2012.11.21

日本における Mizar プロジェクトの現状と展望

信州大学工学部情報工学科
師玉康成

TPP2012 (定理証明と定理証明系に関する研究集会)資料

1　Mizar の歴史と Mizar 言語

Mizar Project については, http://mizar.org/project/　に詳細な資料がある。　計算機によって形式化証明をチェックすることができるように,形式化数学記述言語とその処理系を開発する試みとして,Andrzej Trybulec 教授によって 1973 年頃に開始された。

その目標は,Mizar system (http://mizar.org/system/)の継続的な開発と並行して,現代数学のほとんどの核心部分をカバーする計算機のよって形式的に確認された証明の巨大なライブラリを構築することである。これは, Robert Boyer 教授らの QED プロジェクトの従った, その具体化と言える。現在,本プロジェクトは BiaLystok 大学(ポーランド),アルバータ大学(カナダ)および信州大学(日本)で研究グループによって展開, 維持されている。

その形式化記述の成果は,計算機のよるチェックと在来の学術論文誌と同様の人間に拠る複数・覆面査読制度の 2 重の審査を通過したものについて,

Formalized　Mathematics　http://versita.metapress.com/content/121073/

に掲載される。さらに WEB ページ

http://mizar.uwb.edu.pl/version/current/html/

に公開されている。

Mizar 言語の特徴はその読み易さにある。数学的な文書に共通した,論理(一階述語論理)と形式で記述されている。その記述は ASCII 文字列で書かれているが,ほとんどの数理・・情報科学の研究者が読むことができる数学的表現に十分に類似しており,これを読むのに特別な訓練が必要ないような記述形式を目指している。

しかしながら,そうは言っても,形式化証明には,計算機による証明チェックのために必要な非常な厳密さが要求される。その証明の全過程で,既にライブラリに収録されている定理の引用法も含め,詳細,かつ厳密な論理が要求される。当然ながら,Mizar による証明の形式化記述は通常の形式で書かれた同一内容を記述したものより数段長く,これを記述するには相当な努力と忍耐が必要である。その負担を軽減していくべきことは当然であるが,これは,計算機による厳密な証明チェックというメリットを得るための代償と言える。

2012 年 9 月時点で,この mizar のライブラリ Mizar Mathematical Library 以下略して MML は,200 名を超える著者によって書かれた 1165 のファイルがある。約 52,000 の定理の 10,000 を超える形式上の定義を含んでいる。JORDAN 閉曲線定理他,Brouwer 不動点定理,Godel の完全性定理などが収録されている。

その MML と処理系である **proof checker** は,

http://mizar.org/system/#distribution

から自由にダウンロードすることができる

MML は Tarski・Grothendieck 集合論の公理によって構築されている。
前述の WEB ページの
http://mizar.uwb.edu.pl/version/current/html/
には表示中の proof　という単語をクリックすれば，証明の全文が表示される。さらにそれで引用されるライブラリ中の既存の定理や用語にもリンクが張られ，これらのリンクを繰り返し辿れば，集合論の公理系にまで戻ることができる。

MML を利用するため検索エンジンは
http://mmlquery.mizar.org/mmlquery/three.html　　などがある。
また，Mizar システムの日本語解説書は,システム旧版に対応したものであるが,信州大学中村八束名誉教授の
http://markun.cs.shinshu-u.ac.jp/kiso/projects/proofchecker/mizar/Mizar4/index-j.html
がある。　その他，Mizar に関わる各種附属ソフト,Mizar システム利用者の Forum へ参加ページなどは　　http://mizar.org/project/　にある。

2　Mizar の概要とそれを用いたライブラリ作成

2.1　Mizar での形式化記述の実際

mizar とよばれる形式化数学記述言語は，第一階述語論理と Tarski の集合論をもとに，数学定理の証明を記号論理で記述する。mizar システムは，**proof checker** と呼ばれるもので，第一階述語論理と Tarski・Grothendieck の集合論の公理系をもとに，数学定理の証明を記号論理学の表現で記述し,それの論理的正しさを計算機によって検証する。
例えば，以下はその記述の例である。

形式化記述の実例

環境設定部分

```
environ
vocabularies NUMBERS, INT_1, ORDINAL1, SUBSET_1,  CARD_1,
  ARYTM_3, ARYTM_1, RELAT_1, NAT_1,XCMPLX_0,REAL_1;
notations  SUBSET_1,ORDINAL1, NUMBERS, XCMPLX_0,  INT_1, NAT_1;
constructors  REAL_1,NAT_1,  INT_1, CARD_1;
registrations  XREAL_0,NAT_1, INT_1, ORDINAL1, XBOOLE_0;
requirements REAL, NUMERALS, SUBSET,ARITHM, BOOLE;
definitions INT_1;
theorems INT_1;
schemes NAT_1;
```

本体部分

```
begin
theorem EX1:
for n being Nat holds 2 divides n*(n-1)
proof
defpred P[Nat] means 2 divides $1*($1-1);
0 = 2 * 0;
then
P0: P[0] by INT_1:def 3;
PN: for n being Nat st P[n] holds P[n+1]
proof
let n be Nat;
assume P[n];
then
consider s being Integer such that
A2: n * (n - 1) = 2 * s by INT_1:def 3;
 (n+1)*(n+1-1) = n*(n-1) + n + n*1
.=n*(n-1) + 2*n
.=2*s + 2*n by A2
.= 2*(s+n);
hence
P[n+1] by INT_1:def 3;
end;
thus for n being Nat holds P[n] from NAT_1:sch 2(P0,PN);
end;
```

これらは　大別すると environ 文から始まり，begin 文の前で終わる環境設定部分と begin 文以降の形式化証明本体の記述部分に分かれている。

環境設定部分は　C 言語プログラムのヘッダーや include(ライブラリ指定文)に相当し，Mizar ライブラリに既に収録されている用語や，定理，定義を begin 文以降の本体の形式化記述部分で用いるための設定を記述する。概略以下のようになっている

vocabularies	用語の定義ファイルを指定する
notations constructors	個々の概念の定義する 概念間の階層構造等の関係を表す。
registrations	変数型の階層的な関係, 数学的属性の従属関係などを指定する
requirements	実数の演算や集合の包含関係の演算の自動的な処理を指定する
definitions theorems schemes	begin 文以下の形式化記述の本体で引用される既存の定義や定理, 公理図式の収録ファイルを指定する。

上の形式化記述ではこれに定義が記述されていない 2 つの整数 i1,i2 に関する

述語 i1 devides i2

が用いられているがこれは以下のように INT_1:def 3 として既存のファイル INT_1 に収録されている。　前述の環境設定部分で, INT_1 の定義･定理群を引用するよう指定している。

```
definition
  let i1,i2 be Integer;
  pred i1 divides i2 means
:: INT_1:def 3
  ex i3 st i2 = i1 * i3;
  reflexivity;
end;
```

これは,　【　i1 と i2 を任意の整数とするとき,

述語 i1 devides i2 とは　　i3 が存在して，i2 = i1 * i3 が成り立つことを表す】

という定義である。

また，Mizar ライブラリには数学的帰納法の公理図式として複数のものがあるが

ここでは　NAT_1:sch 2 として収録されている以下を用いている

```
scheme :: NAT_1:sch 2
  NatInd { P[Nat] } : for k being Nat holds P[k]
provided
 P[0] and   for k be Nat st P[k] holds P[k + 1];
```

これは　【以下の条件を充たせば任意の自然数 k に対して P[k]が成り立つ。

条件　P[0]が成り立ち且つ　任意の自然数 k に対して以下が成り立つ
P[k]が成り立つならば，P[k+1]が成り立つ　】
という公理図式である。 以下，上記の形式化記述の begin 以下の記載についてその意味を対応表にして概説しておく。

theorem EX1:	命題名 EX1 という命題のラベル名の宣言
for n being Nat holds 2 divides n*(n-1)	「任意の自然数 n に対して，n*(n-1)は 2 で割り切れる。」　という命題 (注)　Nat は自然数の変数型名である。
proof	命題 EX1 の証明開始 (注)　以下 proof と end;で挟まれた　proof ~end ブロックが書かれ，上記命題 EX1 の証明が記述される。
defpred P[Nat] means 2 divides $1*($1-1);	自然数を引数に持つ述語 P[]で，引数($1)と$1-1 との積($1*($1-1))が，2 で割り切れることを表わす述語を定義(defpred)する。
0 = 2 * 0; then P0: P[0] by INT_1:def 3;	0 = 2 * 0; だから　2 divides 0*(0-1);　であり 補題名 P0:　P[0]　が　定義 INT_1:def 3 により成り立っている。
PN: for n being Nat st P[n] holds P[n+1]	補題 PN: 任意の自然数 n に対して「P[n]が成り立つとき P[n+1]が成り立つ」
proof	補題 PN の証明開始 proof　~end ブロックで記述する。
let n be Nat;	n を任意の自然数とする
assume P[n];	P[n] すなわち 2 divides n*(n-1) を仮定する
then consider s being Integer such that A2: n * (n - 1) = 2 * s by INT_1:def 3;	従って　INT_1:def 3 によれば以下の補題 A2 が成り立つある自然数 s を考えることができる。 補題 A2: n * (n - 1) = 2 * s
(n+1)*(n+1-1) = n*(n-1) + n + n*1 .=n*(n-1) + 2*n .=2*s + 2*n by A2	これを用いて　以下の等式が得られる。 (n+1)*(n+1-1) = n*(n-1) + n + n*1 .=n*(n-1) + 2*n

.= 2*(s+n);	.=2*s + 2*n by A2 .= 2*(s+n);
hence P[n+1] by INT_1:def 3;	よって P[n+1] が INT_1:def 3 によって成り立つ
end;	補題 PN の証明終了
thus for n being Nat holds P[n] from NAT_1:sch 2(P0,PN);	以上補題 P0,PN と数学的帰納法の公理図式 NAT_1:sch 2 を用いて, 任意の自然数 n について P[n]が成り立つ
end;	定理 EX1 の証明終了

以上，簡単に解説したが，このような解説をするまでもなく，begin 以下の英文に近い表現の記述を眺めれば， Nat が 自然数を表すものと知れば,『任意の自然数 n について，２は n*(n-1) を割り切る』 という，命題とそれの数学的帰納法による証明を記述していることが大凡，判読できるであろう。

プルーフチェッカーである mizar はこのように，自然言語に近い表現ができることが，特徴である。

2.2 Mizar ライブラリの現状

現在 1165 の，数学ライブラリが整備されており，そのシステムとライブラリは，インターネット上で以下の URL で全て公開されている。

http://mizar.uwb.edu.pl/version/current/html/

これらのライブラリは数学の公理体系から出発して必要な定理に至る全ての結果・証明が完全に収録され，しかも他を参照する必要のない自己完結的な形式化数学記述のライブラリとして提供されている。 **WEB 表示中の proof という単語をクリックすれば，証明の全文が表示される。さらにそれで引用されるライブラリ中の既存の定理や用語にもリンクが張られ，これらのリンクを繰り返し辿れば，集合論の公理系にまで戻ることができる。**

また，http://mmlquery.mizar.org/

を用いれば,著者別のライブラリ一覧や， 重要度の高いライブラリの表示その他の分類表示ができる。ここで,1165 を超えるライブラリを体系的に全て網羅して説明することは非常に困難であるので筆者らがライブラリ作成に関わった経験的な知識の範井で**以下のような概略表を作成した。 表中の*とあるのは** GROUP_1, GROUP_2, … のように同じファイル名でシリーズになっているもので，ファイル名先頭にシリーズ名の共通部分を持ち，残りの文字や数字が識別用に使われているものを*で表している。

TARSKI,BOOLE,XBOOLE_* ZFMISC_1	集合の包含関係,和,積,排他和などの演算, 直積,冪集合，恒等式・等式変形
CARD_*，ORDINAL*	順序数,集合の濃度，集合族の(添え字集合が非加算な場合を含む)多重直積
RELAT_* ,FUNCT_*，FUNCOP	２項関係，関数の定義,定義域,値域,合成,象,逆像，関数の集合
STRUCT_* ,ALGSTR_*,BINOP_*	基本的な数学的構造，代数構造 ２項演算
NUMBERS, ARYTM_*, REAL，XREAL_*,ABSVALUE, RINFSUP* SUPINF_*,XXREAL_* COMLEX1,XCMPLX*	自然数，整数，有理数，実数，複素数，拡張実数の定義やそれらの包含関係，実数の演算，上界,下界,上極限，下極限，恒等式,不等式，複素数の定義，絶対値，演算，等式変形
NAT_*, INT_* NTALOG_1 WSIERP_1	自然数，整数に関する基礎的な定義，定理, 素因数分解，剰余，素数，ユーグド互除法 拡張ユーグド互除法 中国人剰余定理
RFUNCT_*,FCONT_*, FDIFF_* FUNCSDOM RSSAPCE*	実数値連続関数，実数値関数の微分,中間値定理,平均値定理, 実数値関数の関数空間,実数列の空間,代数
SEQ_*　SERIES_*,COMSEQ_* SINCOS_*，TAYLOR_*, HEINE	実数数列・級数，複素数列・級数 ３角関数，De Moivre の定理,テイラー展開,Bolzano-Weierstrass の定理, Cauchy 列，Cauchy の定理 Heine-Borel の定理
FINSEQ_* RFINSEQ*, RVSUM_*	有限列の定義,性質,連結・分解・逆転・一部置換などの操作，多重対 (数ベクトルや多重対に関する記述等多くの用途がある) 実数有限列の和,差,総和,多重積などの操作

DESCIP_1 CIRCUIT*,ftacell1, gfacirc* PERTRI* morph_01	DES 暗号化アルゴリズム 回路,並列回路 ペトリネット **画像処理と数理形態学**
GOEDELCP	Goedel 完全性定理
GROUP_* GR_CY_*	**群の基本的な定理，正規部分群やそれによる商，群の位数，元の位数，同型定理，群の族の多重直積群，巡回群，Sylow 群など**
RLVECT_*,NORMSP_* VECTSP_* , PRVECT_*, BHSP_*, HAHNBAN	実ベクトル空間，実ノルム空間 一般体を係数とする線形空間 加法群，線形空間の多重直積空間 バナハ空間，ヒルベルト空間, Hahn-Banach の定理
VFUNCT* LOPBAN_*	ベクトル値関数，関数空間 有界線形作用素に関する基本定理 一様有界定理，開写像定理，閉グラフ定理
INTEGRAL_*　INTEGR_*	実数数値関数のリーマン積分の基本定理, Darboux の定理，ベクトル値関数のリーマン積分，微分方程式
JORDAN*	ジョルダン閉曲線定理とそれに関わる 連続曲線,位相
MEASURE* MESFUNC*, MESFUN* PROB_*,RANDOM_* DIST_*	測度論 ルベーグ積分，Egorov の定理，Fatou の補題,有界収束定理，Lp 空間 確率と確率変数 離散集合上の確率変数
NDIFF_*,PDIFF_*	実ベクトル空間，ノルム空間上の関数の微分，偏微分
RMOD_*,ZMOD_* ECPF_*	環上の Module,整数環上の Module, 楕円曲線
POLYNOM* POLYEQ_*	１変数，多変数の多項式と，多項式環, 代数学の基本定理 2,3 次の一変数代数方程式
AMI_*,SCM*	抽象計算機

METRIC_* EUCLID* TOPS_* TOPMETR TOPREAL* BROUWER*	距離空間 ユーグリッド空間 位相空間(位相構造，開集合，閉集合， compact 空間などの定義，定理群) 距離による位相空間 実数直線上や 2 次元平面上の位相空間 Brouwer の不動点定理
YELLOW_*	束論　連続束
WAYBEL_*	Directed Sets, Nets, Ideals, Filters, and Maps

2.3　現在の作業状況

筆者らは現在,
(1)学部や修士学生が学習する微積分や関数解析,代数等の基礎的定理のライブラリ整備
(2)暗号理論や情報工学分野での形式検証に必要なライブラリの整備
を当面の目標にライブラリ作成に取り組んでいる。(1)については,多岐に亘る。前述の表の YELLOW_*や WAYBEL_*などを除き,何らかの形で殆ど関わっている。特に The most important facts in MML
http://mmlquery.mizar.org/mmlquery/fillin.php?filledfilename=mml-facts.mqt&argument=number+102　に現れるものにも相当数貢献している。

(2)　は，　日本側から始めた活動である。
上の表中の

DIST_*	離散集合上の確率変数
DESCIP_1	DES 暗号化アルゴリズム
CIRCUIT*,ftacell1, gfacirc*	回路,並列回路
PERTRI*	ペトリネット
morph_01	画像処理と数理形態学
NTALOG_1	拡張ユーグリッド互除法
ZMOD_*　ECPF_*	整数環上の Module,楕円曲線
AMI_*,SCM*	抽象計算機

などがある。　これらについては，別途個別に説明するが，並列処理の回路や計算機の処理プログラムが正しく動作するかなどを検証するものである。

2.4 WEB 上の mizar システム

この Mizar グループの日本側の代表者は中村教授と筆者らである。筆者らは e-learning を主体とした社会人向けのインターネットによる遠隔教育(通称インターネット大学院)も運営している。 その教材には，mizar が重要な位置を占めているが，我々は WEB 上で学生も使用できる mizar システムを３つ併用している。

一つは日本の教員が作ったもので， これは以下の URL を開くことによって使用

http://www.wakasato.jp/mizar/

これは主に修士の学生の講義用に使用している。 使用法を若干説明すると，まず，自分で用語を新しく定義するか，また，特定のシステムコマンドを使用するかどうかによって Verifier のパッケージを選択し，次の入力画面では，使用する学生の ID や e-mail アドレスを入力する。 その後 Verifier のメインページが表示される。 それには，Mizar 言語で書いた証明を記入するテキストボクスや，ファイル名などを記入する入力ボクスがあり，これらに，データを入力した後 SUBMIT ボタンをマウスでクリックすると，Mizar の proof checker が働いて， 証明をチェックし誤りがあれば，エラーメッセージが表示される。

もう一つは Mizar の開発者の一人である Grzegorz Bancerek 教授が JSPS の客員研究として本学に招聘した際に作成したもので，ライブラリの高度な検索機能をもった

http://mmlquery.mizar.org/mmlquery/three.html

である。さらに，重要なツールとして ，これも Mizar Project の重要なメンバーである Josef Urban 教授による

http://mizar.cs.ualberta.ca/~mptp/MizAR.html がある。

これは既に述べた

http://mizar.uwb.edu.pl/version/current/html/

と同一形式の HTML ファイルを作成できるもので，E-learning 教材を開発する際には，他を参照する必要のない自己完結的な形式化数学記述のライブラリとして提供できる。Mizar の既存のライブラリと一体で数学の公理体系から出発して必要な定理に至る全ての結果・証明を完全に収録した教材が作成できる。次章で説明する。

3 Mizar を使った教材作成と教育

数理的な内容を，理解力や学力に差のある様々な受講学生に，疑問の余地なく，自明な事項から始め論理の飛びが一切なく理解させることは非常に難しい。相当に詳しく書かれた教科書を用いたとしても，受講者自身が「行間を読む」努力が必要であるし，教員がその学生の理解に応じて解説しなければならない。しかも，その理解を徹底させるために

は，学習させた内容に関して何らかの論証問題を解かせ，証明を実際に記述させてみることも必須であろう。　教員が学生個人に応じて教材を解説し，また提出された学生全員の解答について，添削し，完全な回答が得られるまで指導し続けることが，現実的に可能であろうか？　筆者らは，これらの課題に対する解決策として，　形式化数学記述言語 Mizar とその処理システムを援用した教材システムの開発研究を行なっている。　前述したように mizar ライブラリは

http://mizar.uwb.edu.pl/version/current/html/

に公開されている。これらのライブラリは数学の公理体系から出発して必要な定理に至る全ての結果・証明が完全に収録され，しかも他を参照する必要のない自己完結的な形式化数学記述のライブラリとして提供されている。　**WEB 表示中の proof　という単語をクリックすれば，証明の全文が表示される。さらにそれで引用されるライブラリ中の既存の定理や用語にもリンクが張られ，これを繰り返し辿れば，集合論の公理系にまで戻ることができる。**Mizar は TARSKI の集合論と第 1 階述語論理に基づいたシステムで，これで提供される形式化記述言語で，定理の形式化証明を記述し，**proof checker** と呼ばれる処理系でその論証の誤り等を，自動的に検出する。　筆者らは，そのライブラリと，プルーフチェッカーを汎用の CMS(Course Management System) システム Moodle に組み込み，教材システムとして利用する研究を行なっている。

3.1　CMS- Moodle-上のユーザインターフェース

以下筆者らが開発したシステムを紹介する。Moodle のシステムにログインした後(Fig. 3. 1)．画面中央のコース名をクリックするとその科目の学習ができ，各コース内にはその科目のテキストのほか，mizar の課題を行うことができる。

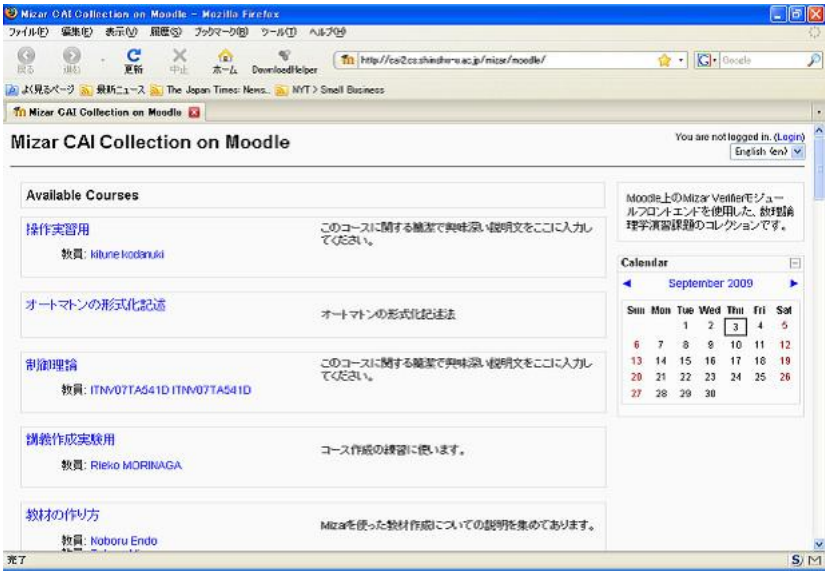

Fig. 3. 1　トップページ

Fig. 3.2. コース選択画面

Fig. 3.2 はmizar 課題演習のページの例である．教員はあらかじめ学習用コンテンツと演習課題を用意しておく必要がある．学生が解答を入力する解答画面を Fig. 3.3 に示す.

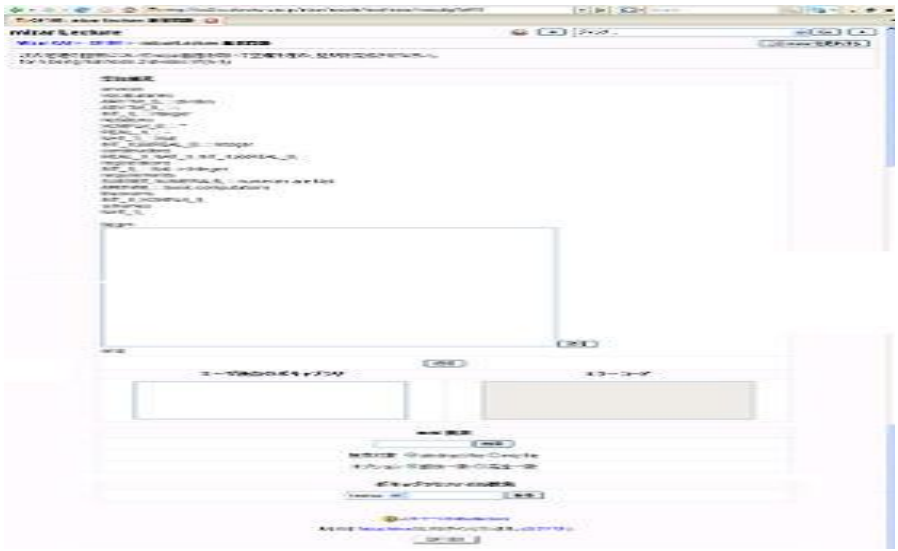

Fig. 3.3　解答用ページ

mizar の課題演習には複数行入力できる。テキストエリアを利用した穴埋め形式を採用している．これは，mizar で記述された証明文では，証明文の冒頭に環境部と呼ばれる証明中で用いられる単語の定義やすでに証明済みの定理を参照するためのライブラリ名を記述するが，学習者にとってはこれらのライブラリの検索に時間を取られ肝心の証明文の記述に労力を費やすことができないことを回避するためであり，環境部や証明の骨子部分を教師が用意し学習者は必要な個所のみ思考すれ

ばよいこととしている．教師が問題を作成する問題作成画面をFig.3.4 に示す．

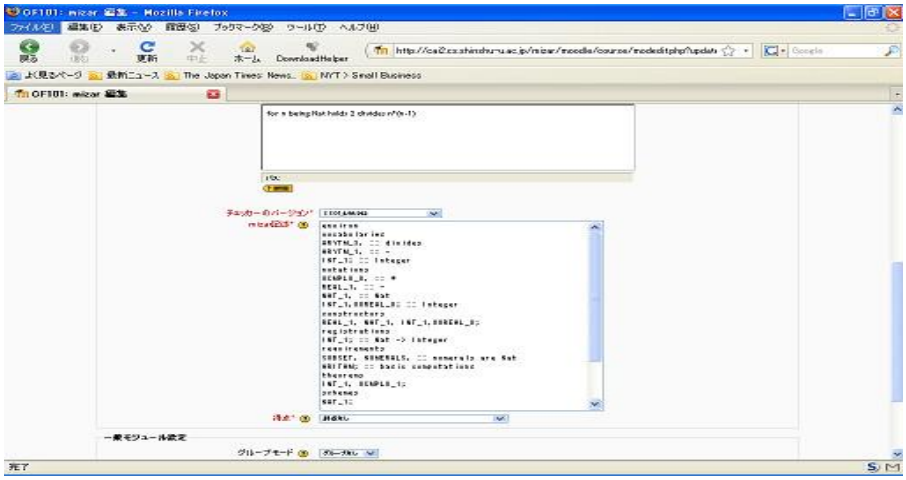

Fig.3.4　問題作成画面

問題作成時，教師は穴埋め問題に対応するために，空欄にする位置を指定する必要がある．Mizar では，その証明文の中で記号::をコメントとして扱うので，空欄の始まりを記号::<--で示し空欄の終わりを記号::-->で表わすこととした．こうすることで教師ユーザは問題用の証明文の任意の個所に空欄部分を作成することができ，一つの問題の中に複数の空欄を作成できるようになる．教師が問題用証明文を入力すると，Moodle のモジュールプログラム側は，問題用証明文中の::<--と::-->で囲まれた部分を一つの中抜き問題ブロックとして画面上に表示させる．このとき，空欄部分の作成場所として以下のようなルールを設け，教員が問題文を入力した際，上記をチェックし，入力箇所に誤りがあった場合は画面上に表示するようにし，問題作成画面に戻ってやり直しできるようにした．

（1）環境部（environ）よりも前に空欄を設けない
（2）証明の終わり（end:）よりも後に空欄を設けない
（3）空欄が全くない問題文を認めない
（4）::<--と::-->と個数を一致させる

またmizar のproof checkでは，proof checkerのバージョンに大きく依存する．これは，過去に蓄積されたライブラリの中でも廃止されたものや新たに追加されたものなどでmizar ライブラリの内容がバージョンによって異なったり証明文の文法が変更になったりしたためである．このことによる弊害としては，数年前に作成した課題が現在のproof checkerではエラーになり正常

に認識されないことがある．そのため，学習課題作成時に教師は，対象となる mizar のバージョンを選択できるようにしている．

3.2 proof checkの仕組み

学生が入力したデータは，一旦ファイルとしてサーバ内に保存される．そのファイルをあらかじめサーバにインストールしているmizar　**proof checker** に受け渡し**proof check**を行う．これが実行されたファイルには，エラーがあればエラーコードとそれに対応するエラーメッセージが追記される．
その後，エラーが記述された証明文のソースファイルを再びモジュールが読み込み修正するため表示する．

3.3 学習の進め方

一般的な学習の進め方は
1)コース上の教材を読み概要を理解する．
2) mizar の課題に挑戦する
3) 不合格ならば再挑戦，合格ならば次の課題へ
という流れをとる．そのため教材には課題解答時のヒントを記述することで，課題に挑戦している際学生が解答しやすいように配慮している．
一通り教材に目を通して内容をある程度理解した学生は，本モジュールで作成した課題を解答する．

学生は，このシステムを使用して，プログラム実習のように，2.1 の形式化記述例で述べたような，簡単な命題の証明から，下記のような教養課程１,２年次の微積分などの定理の証明を記述する演習を受ける。

```
theorem
 for n be Nat, f be PartFunc of REAL,REAL, x0,r be Real st
  ( 0 < r & f is_differentiable_on n+1,  ].x0-r,x0+r.[ )
    for x be Real st x in ].x0-r, x0+r.[ holds
      ex s be Real st 0 < s &  s < 1 &
        f.x=Partial_Sums(Taylor(f, ].x0-r,x0+r.[,x0,x)).n
                + (diff(f,].x0-r,x0+r.[).(n+1)).(x0+s*(x-x0))
                * (x-x0) |^ (n+1)  /((n+1)!);
```

プログラム言語の演習と異なり，記号論理を知られる読者であれば，少々長い数学定理の証明を記述させれば，行き当たりばったりで証明を考えても，その安易な計画性のない試みは挫折し，最初から証明を組織的に考え，途中経過に，適当に補題を配置するなど,論理思考の積み重ねを余儀なくされることは容易に推察できよう。

証明の記述であるから,『解』は一通りには決まらない。与えられた公理や既に証明済みの定理から論理的に結論を導ければ良いので， 回答者の数だけ証明もあるといってよい。例えば，上の簡単な例ですら，n * (n - 1) を　2*(s+n) に変形する過程は幾通りかあるだろう。まして，少々複雑な証明問題なら，証明の組み立て方自体が種々なものになり，学生の数だけ答案がある。そのような課題を，レポート添削に形で,教官が添削したら相当な時間が必要であろう。しかも全ての学生が,正しい証明を書いてくるまで,繰り返し答案に添削を正確に行うことは余程の努力が必要であろう。

このプルーフチェッカーは，計算機の処理能力が許す限り，同時に何人の学生でも利用できる。また，学生のノート PC 上でも動作できる。疲労をすることなく，感情的になることもなく，機械的正確さをもって学生の証明が完成するまで，何万回でも添削し続ける。

４．終わりに

以上，　Mizar の概要,それを用いたプロジェクトの現状と,日本おける同プロジェクトへの参加状況を報告した。　既に述べたが,筆者らは現在,

(1)学部や修士学生が学習する微積分や関数解析,代数等の基礎的定理のライブラリ整備

(2)暗号理論や情報工学分野での形式検証に必要なライブラリの整備

を当面の目標にライブラリ作成に取り組んでいる。

また,

(3)　それの副産物として，数理的思考訓練用の教材を汎用 CMS に組み込む研究も行なっている。

夢想(妄想?)　と期待を述べれば，殆どの数学定理のライブラリ化が完成し，数学，数理科学，応用数学分野の研究者が，形式化数学記述システムを用いて自分たちの最新の研究成果とその証明を記述できるようになることを期待している。証明の正しさは，どのような無名な研究者の論文であっても，学会の権威者ではなく，計算機が瞬時に判定する。従来の人間の査読は人間系によってのみ可能な結果の有用性といったより高度な判定に利用される。プロ，アマチュアの境がなくなり，学問研究の数理科学分野でのユビキタスが実現されることを期待している。