JOURNAL OF APPLIED
COMPUTER SCIENCE
Vol. 18 No. 2 (2010), pp. 57-65

The Gentzen Sequent Calculus in E-Testing. Part II: Algorithms and Implementation

Adam Niewiadomski[1], Andrzej Indrzejczak[2]

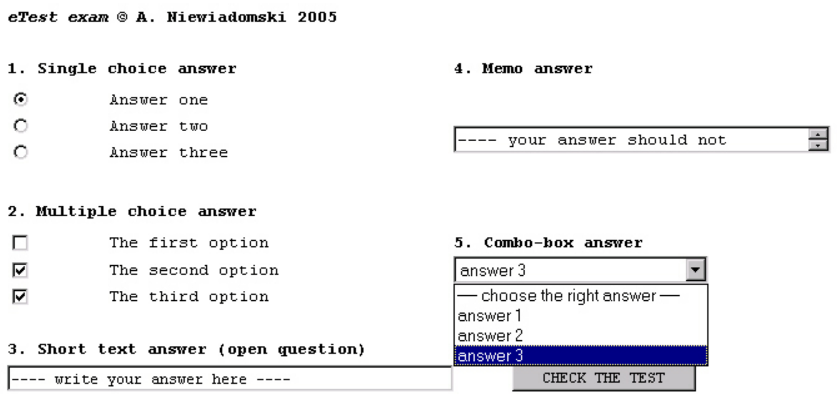
[1]*Institute of Information Technology, Technical University of Łódź*
Wólczańska 215, 90-924 Łódź, Poland
aniewiadomski@ics.p.lodz.pl

[2]*Department of Logics and Methodology of Sciences*
University of Łódź
Kopcińskiego 16/18, 90-232 Łódź, Poland
indrzej@filozof.uni.lodz.pl

Abstract. *This paper contains the description of algorithms and their implementations, the foundations and background of which are presented in Part I of this paper [1].*
Keywords: *E-learning, e-testing methods, Gentzen's sequent calculus, evaluating and scoring of e-tests, decision support.*

1. The GUI of E-tests Evaluated by Sequent Methods

Some possibilities of intelligent support for e-testing are enumerated in Part I. In this section, with respect to applications of the method to be introduced, we intend to describe the design of a graphical user interface in e-testing modules of e-learning platforms. Presented ideas are partially inspired by the kind of test which can be resolved by the algorithm presented in Section 2, nevertheless the authors intend also to comment on some general rules of running e-tests.



eTest exam © A. Niewiadomski 2005

1. Single choice answer

[✓ Answer one]
Answer two
Answer three

2. Multiple choice answer

☐ The first option
☑ The second option
☑ The third option

3. Short text answer (open question)

---- write your answer here ----

4. Memo answer

---- your answer should not

5. Combo-box answer

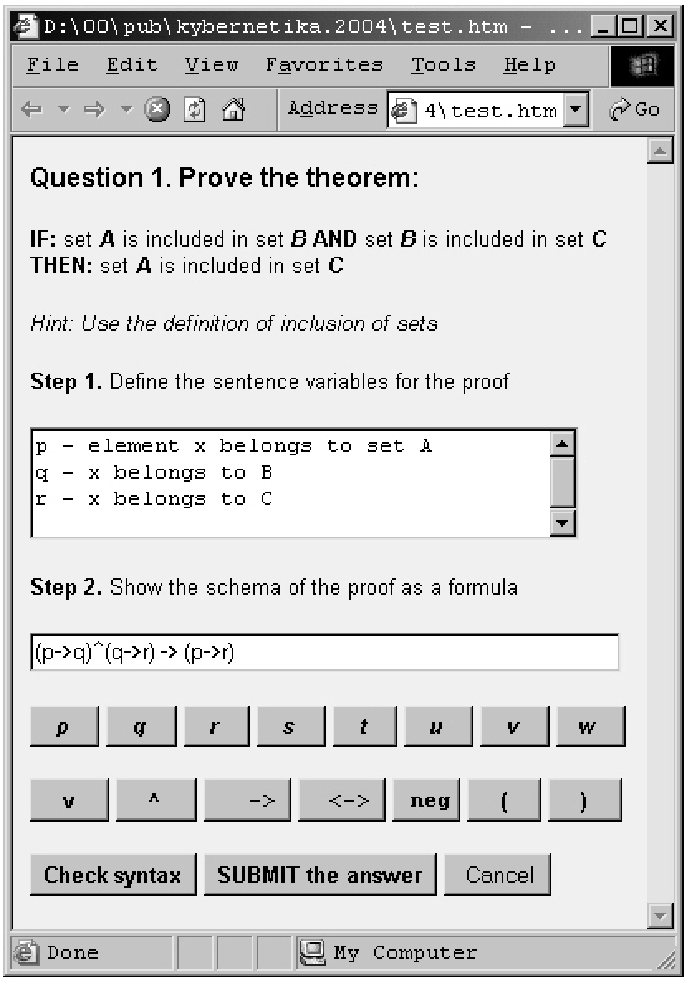
answer 3

— choose the right answer —
answer 1
answer 2
[✓ answer 3]

Figure 1. A sample GUI of e-test

A typical look of a graphical user interface in the e-tests which do not require intelligent methods to be checked or evaluated, can be composed as given in Fig. 3. The figure presents a sample e-test; its structure is simple and does not require any wider comments, instead of a short description of methods which can be used in evaluating students' answers to questions 1.–5. Questions 1., 2., and 5. are typical for the so-called *choice tests*, 1. and 5. – single choice tests, and 2. – multiple choice test. A very basic method based on the Hamming distance is sufficient to determine the correctness of such the test composed of such questions. Questions 3. and 4. require of typing some text as an answer, the former is intended to collect short (max. 5–6 word), while the latter allows to input longer (e.g. 50 words) answers. Determining correctness of such answers can be based on crisp matching ("matches or does not match"), on distances between students' answers and a template of correct answer (the Loewenstein distance, the edit distance [2], or on similarity measures for the units of textual information (basic [3] and generalized [4] *n*-gram methods, String Kernels, etc.).

Let us now introduce a sample GUI for a test in set theory, in which answering the questions requires from students to present a reasoning process or a proof (another domains in which such a test can be run, are mathematics, physics, logic, etc.). See Figure 2 which presents the two steps of answering a question (a theorem which shall be proven). A student is provided with a very clear notation of the


D:\00\pub\kybernetika.2004\test.htm - ...

File Edit View Favorites Tools Help

Address 4\test.htm Go

Question 1. Prove the theorem:

IF: set ***A*** is included in set ***B*** **AND** set ***B*** is included in set **C**
THEN: set ***A*** is included in set **C**

Hint: Use the definition of inclusion of sets

Step 1. Define the sentence variables for the proof

```
p - element x belongs to set A
q - x belongs to B
r - x belongs to C
```

Step 2. Show the schema of the proof as a formula

```
(p->q)^(q->r) -> (p->r)
```

Figure 2. A test in the set theory. Step 1: Preparation of the proof. Step 2: Syntax analysis and proving

theorem and a hint. The first step of answering such a question is the assignment of sentence variables to the statements used in the proof. Notice that symbolical representation for all elements of the proof is required. As it is shown in Fig. 2, a student can choose from a few sentence variables, denoted as $p, q, r \ldots$; also the logical operators of disjunction $\vee$, conjunction $\wedge$, implication $\rightarrow$, equivalence $\leftrightarrow$, and negation $\neg$, as well as left and right brackets can be achieved by clicking the related buttons (which is easier than typing them). A very important detail is that a student has the possibility of checking the syntax of the formula by clicking the ”Check syntax” button, to make sure that his/her answer will not be rejected due to formal reasons (like mismatched brackets, improper number of arguments of the logical operators, transitive implications, e.g. $p \to q \to r$, etc.). When the proof is ready to be evaluated, the student clicks the ”Submit the proof” button, and the answer (the content of the edit box in step 2) is saved in the database.

The algorithm for checking correctness of proofs – answers given by students – is described in Section 2. It is worth underlining that the algorithm is not able to replace at all the participation of a human teacher, nevertheless, it enables automated separation of correct answers from others (partially and totally incorrect). The task which is not possible to be undertaken by an algorithm (at least, in the presented solution) is determining of correctness and relevancy of sentence variables on the base of which a student constructs his/her answer, i.e. the formula to be proven to be a thesis.

There are some additional elements and properties which should be considered and discussed when constructing a GUI for an e-test:

- the question must be clearly formulated, without any doubts which may confuse a student or may provoke the necessity of any additional advice from a tutor;
- answering the question should be divided into two steps: 1. preparation of the proof, and 2. writing of the formula of the proof[1];
- the interface must be user-friendly, easy-to-operate, and non-confusing;
- a help system (”F1”) must be accessible during all the time the test is being taken; moreover, the small text labels with hints should be displayed on each element of GUI if only the mouse cursor is hanged over that a bit longer;
- the time for an answer must be limited, which is related to the fact that...
- ...a student must be able to choose whether the time is displayed on the screen or not (some students prefer to control continuously the time of the test, on the other hand, it is the factor which can increase the level of stress);

[1]The note concerns this specific kind of test only.

- a student must know the number of questions in the test (if they are not displayed all on the same screen, but separated into a few screens with one question each[2]);
- the ”Check syntax” and ”Cancel” buttons must be available for each question separately;
- the limitation of number of revoking the ”Check syntax” procedure can also be considered.

Finally, the ”last but not least” condition which must be respected is to perform the presented kind of a test only for the students who graduated from the basic course of classical two-valued logic including the domain of automated theorem proving.

2. The Answer-Evaluating Algorithm

The following algorithm is responsible for determining whether the given student's answer is correct (i.e. it represents a correct and valid proof) or not.

```
1. get student's answer as a string
2. pre-process the string to obtain the list of formulae
   on the right side of the sequent symbol $\Rightarrow$
3. IF (there are formulae for which the rules
   can be applied) {
      3.1 choose the formula to be processed
      3.2 choose the main operator in the formula
      3.3 apply the rule for this operator
      3.4 goto 3.
   }
   ELSE {
      3.5 IF (for each branch of the tree its both sides
              have a non-empty common intersection)
             return true
          ELSE
```

[2]In case when the time of the test is possible to be displayed, all the questions must provide the same time for finding an answer; otherwise, different time limits for questions can suggest some differences in the difficulty levels of these questions.

```
            return false
   }
4. STOP
```

Ad. 3.1: Choosing the formula which is to be processed before the others is a very crucial moment of the algorithm – it influences to a great extent the number of steps in which the proof is built. In this implementation, simply the non-branching rules, e.g. ($\Rightarrow\rightarrow$) are prior the branching ones. Other various strategies can also be applied and implemented.
Ad. 3.5: The `true` and `false` values returned as results in step 5. mean that the formula given by a student is provable or not, respectively.

Naturally, the algorithm is anticipated by the procedure which checks grammatical correctness of the formula (braces matching, numbers of arguments of logical operators, etc.).

Example Let student's answer be given by the following formula:

$$[(p \rightarrow q) \wedge p] \rightarrow q \tag{1}$$

The evaluating procedure for this formulae works as it is presented below:

```
1. Take $[(p \rightarrow q) \wedge p] \rightarrow q$ as the string representing
   student's answer
2. Denote the string as the right side
   of the sequent $\Rightarrow [(p \rightarrow q) \wedge p] \rightarrow q$
3. Process the tree of the proof:
   3.1 The formula to be processed is $[(p \rightarrow q) \wedge p] \rightarrow q$
   3.2 The main operator in the formula is $\rightarrow$
   3.3 Apply rule ($\Rightarrow\rightarrow$).
       The obtained sequent is $(p \rightarrow q) \wedge p \Rightarrow q$
   3.1' The formula to be processed is $(p \rightarrow q) \wedge p$
   3.2' The main operator is $\wedge$
   3.3' Apply rule ($\wedge\Rightarrow$).
       The obtained sequent is $(p \rightarrow q), p \Rightarrow q$
   3.1" The formula to be processed is $p \rightarrow q$
   3.2" The main operator is $\rightarrow$
   3.3" Apply rule ($\rightarrow\Rightarrow$).
        The obtained sequents are: $q, p \Rightarrow q$ and $p \Rightarrow q, p$
```

```
    // no more formulae with logical operators to apply the rules on them

    3.5 There are two branches in the tree:
          one is finished with $q, p \Rightarrow q$
          and the other with $p \Rightarrow q, p$.
          return true
    // formula is provable, thus it is a thesis
4. STOP
```

The visualization of the algorithm is additionally equipped with the function which shows those places in trees of proofs in which errors (mismatching of the left to the right side of a sequent) occur.

It must be explained that the presented algorithm is able to determine whether a proof is correct or not, i.e. whether the formula written by a student is a thesis or it is not (since the grammatical errors are eliminated via the procedure revoked by clicking the ”Check syntax” button. When a teacher is supported with the tool based on the presented algorithm, his/her only task is to compare the correctness of assignment of sentence variables with the correctness of the proof presented by a student, and, finally to score the answer according to a chosen scale of marks.

3. Implementation Details

The chosen implementation technology is Java Server Pages, Java Servlets, and Java-Beans. Especially, the MVC architecture (Model-View-Controller) is applied since it enables encapsulation of such tasks as database connections (Model), presenting data as HTML (View), and running procedures for parsing students answers and determining their correctness (Controller). The questions presented in tests and the answers given by students are stored in the form of XML documents to enable easy and quick presenting them in both HTML and text formats (the former is used to display data on user's screen, while the latter provides textual data for parsing and proving procedures). In addition, the tag library JSTL (JSP Standard Tag Library by Sun Microsystems) is applied. The implementation has been supported by an open-source project Jakarta Struts Framework[3].

The prototype application has been tested on Linux Slackware 10 operating system, with MySQL 4.0.1 database engine, and with Apache Jakarta Tomcat 5.0.19 application server.

[3]http://jakarta.apache.org

4. Conclusions

Gentzen's sequent calculus and the systems based on it, see Part I., are obviously not the only methods of automated theorem proving. Many other examples are: the method of natural deduction invented independently by Polish logician Jaskowski [5] and Gentzen [6] or truth table tests. Nevertheless, Gentzen's calculus has been chosen as the base of evaluating algorithm due to the following technical reasons:

a) Each proof of a theorem in Gentzen's system is a binary tree which branches out; with respect to the fact that many functions and class libraries support building the tree data structure (e.g. Tree classes in C/C++ or C#), as well as controlling the textual content of tree's nodes (e.g. Java API 1.6, `final class String extends Object`), implementation of the proving algorithm in a high level language is not a complicated task.

b) In comparison to Jaśkowski's system, Gentzen's SC does not use indexing of rows in a proof (due to the implementation, indexing could require some additional and unnecessary information to be processed). From the implementation point of view, it allows to avoid recursive and costly procedures, which may additionally accomplish (and lengthen) the testing phase.

c) Gentzen's system is based on clearly defined rules which allow converting the rows of a proof one to another; as opposite to Jaśkowski's method, in which the primary and secondary rules are distincted (moreover, the number of the latters is uncountable), the number of possible actions (decisions) in a next row of the proof is meaningfully limited. After all, it is possible to establish priorities and criteria according to which the Gentzen rules are applied.

d) In contrary to truth-table tests, Gentzen's sequent calculus is much less sensitive to network transfer errors (swapping single bits, especially, when changing unity to zero or vice-versa could be disastrous!). In our method, rows of the proof are stored and encapsulated as strings (java.lang.String objects), hence there is no risk of losing some pieces of data.

e) Gentzen's calculus is decidable for classical propositional logics which means that each formula can be shown as a provable or not in a finite time.

References

[1] Niewiadomski, A. and Indrzejczak, A., *The Gentzen Sequent Calculus in E-testing. Part I: Foundations*, Journal of Applied Computer Science, Vol. 18, No. 1, 2010, pp. 39–48.

[2] Mitra, S. and Acharya, T., *Data Mining. Multimedia, Soft Computing, and Bioinformatics*, John Wiley & Sons, Inc. Publication, 2003.

[3] Bandemer, H. and Gottwald, S., *Fuzzy Sets, Fuzzy Logic, Fuzzy Methods with Applications*, John Wiley & Sons, 1995.

[4] Niewiadomski, A., Rybusiński, B., Sakowski, K., and Grzybowski, R., *An application of multivalued similarity relations in automated evaluation of grammar tests*, In: The Online Academy (Akademia On-Line), edited by J. Mischke, WSHE Press, 2005, pp. 149–154, (in Polish).

[5] Jaśkowski, S., *On the Rules of Suppositions in Formal Logic*, Studia Logica, Vol. 1, 1934, pp. 5–32.

[6] Gentzen, G., *Untersuchungen Über das Logische Schliessen*, Mathematische Zeitschrift, Vol. 39, 1934, pp. 176–210, 405–431.