フォーマルアプローチの基本技術習得のための学習支援システムの試作

宮澤清介[†] 岡野浩三[†] 楠本真二[†]

ソフトウェア設計開発においてフォーマルアプローチが近年注目を浴びている．この技術の基本は数理論理学であり，とりわけ形式証明の概念を理解することは，フォーマルアプローチの基本技術の理解に対して重要である．形式証明の学習は形式体系の重要な概念であるが，厳密な操作を要求されるため初学者に対する敷居は高いと思われる．そこで，学部生の論理学の授業において形式証明の学習を支援するツールの試作と簡単な評価実験行った．ツールの設計と実装にあたり，通常の学習者がとる手書きによる問題演習の欠点を改善する目標を定めた．具体的には，以下の 3 点である．1. 長い論理式に対する入力支援 2. 証明系列の作成時のコピー＆ペーストに相当する操作を要求しない．3. 証明結果の LaTeX 出力．実験評価の結果，ツールを使用しない被験者にはその問題演習の欠点が現れたが，ツールを使用した場合ではそれが改善された．

A Prototype of Learning Support System for Formal Proof

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

Recently, formal approach has been obtaining a lot of attention in software development. The basis of this technology is mathematical logic. Especially, it is important to understand the concept of the formal proof, because it is closely related to the basic technique of the formal approach. However, it is difficult for the novices to study formal proof because of rigorous operations. To resolve such a problem, the authors propose and implement a prototype of a learning support tool for formal proof in the class of Logic. The purpose of the proposed system is to prevent the drawbacks which occur when normal learners exercise by the hand. The authors also propose design and implementation of the tool. The advantages of the proposed method are the following: 1. Input support for long logical expressions; 2. the users are not required to operate copy and paste when they make proofs; and 3. the proofs are able to output in LaTeX files. Experimental results show that the tool improves the drawbacks of conventional exercises.

1. まえがき

フォーマルアプローチに基づいた情報システム開発はミッションクリティカルなシステム開発などを対象として，MDA（Model Driven Architecture）のコア技術として利用されている．フォーマルアプローチ手法として大きく， Coq[1)]，Agda，Isabelle/HOL などを用いた対話型定理証明とモデル検査手法[2)] の 2 つのアプローチがある．前者は論理学の形式的証明が必須であり， 後者は論理式の充足不能性判定の高速化が重要な技術となっている．

一方，文献[3)] で “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 behaviors and for analyzing system features.” とあるように，ソフトウェア工学において数理論理学の理解は重要であり，その学習を支援し，フォーマルアプローチの基本技術の習得を促進させることで，ソフトウェアの生産性を高めることができる．その他，数理論理学は Computing Curricula 2001[4)] で CS 分野のコアカリキュラムの一部として設定されている．日本でも情報処理学会の標準カリキュラム J07 で CS 分野の基礎科目として位置づけられている．本研究で扱う教育支援システムは公理に基づいた形式証明の教育を支援するものである．

関連研究としては以下が挙げられる．スタンフォード大学の Jon Barwise らは学生のモデル理解を促進させるため，チューリング機械をグラフィカルに学習するシステムである Turing’s World や，数理論理学の意味論をグラフィカルに学習するシステムである Tarski’s World[5)] などを開発した[6)]．

Tarski’s World は，3D 空間に大きさや形状を自由

† 大阪大学 大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University

に変更できるブロックを配置することにより，その空間に対して記述された論理式の真偽を解答する問題や，その空間の意味を論理式を用いて記述する問題など，様々な形式の問題を出題できる．また，Gentzen 流の自然演繹[7] の学習支援システムとして，MacLogic[8] というソフトウェアがある．

本ツールは論理学の初学者を対象にしているため，対話型定理証明器の直接の使用は考えていない．逆に論理学の初学者を対象にした機能構成を考えた．また，関連研究で紹介したツールは意味理解に重点をおいているものが多いが，本ツールは形式証明に重点をおいている．また，授業では Hilbert の公理系を中心に扱っており，異なる公理系を演習で用いるのは上級学習者にとっては有益であるが，初学者にはかえって混乱を招く点で不適切と考えた．

著者らが調査した限りでは，Hilbrert 流の形式証明を学習するためのツールは見つからなかった．また，初学者に対して評価実験を行い，一定の評価を得ることができたと考える．

以降 2 節では形式的証明について簡単に述べ，3 節では本研究で試作したシステムについて触れる．4 節ではシステムの評価実験について述べ，5 節で考察を，最後に 6 節でまとめる．

2. 形式的証明

公理とは，その他の命題を矛盾無く導出するための前提である．そして推論規則とは，証明済みの論理式から新たな論理式を導出するための規則である．例えば，前件肯定（modus ponens）と呼ばれる推論規則は $P \to Q$ と P から Q を導くことができるという規則である（P と Q は任意の論理式）．公理と推論規則をまとめて，公理系と呼ぶ．また，公理系から導かれる論理式を定理と呼ぶ．

公理から推論規則を用いて定理を証明していく過程を形式的証明と呼ぶ．図 1 に $\vdash X \to X$ という定理を導く形式的証明の例を挙げる．

公理系
(1) $\vdash (P \to (Q \to P))$
(2) $\vdash ((P \to (Q \to R)) \to ((P \to Q) \to (P \to R)))$
(3) $\vdash ((\neg P \to \neg Q) \to ((\neg P \to Q) \to P))$

推論規則
(1) P と $P \to Q$ から Q を得る.

証明系列
$\vdash ((X \to ((X \to X) \to X)) \to ((X \to (X \to X)) \to (X \to X)))$
（公理2 に $P = X$, $Q = (X \to X)$, $R = X$ を代入） (1)
$\vdash (X \to ((X \to X) \to X))$
（公理1 に $P = X$, $Q = (X \to X)$ を代入） (2)
$\vdash (X \to (X \to X))$
（公理1 に $P = X$, $Q = X$ を代入） (3)
$\vdash ((X \to (X \to X)) \to (X \to X))$
（証明系列（1），(2) と推論規則1 より） (4)
$\vdash (X \to X)$
（証明系列（3），(4) と推論規則1 より） (5)

図 1 証明系列例
Fig. 1 A proof example

3. 一階述語論理の形式証明学習支援システム

本研究で作成したシステムである “Learning Assist System for Proof”（以降 LASP）について述べる．

3.1 システム概要

本研究で試作した学習支援システム LASP は，大阪大学基礎工学部情報科学科の数理論理学の授業である,「情報論理学」で使用されることを目的としている．「情報論理学」では，Hilbert の公理系[9] に基づく形式的証明を扱っており，1 節で挙げた Tarski’s World は意味論を扱っている点で，MacLogic は Gentzen 流の自然演繹を扱っている点で，提案するツールとは異なっている．

手書きによる証明の問題演習の欠点として，以下のようなことが挙げられる．

(1) 長い論理式を手書きにすると，構文対応や変数名の書き損じを起こす
(2) 証明系列の作成にあたり，コピー＆ペーストに相当する操作をすることが頻繁に起こる

したがって，以上の点を改善するため，本ツールでは以下のような目標を定めた．

(1) 論理式の入力を最小限にとどめる入力インタフェースを提供
(2) コピー＆ペーストに相当する操作を要求しない

以下の節で，それらの具体的な機能やインタフェースについて述べる．

3.2 問題データ読み込み

本機能の目的は，問題を解答するにあたって，不要な手間を削減することにある．

本機能は，ユーザが問題データを記述したファイル選択すると，その問題における公理，推論規則，仮定，推論すべき定理のデータをシステムに入力する．

本機能で使用する問題データは xml 形式で記述する．xml のタグは表 1 のように定義した． また，xml 形式で作成した問題データの例を図 2 で示す．

3.3 代 入 支 援

LASP の代入支援機能について述べる．

本機能の目的は，代入ミスによる証明時間の浪費を削減することにある．なぜなら，代入ミスは本来の証明における学習とは無関係だからである．

例を挙げて説明する．以下の公理

$$(P \to (Q \to R)) \to ((P \to Q) \to (P \to R)) \quad (1)$$

の P, Q, R にそれぞれ，$X \to Y, Z, ((Y \to Z) \to X)$

表 1　XML タグの定義
Table 1　Definition of the XML tag

root	ルート要素
axioms	axiom 要素の集合
axiom	公理を表す要素
inference_rules	inference_rule 要素の集合
inference_rule	推論規則を表す要素
proof	1 つの question 要素と， 0 個以上の hypothesis 要素からなる集合
question	推論すべき定理を表す要素
hypothesis	仮定を表す要素

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

図 2　xml ファイルの例
Fig. 2　An example of the xml file

を代入すると，

$$((X \to Y) \to (Z \to ((Y \to Z) \to X))) \to (((X \to Y) \to Z) \to ((X \to Y) \to ((Y \to Z) \to X))) \quad (2)$$

論理式（2）のように複雑な論理式になる．もし代入ミスをしていた場合，どの部分を間違えたのかを探していると，非常に手間がかかってしまう．前述の通り，このプロセスは命題証明における学習とは無関係である．また，手書きで論理式（2）のような長くて複雑な式を書くのは非常に時間と手間がかかる．自動で代入を行うことで，無駄な時間と手間を削減する効果も期待できる．

本機能の概要を述べる．本機能は，ユーザが任意の公理を選択し，その公理の各命題変数に対して，任意の論理式を入力すると，各命題変数を各論理式に置換して，新しい論理式を生成する機能である．

例えば，ユーザが（3）で表す公理を選択し，

$$(P \to (Q \to P)) \quad (3)$$

この公理 3 の P に $(X \to Y)$ を，Q に Z を代入する，ということを指定すると，論理式（4）を自動で生成する．

$$((X \to Y) \to (Z \to (X \to Y))) \quad (4)$$

また，一階述語論理における変数への代入に関しては，代入対象の変数が自由変数であるか，そして代入する項が元の論理式の変数に対して自由かどうかを考慮しなければならない．例えば以下の論理式（5）を考える．

$$\forall x \exists y f(y, z) \quad (5)$$

論理式（5）の y は $\exists y$ に束縛されているため，代入することはできない．また z に，x または y を含んだ項を代入すると，その結果が量化記号に束縛されてしまう．例えば z に $g(x)$ を代入すると論理式（6）になる．

$$\forall x \exists y f(y, g(x)) \quad (6)$$

論理式（6）中の変数 x は，$\forall x$ に束縛されているため，論理式（5）とは異なる意味になってしまっている．本ツールでは，各変数に代入禁止変数リストを設けることによって，これらの不正な代入が行われそうになると，例外処理を発生するようにした．

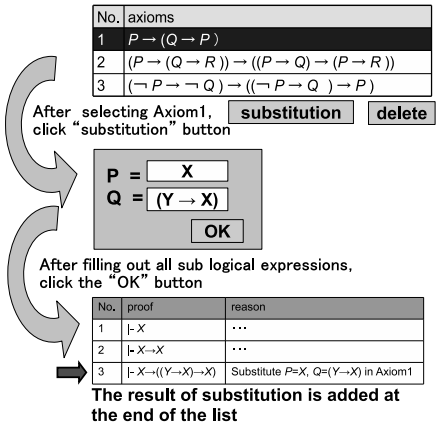

図 3　代入の流れ
Fig. 3　term substitution flow

実際のソフトウェア上における，ユーザへの機能の提供方法について述べる．公理は，図 3 の上のように，テーブルを用いて管理する．

ユーザはテーブルから，代入を行いたい公理をクリックして選択する．選択状態にした後,「代入」ボタンをクリックする．すると，図 3 の中にあるような代入パネルが開くので，各命題変数に，代入したい論理式を入力し，完了ボタンを押す．

完了ボタンを押すと，証明系列のテーブルの最後に，代入した結果が追加される（図 3）．

3.4　推 論 支 援

証明系列に推論規則を適用する機能について述べる．

本機能の目的は，手書きで証明をする上で，頻繁に起こりうるコピー＆ペーストに相当する操作を削減することにある．なぜなら，推論規則や演繹定理の概念

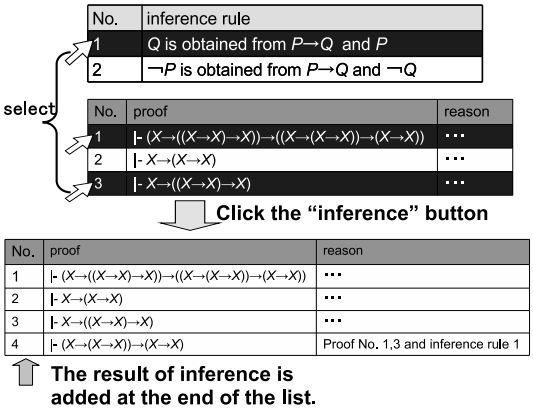

図 4 推論の流れ
Fig. 4 Flow of constructing a proof with the system

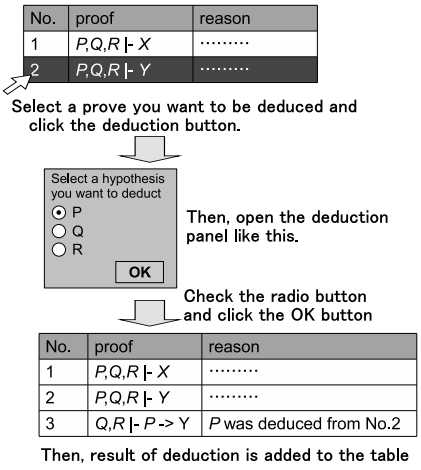

図 5 演繹定理の過程
Fig. 5 Applying Deductive theorem with the system

を理解している者が手書きで演習するにあたって，コピー＆ペーストの操作はただ時間を浪費するだけだからである．

また，推論の自動化による証明の効率化のため，推論ミスを無くすために，推論規則を適用しようとしている論理式が，本当に適用できる形であるのかをチェックし，正しい場合にのみ推論結果と，生成理由を出力する．生成理由とは，生成された論理式が，どの論理式と推論規則から生成されたのかを示すものである．また，推論を自動で行うことによって，冗長で複雑な論理式と生成理由を書く手間が省ける．

本機能の概要について述べる．本機能ではユーザが，適用したい推論規則と，その推論規則に適合する証明済みの論理式を選択すると，規則に対応する新しい論理式を自動で生成する．また，自動で生成した推論結果に，その推論理由を付加することで，手書きで証明をする際の生成理由を付加する手間も削減する．

実際のソフトウェア上における，ユーザへの機能の提供方法について述べる．推論規則と証明系列は，それぞれテーブルで管理している．ユーザはまず，適用したい推論規則を，推論規則のテーブルからクリックして選択する．次に，選択した推論規則に適合する証明系列をクリックして選択し，導出ボタンを押すと，正しく証明系列を選択できていた場合に対し，推論規則に基づいて，新たな論理式を生成する．システム上での実行は，図 4 のようになる．

3.5 演 繹 定 理

演繹定理は，形式証明において必要不可欠な定理である．ユーザが実行する過程は図 5 のようになる．

3.6 LaTeX 清書

本機能の目的は，レポートや答案を作成する手間を省き，証明結果を目視で用意に確認可能にすることで

図 6 LASP
Fig. 6 Overview of LASP

表 2 LOC (Lines Of Code) の内訳
Table 2 details of LOC

論理式のパーサ	約 1300 行	GUI	約 1000 行
内部データ処理	約 1000 行	代入支援	約 60 行
問題読み込み	約 70 行	推論支援	約 250 行
LaTeX 出力	約 100 行	演繹定理	約 30 行

ある．3.3 節でも述べたが，証明問題を解く上で，証明系列を生成することは，非常に文章量が多く手間がかかる．3.3 節や 3.4 節で導入した学習支援機能を利用して証明した結果を，自動でファイルに出力でき，目視で容易に確認できることは，非常に学習に効果があると考える．この機能により生成された tex ファイルをコンパイルすると，図 1 の出力が得られる．

3.7 LASP の開発

LASP の外観を図 6 に示す．

LASP の開発には Java を用いた．総クラス数は 17，規模は約 4000 行になった．その内訳を表 2 で示す．

4. 評 価 実 験

4.1 目　　的

本実験の目的は，第一に，現状における学習支援システムの機能がユーザにとってどの程度有用であるかを計測すること．第二に，本システムをさらに有用なものにするためには，どのような機能を実装するべきかという意見を集めるためである．

4.2 評 価 項 目

評価項目は大きく 2 種類である．第一に，システムの使い易さの評価，第二に，システムを利用した場合と手書きで証明した場合との証明時間の差である．システムの使い易さの評価として，文献[10] などを参考にして以下のような項目を尋ねるアンケートを作成した．

Q1　代入支援機能はどの程度使いやすいか
Q2　推論支援機能はどの程度使いやすいか
Q3　手書きと比較して，証明は楽になったか
Q4　他の同様な問題が解けるようになったと思うか
Q5　LASP に，他にどのような機能があれば便利だと思うか

回答は 5 段階評価で，数字が大きいほど評価が高い．低評価（1 点または 2 点）を付けた被験者には，その具体的な理由を記入していただいた．

4.3 方　　法

評価実験を行った手順は，以下の通りである．

(1)　命題論理についてのテキストを被験者に配布
(2)　30 分間，被験者に勉強をしてもらう
(3)　LASP の使い方を被験者に説明
(4)　問題を手書きで 2 問，LASP を利用して 2 問，交互に解いてもらう
(5)　被験者にアンケートの回答を行ってもらう

テキストは，文献[11] の pp110-120 を用いた．

被験者が実験中に学習してしまうことを考慮して，問題はランダムに割り当て，手書きと LASP 利用を交互に行った．解答開始は被験者全員が一斉に行い，証明が完了したら被験者に手を挙げてもらい，試験者が解答時間を記録した．

4.4 環　　境

ここでは，本実験における被験者について記述する．被験者は所属研究室の学生 12 人（情報科学研究科博士後期課程 1 年 1 人，博士前期課程 2 年 5 人，1 年 3 人，情報科学科 4 年 3 人）である．

4.5 出題した問題

出題した問題は以下の通りである．

公理 1　$\vdash (P \to (Q \to P))$
公理 2　$\vdash ((P \to (Q \to R)) \to ((P \to Q) \to (P \to R)))$
公理 3　$\vdash ((P \to Q) \to ((P \to \neg(Q)) \to \neg(P)))$
推論規則 1　P と $P \to Q$ から Q を得る．

以上の公理系において，以下を証明せよ．

問題 1　$X \vdash \neg\neg X$
問題 2　$(X \to Y), \neg Y \vdash \neg X$
問題 3　$\neg(X \to Y), \neg X \vdash \neg Y$
問題 4　$X \to Y, W \to Z, \neg X \to W, \neg Y \vdash Z$

表 3　アンケート結果
Table 3　Result of questionnaire

被験者		Q1		Q2		Q3		Q4	
1	7	2	4	4	5	4	5	3	3
2	8	2	5	2	5	4	5	2	3
3	9	4	4	4	2	4	4	3	3
4	10	4	4	4	4	5	3	3	3
5	11	3	4	3	4	5	5	4	5
6	12	4	5	4	4	3	5	1	3
平均点		3.75		3.75		4.33		3.00	

4.6 結　　果

4.6.1 アンケート結果

本実験で得られたアンケート結果を，表 3 に示す．Q1 ∼ Q4 は，4.2 節のアンケート項目 Q1 ∼ Q4 に対応している．また，低評価を付けた被験者の意見として，以下が挙げられた．

- フォントの色分けなどで，論理式のネストを明確にできれば良い
- マウスカーソルを毎回合わせることが面倒である
- マウス移動が大きく，面倒である
- ウインドウのサイズを自由に調節したい
- 公理，推論，証明系列などのパネルの配置を考えるべき
- ショートカットキーがあれば良い
- 入力エラーのメッセージを具体的に表示して欲しい
- ヒント機能があれば良い
- 解答を表示してくれる機能が欲しい

4.6.2 解答時間の測定結果

手書きで証明した場合と，LASP を利用して証明した場合の証明完了までの時間を測定した結果を表 4 に示す．解答時間は，証明が終了すると同時に被験者に手を挙げてもらうことで記録した．表 4 の解答時間は，制限時間内に解けなかった場合と，不正解の場合の解答時間を除外して平均値を計算した．

この結果から，解答時間は手書きと比較しても余り差が見られなかった．しかし，全ての解答を調査したところ，手書きで証明を行った解答から，2 件の代入

表 4 手書きの場合と LASP を使用した場合の解答時間の比較結果 (問題を解けた被験者の数)
Table 4 Result of answer time of comparing handwriting with using the LASP

	手書き	LASP
問題 1	14 分 11 秒 (1)	11 分 00 秒 (1)
問題 2	4 分 57 秒 (5)	5 分 41 秒 (4)
問題 3	5 分 13 秒 (1)	3 分 56 秒 (3)
問題 4	7 分 32 秒 (3)	7 分 40 秒 (4)
平均	7 分 58 秒	7 分 04 秒

ミスが見つかった.

5. 考　　察

まず，表 3 の Q3 に着目すると，全回答の平均点が 4.33 点であることから，LASP を利用した方が，ユーザにとって証明が楽になることが言える．また，表 4 の結果を見ると，実際に証明にかかる時間も短くなっているので，証明を効率化する機能は十分に有用であることを示している．

また，Q1 と Q2 に着目すると，平均点は 3.75 点であることから，多数のユーザを満足させられるほどの高評価は得ることができなかったが，代入支援や推論支援機能は，どちらかと言えば使い勝手が良く，便利であるということがわかった．アンケートでこれらの機能について低評価を付けた被験者から得られた意見を参考にすると，まだユーザ・インタフェースにおいて改良の余地があるということであった．

これらの意見を考慮して，ユーザの立場でインタフェースのユーザビリティを改善することで代入支援と推論支援機能の質が向上すると考えられる．

さらにアンケートの自由記述欄の意見である，ヒントや解答の表示機能，加えて，解答者のレベルに合わせて出題する機能やゲーム機能といったものを実装すれば，より学習効果の向上が期待されるため，積極的に実装すべきである．

表 4 の結果を見ると，問 2 のみ LASP の方が遅いという結果になっている．これは被験者の一人がほぼ 15 分かけて解答したためで，母数が少数であるため，外れ値により大きく平均値が変化してしまったと考えられる．この被験者は手書きの問題を 2 問とも制限時間内に解答することができず，LASP を使用して片方の問題のみ解答できた．したがって，問 2 の平均解答時間は大きくなっているものの，LASP によるデメリットとは考えにくい．問 2 以外の問題や，平均的な解答時間に大差は見られなかったが，手書きの解答から 2 件の代入ミスが見つかったため，本研究の目的は達成できたと言える．

6. あとがき

本研究では，手書きによる問題演習の欠点を改善するために，問題データ読み込み，代入支援，推論支援などの機能を実装した形式証明の学習支援システムを試作した．今後の課題として，ユーザ・インタフェースの改良，及び，学習者が理解できないところを明確にし，理解を促すような機能の実装が挙げられる．

謝辞　本研究は一部，科学研究費補助金基盤 C (21500036) と文部科学省「次世代 IT 基盤構築のための研究開発」(研究開発領域名：ソフトウェア構築状況の可視化技術の開発普及) の助成を得た．

参 考 文 献

1) Bertot, Y. and Castéran, P.: *Interactive theorem proving and program development: Coq'Art: the calculus of inductive constructions*, Texts in Theoretical Computer Science. An EATCS Series, Springer-Verlag (2004).
2) Baier, C. and Katoen, J.-P.: *Principles of model checking*, MIT Press (2008).
3) Henderson, P.: Mathematical reasoning in software engineering education, *Communications of the ACM*, Vol. 46, pp. 45–50 (2003).
4) Engel, G. and Roberts, E.: *Computing Curricula 2001 Computer Science*, IEEE Press (2001).
5) Dave Barker-Plummer, J. B. and in collaboration with Albert Liu, J. E.: *Tarski's World: Revised and Expanded*, Center for the Study of Language and Inf (2007).
6) Barwise, J. and Etchemendy, J.: Computers, visualization, and the nature of reasoning, *The digital phoenix: How computers are changing philosophy*, pp. 93–116 (1998).
7) Gentzen, G.: Investigations into logical deductions, 1935, *The Collected Papers of Gerhard Gentzen*, pp. 68–131 (1969).
8) Dyckho, R.: MacLogic: A Proof Assistant for First Order Logic on the Macintosh, *Computational Science Division, University of St. Andrews* (1989).
9) Hilbert, D.: *The foundations of geometry*, K. Paul, Trench, Trübner & co., ltd. (1902).
10) 吉野孝, 井上穣, 由井薗隆也, 宗森純, 伊藤士郎, 長澤庸二: インターネットを介したパーソナルコンピュータによる遠隔授業支援システムの開発と適用, 情報処理学会論文誌, Vol. 39, No. 10, pp. 2788–2801 (1998).
11) 小倉久和, 高濱撤行: 情報の論理数学入門 ブール代数から述語論理まで, 近代科学社 (1991).