[Practical Paper] LASP — a Learning Assistant System for Formal Proofs — and Its Application to Education

Kiyoyuki Miyazawa[†], Kozo Okano[†], and Shinji Kusumoto[†]

[†]Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita City, Osaka 565-0871, Japan

Abstract - The basis of formal techniques is mathematical logic. Especially, it is important to understand the concept of formal proofs. It is, however, difficult for novices to study formal proofs because of the rigorousness of the operations involved. To solve this problem, we propose and implement a prototype of a learning assistant tool for formal proofs called LASP. The purpose of LASP is to prevent the difficulties which occur when normal learners perform exercises by hand. The advantages of the proposed method are the following: 1. input support for long logical expressions; 2. the users are not required to perform copy and paste operations when they construct proofs; 3. hint features facilitate the construction of proofs; and 4. the proofs can be output as LaTeX files. Experimental results show that LASP avoids the drawbacks of conventional exercises.

Keywords: Logics, Formal Proof, Computer Aided Education

1 Introduction

The formal approach is a core technique of modern software development. There are mainly two approaches referred to as the formal approach: interactive theorem proof and the model-based approach, which includes model checking[2]. Well-known interactive theorem proof systems include Coq[6], Agda[7], and Isable/HOL, among other. These systems require the user to understand the logic of formal proofs. Model checking also requires the user to apply logic in order to represent properties on the target system. Other model-based approaches also require logic to describe constraints on models representing the target systems and software.

On the other hand, it is said that "Mathematical reasoning is intrinsic to both traditional engineering and software engineering, . . . Software engineers usually use discrete mathematics and logic in a declarative mode for specifying and verifying system behaviours and for analysing system features"[11]. This statement illustrates how it is important to understand mathematical logic in software engineering. Supporting the learning of mathematical logic can promote the learning of basic techniques of the formal approach. We can promote productivity in software development by teaching the formal approach.

A learning assistant system which we developed in this research supports learning formal proofs based on Hilbert's axiom system. As related works, Jon Barwise et al. have developed Turing's World[4] and Tarski's World[3], [5]. One can learn graphically Turing machine using Turing's World, and we can learn graphically the semantics of mathematical logic using Tarski's World. Tarski's World shows simple 3D computer graphic worlds in which geometric blocks of various kinds and sizes are distributed. Tarski's World gives you a first-order predicate sentence and a 3D figure and lets you decide whether the sentence is true or false for the provided figure. For example, a sentence "there exists a cube among the objects" is given and a user has to decide whether the sentence is true by looking at the figure. Three-dimensional views are sometimes used in computer-aided education. For example, paper[12] provides a 3D geometric construction tool specifically designed for mathematics, especially geometry, education. The tool is based on a mobile augmented reality (AR) system.

Another approach for computer-aided education is using a collaborative web-based environment, such as Moodle or WEB-CT. Several other approaches have been proposed, including paper [10], which provides a flexible learning scheme for selected pilot courses in engineering education. In such schemes, traditional lectures and written exercises are combined with separate Web-based learning resources.

In the case of logic-oriented education, MacLogic[9] supports the learning of Gentzen's natural deduction. Logic-Tutor[1] supports students as they are learning logic. It has a feature for analysing and reasoning a student's mistakes during the solving of logic exercises. It has been used in logic classes in the Department of Computer Science of the University of Sydney.

In this research, we target novices in logic and provide tools for them. Therefore, we do not consider that simply applying tools such as conventional theorem provers [6], [7] to education is suitable, although others have used a similar approach. For example, "Bringing Research Tools into the Classroom"[14] helps move computational tools used for research into the classroom. It successfully brings high- performance computing into modelling courses, builds software for both protein structure visualization and hydrological analysis of watersheds, and so on. A project of Buchberger [8] aims at supporting the entire process of mathematical theory exploration within one coherent logic and software system. This project uses formal logic and a computer-aided approach to help a learner to understand the core of mathematics.

While other related research mainly focuses on understanding semantics, here we focus on understanding formal proofs. For understanding the semantics of logic, an approach like Tarski's World, which utilizes 3D graphics, is useful. We, however, focus on formal proofs, in which 3D graphics may be of little help. In addition, our tool deals with Hilbert's axiom system because the class on mathematical logic held

Table 1: Axioms for First-order Logic

A1	$P \to (Q \to P)$
A2	$(P \to (Q \to R)) \to ((P \to Q) \to (P \to R))$
A3	$(\neg P \to \neg Q) \to ((\neg P \to Q) \to P)$
A4	$\forall x\, T(x) \to T(t)$, where term t is free from variable x of $T(x)$.
B1	$P, P \to Q \vdash Q$ (MP, modus ponens)
B2	$P \to Q \vdash P \to \forall x\, Q$, where term P is free from a free variable x

in the Department of Informatics and Mathematical Science of Osaka University deals only with this system. As far as we know, no tools for learning Hilbert's formal proof exist. Therefore, we developed our tool and evaluated its effectiveness.

The paper is organised as follows. Section 2 presents a brief introduction of formal proofs. Sections 3 and 4 describe our tool and an experimental evaluation, respectively. Section 5 presents a discussion. Finally, Section 6 concludes our paper.

2 Formal Proof

A formal proof is a process which proves theorems by axioms and inference rules. An axiom is a major premise to derive a concrete logical expression instance without contradictions, while an inference rule is used to derive a new logical expression from proved ones. For example, modus ponens is an inference rule which derives Q from $P \to Q$ and P (where P and Q are arbitrary logical expressions).

An axiom system consists of axioms and inference rules, and the logical expressions proved by an axiom system are called theorems. Figure 1 shows an example of the formal proof which proves a theorem of $\vdash X \to X$.

Table 1 shows Hilbert's axioms and inference rules for first-order logic.

3 LASP, a Learning Assistant System for Formal Proofs

In this section, we will present our tool "Learning Assistant System for Proofs, (LASP)."

3.1 System Overview

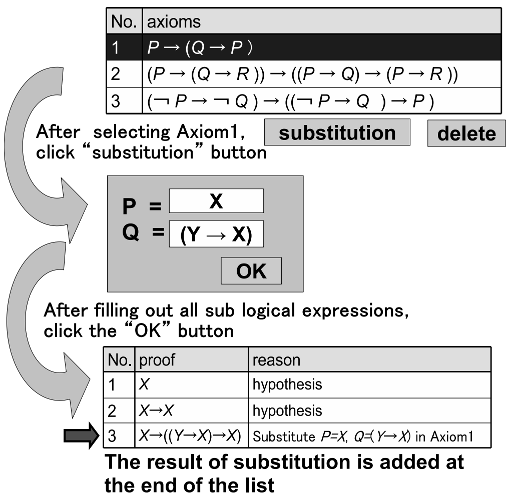
We developed LASP to reduce problem practice time relative to writing proofs by hand. This tool is based on Hilbert's axiom system.

The drawbacks of practicing formal proof by hand are the following.

1. We may make a mistake in the writing of a long expression corresponding to an axiom, especially the correspondence between terms and variables.

2. We are often required to perform operations similar to copy and paste when constructing a formal proof.

To resolve the above problems, we define the goal as follows.



No.	axioms
1	$P \to (Q \to P)$
2	$(P \to (Q \to R)) \to ((P \to Q) \to (P \to R))$
3	$(\neg P \to \neg Q) \to ((\neg P \to Q) \to P)$

After selecting Axiom1, click "substitution" button

After filling out all sub logical expressions, click the "OK" button

No.	proof	reason
1	X	hypothesis
2	$X \to X$	hypothesis
3	$X \to ((Y \to X) \to X)$	Substitute $P = X$, $Q = (Y \to X)$ in Axiom1

The result of substitution is added at the end of the list

Figure 1: Substitution

1. LASP should have an interface to minimise the input of logical expressions.

2. We should not be required to perform operations like copy and paste.

LASP is implemented in Java with 8660 LOC. The system is available on the WEB[13].

We will next describe the specific features and interfaces of LASP.

3.2 Substitution Assistant

We implemented a substitution assistant feature in order to reduce time wasted during the proof; this feature also allows users to avoid careless written mistakes, such as inconsistencies between variables and terms, sentence structures, and variable name. Such mistakes also often occur when handwriting long and complex logical expressions. These mistakaes are unrelated to learning essential proofs. Instead, it is important to learn the thinking process of problem resolution in mathematical logic.

Let us consider the following example. Substitution of variables P, Q, and R of axiom A2 in Table 1 with the terms $X \to Y$, Z, and $((Y \to Z) \to X)$, respectively, yields the logical expression (1), which is obviously long and complex. This expression is also hard to read and write correctly.

$$\begin{aligned} ((X \to Y) &\to (Z \to ((Y \to Z) \to X))) \\ &\to (((X \to Y) \to Z) \to ((X \to Y) \\ &\to ((Y \to Z) \to X))) \end{aligned} \tag{1}$$

Substitution Assistant automatically generates new logical expressions when a user selects an arbitrary axiom and inputs arbitrary logical expressions as the substitution terms for that axiom.

Figure 1 shows a substitution flow on LASP.

Axioms are managed using a table as shown at the top of Fig. 1. When a user clicks on the axiom in the table which he wants to use, the Substitute Panel opens. If he inputs logical expressions for every propositional variable in Substitute Panel and clicks the OK button, then the result expression is added at the end of the proof list.

Such a feature has already been built into many interactive theorem proof systems. However, these systems are not easy to use in undergraduate classes.

3.3 Support for First-order Logic

LASP supports not only propositional logic but also first-order logic.

We have to consider whether the variable is free or bound when we want to perform substitution for a variable of an expression in first-order logic. For example, variable y of a logical expression (2) is bound by $\exists y$. Therefore, this variable cannot be substituted.

$$\forall x \exists y f(y, z) \tag{2}$$

Furthermore, any term including variable x or y cannot be also substituted into variable z of expression (2), because such substitution resulted bound by the quantifier. If expression $g(x)$ was substituted into z, the result would be the logical expression (3).

$$\forall x \exists y f(y, g(x)) \tag{3}$$

In the expression, variable x is bound by $\forall x$ and the semantic of the expression (3) is different from the expression (2). LASP generates an exception if an illegal substitution like the above occurs by using a substitution inhibition list.

3.4 Inference Assistant

Inference Assistant is implemented to reduce the practice time by reducing the time for operations such as copy and paste that take considerable time when performing by hand.

When a user selects an inference rule that he wants to apply and proven logical expressions that conform with inference rule, a new logical expression that corresponds to the rule is generated. The reason for the new logical expression is also generated, which reduces the amount of time needed to handwrite reasons.

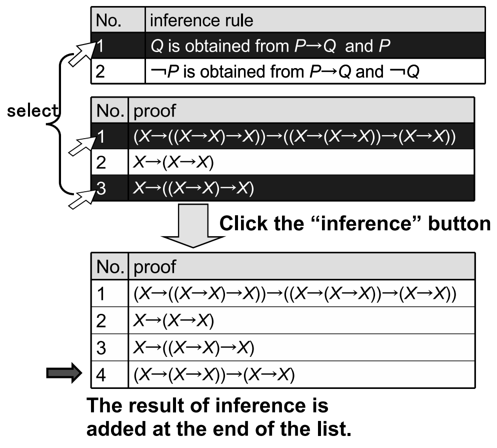
Figure 2 shows the flow of the application of an inference rule on LASP. Inference rules and proven theorems are displayed in a table. First, a user clicks to select the inference rule which he wants to apply from the inference rule table. Second, he clicks to select the proven theorems matching the inference rule. Finally, he clicks the inference button. If he selects the proven theorems correctly, then a new logical expression is added at the end of the proven theorem table.

Figure 3 shows the screen-shot of LASP.

3.5 Deduction Theorem Assistant

The deduction theorem is a useful theorem for proving theorems efficiently. Therefore, LASP also supports the deduction theorem, which is the following.

From the fact that $\Gamma, P \vdash Q$, we obtain $\Gamma \vdash P \to Q$.



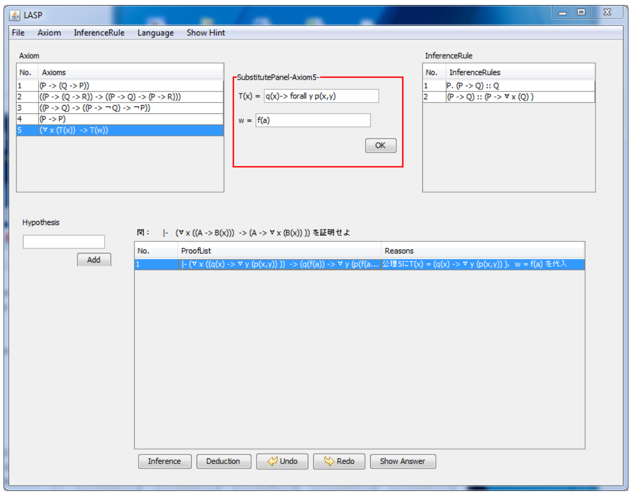
No.	inference rule
1	Q is obtained from $P \to Q$ and P
2	$\neg P$ is obtained from $P \to Q$ and $\neg Q$

No.	proof
1	$(X \to ((X \to X) \to X)) \to ((X \to (X \to X)) \to (X \to X))$
2	$X \to (X \to X)$
3	$X \to ((X \to X) \to X)$

No.	proof
1	$(X \to ((X \to X) \to X)) \to ((X \to (X \to X)) \to (X \to X))$
2	$X \to (X \to X)$
3	$X \to ((X \to X) \to X)$
4	$(X \to (X \to X)) \to (X \to X)$

select
Click the "inference" button
The result of inference is added at the end of the list.

Figure 2: Flow of constructing a proof with LASP



Axiom

No.	Axioms
1	(P -> (Q -> P))
2	((P -> (Q -> R)) -> ((P -> Q) -> (P -> R)))
3	((P -> Q) -> ((P -> ¬Q) -> ¬P))
4	(P -> P)
5	(∀ x (T(x)) -> T(w))

SubstitutePanel-Axiom5-

T(x) = q(x)-> forall y p(x,y)

w = f(a)

InferenceRule

No.	InferenceRules
1	P. (P -> Q) :: Q
2	(P -> Q) :: (P -> ∀ x (Q))

問： |- (∀ x ((A -> B(x))) -> (A -> ∀ x (B(x)))) を証明せよ

No.	ProofList	Reasons
1	|- (∀ x ((q(x) -> ∀ y (p(x,y)))) -> (q(f(a)) -> ∀ y (p(f(a...	公理5にT(x) = (q(x) -> ∀ y (p(x,y))), w = f(a) を代入

Figure 3: Screenshot of LASP

Table 2: Fill-in-the-blank hints

1		Hypothesis
2		"?" is substituted in axiom "?"
3		"A" and "?" are substituted for "P" and "Q," respectively, in axiom "1"
4	$(\neg A \rightarrow A)$	Applying the inference rule "?"
5		"?" and "?" are substituted for "?" and "?" in axiom "?"
6	$((\neg A \rightarrow \neg A) \rightarrow \neg\neg A)$	Applying the inference rule "?"
7		Applying the inference rule "?"
8	$(A \rightarrow \neg\neg A)$	Deducing "?" from proven theorem "?"

Table 3: Milestone hints

1		
2		
3		
4	$(\neg A \rightarrow> A)$	Applying the inference rule "1" to proven theorems "1" and "3"
5		
6	$((\neg A \rightarrow \neg A) \rightarrow \neg\neg A)$	Applying the inference rule "1" to proven theorems "4" and "5"
7	$\neg\neg A$	Applying the inference rule "1" to proven theorems "2" and "6"
8		

First, a user selects a proven theorem to which he wants to apply the deduction theorem. Second, he clicks the deduction button. Then, the deduction panel opens. He clicks a radio button to select a theorem to which to apply the deduction theorem and then clicks the OK button. Finally, the result of applying the deduction theorem is added at the end of the proven theorem table.

3.6 Hint Features

The hint features provide three levels of hints for users who are unfamiliar with formal proofs. These features were implemented after obtaining feedback from Experiment 1, which is described in Section 4. The specific feedback is that it is difficult to solve exercises which have many steps by oneself. The type of hints prepared by LASP are as follows.

1. fill-in-the-blank hint
2. milestone hint
3. next step hint

Table 2 shows examples of fill-in-the-blank hints for a proof of $A \rightarrow \neg\neg A$ under the assumption that $P \rightarrow P$ has already been proved. This hint is generated randomly when LASP reads the data of an exercise.

Table 3 shows examples of milestone hints for the same proof. The hint feature shows expressions that can be proved by applying the inference rule.

We assume that a theorem $P \rightarrow P$ is given as axiom A4. Table 4 shows a typical solution.

3.7 Undo/Redo Features

LASP supports undo/redo features. Learners can revise their proofs using these features.

It would be better if LASP supported a feature in which a user can edit his/her proof after he/she has completed it. However, such a feature might destroy the reasoning chain of the proof. Thus, the current version of LASP only supports Undo/Redo features. Undo/Redo works for as many steps as the length of the proof users have made.

3.8 Input/Output Features

Logical expressions are often long and deeply nested. Therefore, the string representation of a logical expression is sometimes difficult to understand. This feature visualises the parse tree of a logical expression (see the example in Fig. 4). Clicking a node, the subtree is expanded or collapsed (Fig. 5).

Table 4: Solution

1	A	Hypothesis
2	$\neg A \rightarrow \neg A$	P is substituted into axiom A4
3	$A \rightarrow (\neg A \rightarrow A)$	"A" and $\neg A$ are substituted for "P" and "Q," respectively, in axiom "1"
4	$(\neg A \rightarrow A)$	Applying the inference rule "1" to proven theorems "1" and "3"
5	$((\neg A \rightarrow A) \rightarrow (\neg A \rightarrow \neg A) \rightarrow \neg\neg A)$	"$\neg A$" and A are substituted for "P" and "Q," respectively, in axiom "3"
6	$((\neg A \rightarrow \neg A) \rightarrow \neg\neg A)$	Applying the inference rule "1" to proven theorems "4" and "5"
7	$\neg\neg A$	Applying the inference rule "1" to proven theorems "2" and "6"
8	$A \rightarrow \neg\neg A$	Applying the deduction theorem

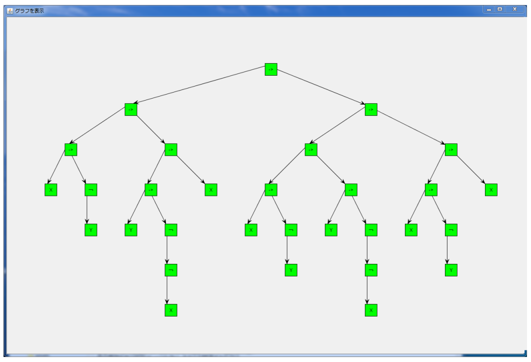


Figure 4: Parse tree representation

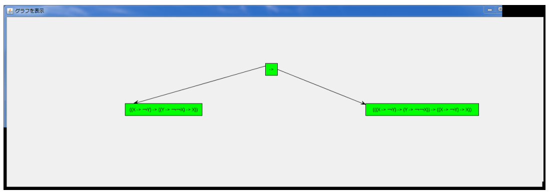


Figure 5: Parse tree representation (collapsed)

LASP also supports the LaTeX format for the output of constructed proofs. LaTeX supportof LASP also helps users, especially for producing reports or homework.

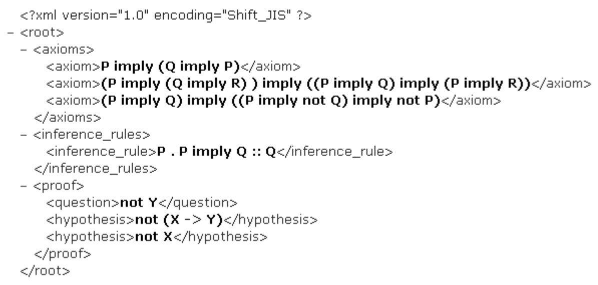
The exercises of LASP are input as XML files (Fig 6). Therefore, a teacher can easily prepare new sets of exercises with hints.

4 Evaluation

This section describes an evaluation of LASP. We performed two experiments. Experiment 1 was conducted in a mathematical logic class. The subjects were 50 undergraduate students. Experiment 2 was conducted after reflecting on the feedback received from Experiment 1. In this experiment, the number of subjects was 16. The subjects in Experiment 2 were a doctoral student, ten master course students, and four undergraduate students.

4.1 Goals of the Evaluation

The objective in these experiments was as follows: to measure the degree of effectiveness for users, and to collect feedback to make LASP more useful.



```
  <?xml version="1.0" encoding="Shift_JIS" ?>
- <root>
  - <axioms>
      <axiom>P imply (Q imply P)</axiom>
      <axiom>(P imply (Q imply R) ) imply ((P imply Q) imply (P imply R))</axiom>
      <axiom>(P imply Q) imply ((P imply not Q) imply not P)</axiom>
    </axioms>
  - <inference_rules>
      <inference_rule>P . P imply Q :: Q</inference_rule>
    </inference_rules>
  - <proof>
      <question>not Y</question>
      <hypothesis>not (X -> Y)</hypothesis>
      <hypothesis>not X</hypothesis>
    </proof>
  </root>
```

Figure 6: The output XML file

4.2 Items of the Evaluation

In Experiment 1, we mainly investigated how effectively a user can solve an exercise and the usability of LASP. The degree of efficiency was measured as the time it took subjects to finish solving the given problems.

As the evaluation of usability, we prepared questionnaires to research how subjects can become familiar with LASP and what kind of interface is needed to enhance usability. The items of the questionnaires are summarised as follows.

$Q1$ The degree of user-friendliness of Substitute Assist.

$Q2$ The degree of user-friendliness of Inference Assist.

$Q3$ The degree of user-friendliness of LASP as a whole.

$Q4$ The degree of efficiency improvement using LASP.

$Q5$ The degree of reduction of trouble in proving from using LASP.

$Q6$ The degree of effects of learning formal proof.

The items are on a scale of one to five, where a five means a high degree. A free comment space to collect opinions about LASP or the experiment is also included.

4.3 Procedure of Experiments

The procedure of Experiment 1 is as follows.

1. We divide all subjects into six groups.
2. Subjects solve two practice problem by hand.
3. Two weeks later, subjects solve two practice problems by using LASP.
   (a) We distribute tool manuals to all subjects.
   (b) We let subjects use LASP to solve sample exercises for 20 minutes.
   (c) Subjects solve the exercises.
4. Subjects answer the questionnaire.

There are four exercises in all. The order of the exercises is randomly chosen for each group. The time limit that a subject has to solve an exercise is 15 minutes.

The procedure of Experiment 2 is as follows.

1. We distribute tool manuals and textbooks about mathematical logic to all subjects.
2. We give all subjects one hour to solve four sample exercises and familiarise them with LASP.
3. Subjects alternate between solving exercises by hand and using LASP.
4. Subjects answer the questionnaires.

Table 5: Results of the questionnaires in Experiments 1 and 2

item	Exp.1	Exp.2
$Q1$	3.70	4.69
$Q2$	3.28	4.56
$Q3$	2.84	4.13
$Q4$	2.88	3.63
$Q5$	3.28	3.75
$Q6$	2.84	3.63

Table 6: Problems in Experiment 1

	problem
$Q1$	$A \to \neg\neg A$
$Q2$	$A \to B \vdash \neg B \to \neg A$
$Q4$	$\forall x(P(x) \to Q(x)) \to (\forall x P(x) \to \forall x Q(x))$

The time for subjects to familiarise themselves with LASP is one hour in Experiment 2 because twenty minutes was not enough in Experiment 1. There are four exercises in all, the same as in Experiment 1. The order of exercises differs by group. The way of measuring the solving time is the same as in Experiment 1.

4.4 Results of the Experiments

4.4.1 Results of the Questionnaires

Table 5 shows the average scores of the questionnaire items for Experiments 1 and 2.

The following feedback was obtained from the free comment space of Experiment 1.

- It would be better if the nested logical expressions were shown clearly by using a colour-coded fonts.
- It is troublesome that users have to use a mouse.
- Users want to adjust the window size freely.
- Panel alignment should be improved.
- Shortcut keys should be provided.
- A hint feature should be provided.
- It should show the answer.
- Users seem to be able to become familiarised with it, given enough time.

The following feedback was obtained from the free comment space of Experiment 2.

- LASP should notify the user clearly when the proof is correctly constructed.
- It is troublesome for users to write expressions including quantifiers because users have to input "forall" to display ∀.
- Users are not allowed to delete logical expressions except the latest one.
- Users are not allowed to add logical expressions except at the end of their list.
- It would be better if there was a memo panel for planning proof tactics.
- It would be better if there were a feature for users to make their own exercises.

Table 7: Number of subjects who correctly solved an exercise in Experiment 1 (/total number of subjects)

	handwriting	LASP
$Q1$	5/23	7/23
$Q2$	6/24	7/23
$Q4$	0/27	1/22

Table 8: Average solving time in Experiment 1

	handwriting	LASP
$Q1$	9m13s	10m04s
$Q2$	6m56s	8m15s
$Q4$	No one solves	8m18s

- Shortcut keys should be implemented.
- Panel alignment should be improved.
- It is hard to see the nested expressions.
- Hints should be improved.

From the comments, we conclude that the following are the advantages of LASP relative to doing proofs by hand.

- Users can reduce their amount of effort.
- Users can reduce the number of careless mistakes they make.
- It is easier to solve exercises because users can try many tactics within a short time.
- Amount of time which users can use to learn by making mistakes is increased because less time and effort is required to make substitutions and do inference.

4.4.2 Results for Solving Time

Table 6 shows the problems used in Experiment 1. Every problem assumes that theorem $P \to P$ has already been proved.

Tables 7 and 8 show the results of Experiment 1. We omit the results of Exercise 3 because it involves a mistake. Table 7 shows the number of subjects who solve exercises correctly (S_c) and all subjects (S_{all}). Table 8 shows the average solving time of subjects who correctly solved the exercise.

Table 9 shows the problems used in Experiment 2. Every problem assumes that the theorem $P \to P$ has already been proved.

Tables 10 and 11 show the results of Experiment 2. Table 10 shows S_c/S_{all} and Table 11 shows the average solving time for subjects who correctly solved the exercise.

5 Discussion

First, we consider the usability which is evaluated in Experiment 1. We can see that users are unsatisfied with LASP's usability from the results of $Q1, Q2$, and $Q3$ in Experiment 1, shown in Table 5. This is also shown by the results for $Q6$ in Experiment 2. Therefore, LASP's user interface should be improved. In order to improve the interface, we have to consider the free comments. We think that implementing shortcut

Table 9: Problems in Experiment 2

	problem
$Q1$	$\neg A \to B \vdash \neg B \to A$
$Q2$	$\neg X \to (X \to \neg Y)$
$Q3$	$(\neg Y \to \neg X) \to ((\neg Y \to X) \to Y)$
$Q4$	$\forall x(A \to B(x)) \to (A \to \forall x B(x))$

Table 10: Number of subjects who correctly solved an exercise in Experiment 2 (/total number of subjects)

	handwriting	LASP
$Q1$	5/8	2/8
$Q2$	4/8	4/8
$Q3$	4/8	2/8
$Q4$	2/8	2/8

keys and a feature which shows the users that they have finished correctly proving a proof are easy tasks.

Second, we consider the efficiency of proving, which was mainly evaluated by Experiment 2. From the results of $Q1, Q2$, and $Q3$ shown in Table 5, we can see that the substitution assistant feature and the inference assistant feature have contributed sufficiently to the efficiency of proving. Users also feel that. In addition, twelve out of sixteen subjects in Experiment 2 comment that LASP can reduce the required effort relative to doing problems by hand. Therefore, the goal of LASP's development seems to have been sufficiently achieved. The reason that the average score for $Q3$ is less than those of $Q1$ and $Q2$ seems to come from dissatisfaction with LASP's usability. Therefore, we should implement shortcut keys and other features.

From the results of Tables 8 and 11, we can see that whereas proving by LASP is slower than by hand in Experiment 1, the opposite result was obtained in Experiment 2. The reason for this seems to be that, while the time during which subjects could familiarise themselves with LASP was twenty minutes in Experiment 1, it was an hour in Experiment 2. Therefore, we conclude that if users are familiar with LASP, then they can solve practice problems more effectively.

The sample size of Experiment 2 was insufficient; we should perform experiments on a larger scale to prove the correctness of this hypothesis.

From the results shown in Tables 7 and 10, we can see that the number of subjects who can correctly solve exercises using LASP is the same as the number who can correctly solve exercises by hand. Thus the results suggest that the increased efficiency may not influence the correctness of answering questions. We cannot simply say that LASP helps students to easily solve exercises on logic, because fewer subjects could solve exercises Q1 and Q3 using LASP than by hand. However, in the handwritten answers, there were a mistake in applying inference rules and also two substitution mistakes. Using LASP, users can avoid careless mistakes because it shows an error message when users wrongly apply inference rules.

Table 11: Average solving time Experiment 2

	handwriting	LASP
$Q1$	10m35s	5m57s
$Q2$	7m19s	6m04s
$Q3$	8m48s	5m54s
$Q4$	12m56s	5m19s

In general, a learner needs trial and error to obtain a correct answer. LASP can store every step (trial) of a proof that the learner produces. From such a set of steps of a proof, the learner can select the essential sequence of the proof. For a simple proof task, the learner easily finds the essential sequence of the proof; however, it is difficult for users in the case of a complex proof task. Therefore, a feature giving assistance in the rewriting of the proof is being considered. Such a feature can be implemented by an automatic choice of directly relating steps of a focusing step. We want to add such a feature to LASP.

In conclusion, LASP is effective for learning formal proofs.

6 Conclusion

This paper describes our learning assistant system for formal proof, LASP. LASP has features such as inputting exercise data, a substitution assistant, an inference rule assistant, and so on. From two experiments, we obtained results implying that LASP is effective, although the user interface should be improved. Future work includes the improvement of the user interface and actual application in a class.

We strongly believe that the hint feature is useful for a learner to reach the correct answer. However, the design of hints requires as much care as the design of exercises because too much information can let a learner obtain the correct answer too easily; on the other hand, a little information seldom helps a learner obtain the correct answer. Therefore, the teacher has to construct hints manually. Of course, the system can help the teacher to design hints. To develop such an assistance feature for designing hints and to evaluate the power of the hint feature is part of our future work.

Acknowledgments

This work is being conducted as a part of Stage Project, the Development of Next Generation IT Infrastructure, supported by the Ministry of Education, Culture, Sports, Science and Technology, Japan, as well as a Grant-in-Aid for Scientific Research C (21500036). We also thank all subjects who participated in the experiments.

REFERENCES

[1] D. Abraham, L. Crawford and et al, “A tool to practice formal proofs,” Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications, pp.7–8 (2001).

[2] C. Baier and J. Katoen, “Principles of model checking,” MIT Press (2008).

[3] D. Barker-Plummer, J. Barwise, and J. Etchemendy in collaboration with A. Liu, “Tarski's World: Revised

and Expanded,” Center for the Study of Language and Inf (2007).
[4] J. Barwise and J. Etchemendy, “Turing’s World 3.0: An Introduction to Computability Theory,” University of Chicago Press (1993).
[5] J. Barwise and J. Etchemendy, “Computers, visualization, and the nature of reasoning,” The digital phoenix: How computers are changing philosophy, pp.93–116 (1998).
[6] Y. Bertot and P. Castéran, “Interactive theorem proving and program development: Coq’Art: the calculus of inductive constructions,” Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag (2004).
[7] A. Bove, P. Dybjer and U. Norell, “A brief overview of agda — a functional language with dependent typess,” In Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics (TPHOLs ’09), pp.73–78, (2009).
[8] B. Buchberger, A. Crăciun and et al, “Theorema: Towards computer-aided mathematical theory exploration,” Journal of Applied Logic, Vol.4, No.4, pp.470–504 (2006).
[9] R. Dyckho, “MacLogic: A Proof Assistant for First Order Logic on the Macintosh,” Computational Science Division, University of St. Andrews (1989).
[10] D. Gillet, A. Vu Nguyen Ngoc and Y. Rekik, “Collaborative web-based experimentation in flexible engineering education,” IEEE Transactions on Education, Vol.48, No.4, pp.696–704 (2005).
[11] P. Henderson, “Mathematical reasoning in software engineering education,” Communications of the ACM, Vol.46, No.9, pp.45–50 (2003).
[12] H. Kaufmann and D. Schmalstieg, “Mathematics and geometry education with collaborative augmented reality,” Journal of Computer and Graphics, Vol.27, No.3, pp.339–345 (2007).
[13] K. Miyazawa, “LASP” (2010). `http://sdl.ist.osaka-u.ac.jp/~k-miyazw/`.
[14] C. Shubert, I. Ceraj and J. Riley, “Bringing research tools into the classroom,” Journal of Computers in Mathematics and Science Teaching, Vol.28, No.4, pp.405–421 (2009).

(Received February 27, 2012)
(Revised March 1, 2013)

Kiyoyuki Miyazama received the BE and MI degrees in computer science from Osaka University in 2009 and 2011, respectively. His research interests include model driven architecture and education of formal approach.

Kozo Okano received the BE, ME, and PhD degrees in information and computer sciences from Osaka University in 1990, 1992, and 1995, respectively. Since 2002 he has been an associate professor in the Graduate School of Information Science and Technology, Osaka University. In 2002, he was a visiting researcher of the Department of Computer Science, University of Kent at Canterbury. In 2003, he was a visiting lecturer at the School of Computer Science, University of Birmingham. His current research interests include formal methods for software and information system design. He is a member of IEEE, IEICE of Japan, and IPS of Japan.

Shinji Kusumoto received the BE, ME, and DE degrees in information and computer sciences from Osaka University in 1988, 1990, and 1993, respectively. He is currently a professor in the Graduate School of Information Science and Technology at Osaka University. His research interests include software metrics and software quality assurance techniques. He is a member of the IEEE, the IEEE Computer Society, IPSJ, IEICE, and JFPUG.