intelligent Decision Technologies* (pp. 135–144). Springer Berlin Heidelberg.

Perikos, I., Grivokostopoulou, F., Hatzilygeroudis, I. (2012). Automatic marking of NL to FOL conversions. In *Proceedings of 15th IASTED International Conference on Computers and Advanced Technology in Education (CATE), Napoli, Italy* (pp. 227–233).

Perikos, I., Grivokostopoulou, F., Kovas, K., & Hatzilygeroudis, I. (2016). Automatic estimation of exercises' difficulty levels in a tutoring system for teaching the conversion of natural language into first-order logic. *Expert Systems, 33*(6), 569–580.

Roll, I., Aleven, V., McLaren, B. M., & Koedinger, K. R. (2011). Improving students' help-seeking skills using metacognitive feedback in an intelligent tutoring system. *Learning and Instruction, 21*(2), 267–280.

Roll, I., Baker, R. S. D., Aleven, V., & Koedinger, K. R. (2014). On the benefits of seeking (and avoiding) help in online problem-solving environments. *Journal of the Learning Sciences, 23*(4), 537–560.

Russell, S., & Norvig, P. (2010). *Artificial intelligence: a modern approach* (3rd ed.). Upper Saddle River: Prentice Hall.

Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research, 78*(1), 153–189.

Sieg, W. (2007). The AProS project: strategic thinking & computational logic. *Logic Journal of IGPL, 15*(4), 359–368.

van der Kleij, F. M., Eggen, T. J., Timmers, C. F., & Veldkamp, B. P. (2012). Effects of feedback in a computer-based assessment for learning. *Computers & Education, 58*(1), 263–272.

VanLehn, K. (2006). The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education, 16*(3), 227–265.

VanLehn, K. (2011). The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. *Educational Psychologist, 46*(4), 197–221.

VanLehn, K., Lynch, C., Schulze, K., Shapiro, J. A., Shelby, R., Taylor, L., & Wintersgill, M. (2005). The Andes physics tutoring system: lessons learned. *International Journal of Artificial Intelligence in Education, 15*(3), 147–204.

Vasilyeva, E., Pechenizkiy, M., De Bra, P. (2008). Adaptation of elaborated feedback in e-learning. In *Adaptive hypermedia and adaptive web-based systems* (pp. 235–244). Springer Berlin Heidelberg.

Wager, W., & Wager, S. (1985). Presenting questions, processing responses, and providing feedback in CAI. *Journal of Instructional Development, 8*(4), 2–8.

Wang, S. L., & Wu, P. Y. (2008). The role of feedback and self-efficacy on web-based learning: the social cognitive perspective. *Computers & Education, 51*(4), 1589–1598.

Yacef, K. (2002). Intelligent teaching assistant systems. In *Computers in Education, 2002. Proceedings International Conference on IEEE* (pp. 136–140).

Yacef, K. (2005). The Logic-ITA in the classroom: a medium scale experiment. *International Journal of Artificial Intelligence in Education, 15*(1), 41–62.

Zhou, Y., Freedman, R., Glass, M., Michael, J. A., Rovick, A. A., Evens, M. W. (1999). Delivering Hints in a Dialogue-Based Intelligent Tutoring System. In *AAAI/IAAI* (pp. 128–134).