FOLIA 82

Annales Universitatis Paedagogicae Cracoviensis

Studia ad Didacticam Mathematicae Pertinentia III (2010)

Ewa Borak

System komputerowy MIZAR narzędziem do komputerowego wspomagania nauczania w szkole wyższej*

Abstract. The MIZAR system is a computer system for representing mathematical proofs in such a way that the computer checks their correctness. The texts written in the MIZAR language are called *Mizar articles* and are organized into the *Mizar Mathematical Library* (MML). Since the very beginning of the development of the MIZAR system, experiments using MIZAR as a tool for teaching mathematics have been conducted. Numerous courses were organized which were based on different versions of the system: starting from the first implementation of its processor, through MIZAR-MSE, MIZAR-4 and PC-MIZAR, up till its current version. The first MIZAR-aided classes were introduced in 1975. In this paper we present the MIZAR system and the Mizar language and discuss courses conducted in the Institute of Informatics at the University of Białystok. These courses employ MIZAR as the main tool of instruction.

Wstęp

MIZAR jest międzynarodowym projektem[1], który rozpoczął się w Płocku w 1973 roku. Pomysłodawcą i koordynatorem projektu był Andrzej Trybulec, ówczesny pracownik Filii Politechniki Warszawskiej w Płocku, obecnie pracownik Uniwersytetu w Białymstoku. Głównym celem projektu jest stworzenie systemu do komputerowo wspomaganej formalizacji matematyki (w skład którego wchodzą: język MIZAR i niezbędne oprogramowanie do sprawdzania poprawności zapisanych w tym języku tekstów), a także utworzenie bazy danych dla matematyki. W tworzeniu bazy danych wzięło udział 219 autorów z wielu krajów świata (m.in. z Polski, Japonii, Kanady, Chin Kontynentalnych, Niemiec).

Język MIZAR jest językiem służącym do praktycznej formalizacji matematyki. Bazuje na logice pierwszego rzędu, a dowody pisane są w stylu naturalnej dedukcji zaproponowanym przez Jaśkowskiego (1934) (por. Fitch, 1952; Ono, 1962). Wszystkie teksty napisane w języku MIZAR nazywane są *artykułami Mizarowymi* i tworzą matematyczną bazę danych *Mizar Mathematical Library* (MML). MML bazuje na teorii zbiorów Tarskiego-Grothendiecka[2] (Trybulec, 1990).

*The MIZAR computer system as a tool for a computer assisted teaching in higher education

[1] http://mizar.org

[2] http://en.wikipedia.org/wiki/Tarski-Grothendieck_set_theory

W czasie trwania prac nad rozwojem systemu w sposób naturalny zrodził się pomysł, aby system MIZAR wykorzystywać jako narzędzie do wspomagania nauczania matematyki. Różne wersje systemu stosowane były w dydaktyce matematyki na różnych poziomach edukacyjnych, od szkoły średniej począwszy, a skończywszy na pisaniu tez doktorskich. Historię kursów Mizarowych omówiono w punkcie 3.

1. System MIZAR

MIZAR jest to język, w którym zapisuje się tekst matematyczny (definicje, twierdzenia, dowody), system komputerowy, służący do sprawdzenia poprawności tego tekstu oraz matematyczna baza danych (MML). Tekst Mizarowy, zwany dalej artykułem, tworzy się przy użyciu zwykłego edytora tekstowego (Emacs, notatnik lub inne). Przykład takiego tekstu podany jest poniżej (dowód faktu z zakresu teorii relacji binarnych: *R jest relacją przechodnią wtedy i tylko wtedy, gdy $R \circ R \subseteq R$*; symbolem złożenia relacji w języku MIZAR jest $*$).

```
environ
vocabularies ENUMSET, RELATION;
notations ENUMSET, RELATION;
constructors ENUMSET,RELATION;
theorems ENUMSET, RELATION;
definitions ENUMSET, RELATION;

begin

reserve P,Q,R,S,T for Relation, x,y,z,v for set;

:: zadanie 1

 R is transitive iff R*R c= R
  proof
   thus R is transitive implies R*R c= R
    proof
     assume a1: R is transitive;
     let x,y; assume [x,y] in R*R;
     then consider z such that
      a2: [x,z] in R and a3: [z,y] in R by RELATION:def 7;
     thus [x,y] in R by a2,a3,a1,RELATION:def 12;
    end;
   thus R*R c= R implies R is transitive
    proof
      assume a1: R*R c= R;
      let x,y,z; assume [x,y] in R & [y,z] in R;
      then [x,z] in R*R by RELATION:def 7;
      hence [x,z] in R by a1,RELATION:def 9;
    end;
  end;
```

Chwilowo pomińmy znaczenie poszczególnych elementów języka występujących w tym przykładzie. Zostaną one opisane w dalszej części artykułu. Zapewne jednak i bez tego można wyróżnić dwie składowe powyższego tekstu, wyznaczone przez słowa `environ` i `begin`. Część *Deklaracji Środowiskowych* (to ta rozpoczynająca się słowem `environ`) składa się z ciągu *Dyrektyw* (vocabularies, notations,

constructors, registrations, definitions, theorems, schemes), natomiast część *Tekstu Właściwego* (rozpoczynająca się od `begin`) jest tekstem napisanym z inwencji autora artykułu. Najczęściej będą to twierdzenia czy definicje i ich dowody.

1.1. Język MIZAR

Język MIZAR jest językiem formalnym, tzn. językiem stworzonym do zapisywania tekstów w taki sposób, aby komputer mógł sprawdzić ich poprawność. Jak można zauważyć z podanego powyżej przykładowego tekstu Mizarowego, język ten jest bardzo zbliżony do standardowo używanego w tekstach matematycznych języka angielskiego. Można powiedzieć, że słownik języka MIZAR jest podzbiorem słownika języka angielskiego. Zawiera 27 specjalnych symboli i 110 zarezerwowanych słów – w tablicy 1 podano najważniejsze z nich. W tablicy 2 podano kilka przykładowych formuł języka MIZAR.

Tabela 1. Notacja Mizarowa

spójniki logiczne		**spójniki logiczne w Mizarze**
negacja	$\sim$	`not`
koniunkcja	$\wedge$	`&`
alternatywa	$\vee$	`or`
implikacja	$\rightarrow$	`implies`
równoważność	$\leftrightarrow$	`iff`
kwantyfikatory		**kwantyfikatory w Mizarze**
egzystencjalny	$\exists$	`ex`
ogólny	$\forall$	`for`
symbole teorio-mnogościowe		**symbole teorio-mnogościowe w Mizarze**
suma	$\cup$	`\/`
przekrój	$\cap$	`/\`
różnica	$\setminus$	`\`
należenie	$\in$	`in`
zawieranie	$\subseteq$	`c=`
zbiór pusty	$\emptyset$	`{}`

Tabela 1. Przykładowe formuły w języku MIZAR

formuła	**formuła w Mizarze**
$\bot$	`contradiction`
$\neg\phi$	`not` ϕ
$\phi \vee \psi$	ϕ `or` ψ
$\phi \wedge \psi$	ϕ `&` ψ
$\phi \rightarrow \psi$	ϕ `implies` ψ
$\phi \leftrightarrow \psi$	ϕ `iff` ψ
$\forall_x P[x]$	`for x holds P[x]`
$\exists_x P[x]$	`ex x st P[x]`

Sama znajomość notacji Mizarowej nie wystarczy do poprawnego napisania jakiegokolwiek dowodu. Dlatego też każdy użytkownik systemu powien poznać taktyki dowodzenia i sposoby uzasadniania kolejnych kroków dowodowych.

1.2. Taktyki dowodzenia

Tak jak rzecz się ma w przypadku języka naturalnego, tak i w języku MIZAR ten sam dowód może być zapisany w różnym stylu. Wybór stylu zależy od autora tekstu, jednakże biorąc pod uwagę fakt, że jego poprawność ma być zweryfikowana przez system komputerowy, autor powinien przestrzegać sprecyzowanych reguł pisania. Dowody w Mizarze pisane są w stylu naturalnej dedukcji. W każdym miejscu dowodu mamy tzw. „aktualną tezę" do udowodnienia. Teza ta (stwierdzenie, które w danym momencie mamy udowodnić) jest oznaczana słówkiem `thesis`, ale oczywiście można ją (w miarę potrzeby) wypisywać explicite. I tak, na początku dowodu zmienna formuła `thesis` oznacza główną tezę dowodu, jednakże kolejne składowe szkieletu dowodu dynamicznie zmieniają wartość tej zmiennej.

Poniżej podane są niektóre taktyki dowodzenia. Podstawowe to: taktyka założeniowa, taktyka adjunkcji, taktyka generalizacji, taktyka egzemplifikacji. Obok kolejnych kroków dowodowych podano wartość zmiennej `thesis`.

1. *Taktyka założeniowa*
   Aby udowodnić tezę w postaci implikacji, należy założyć jej poprzednik i udowodnić następnik:

```
A implies B              :: thesis = A implies B
  proof
   assume a1: A;         :: thesis = B
   ...
   thus B;               :: thesis = {}
  end;
```

2. *Taktyka adjunkcji*
   Aby udowodnić tezę w postaci koniunkcji, musimy udowodnić każdy z jej członów:

```
A & B                    :: thesis = A & B
  proof
   ...
   thus A;               :: thesis = B
   ...
   thus B;               :: thesis = {}
  end;
```

3. *Taktyka generalizacji*
   Kiedy dowodzimy zdania ogólnego, używamy słówka `let` w sposób następujący:

```
for x holds P[x]         :: thesis = for x holds P[x]
  proof
   let x;                :: thesis = P[x]
   ...
   thus P[x];            :: thesis = {}
  end;
```

4. *Taktyka egzemplifikacji*
   Dowodząc zdania egzystencjalnego, używamy słówka `take`:

```
ex x st P[x]              :: thesis = ex x st P[x]
  proof
   ...
   take a;                :: thesis = P[a]
   ...
   thus P[a];             :: thesis = {}
  end;
```

Wszystkie dopuszczalne w Mizarze taktyki dowodzenia twierdzeń omówione są w książkach „An Outline of PC Mizar" M. Muzalewskiego (1993) oraz „Mizar Lecture Notes 4^{th} Edition" Y. Nakamury, T. Watanabe, Y. Tanaki, P. Kawamoto (2002), a tutaj warto jeszcze wskazać następujące:

- *Dowód wprost*

```
A                         :: thesis = A
 proof
  ...
  thus A;                 :: thesis = {}
 end;
```

- *Dowód przez sprzeczność*

```
A                         :: thesis = A
 proof
  assume not A;           :: thesis = contradiction
  thus thesis;            :: thesis = {}
 end;
```

- *Dowód alternatywy zdań*

```
A or B                    :: thesis = A or B
 proof
  assume not A;           :: thesis = B
  thus thesis;            :: thesis = {}
 end;
```

- *Dowód tezy w postaci implikacji, której poprzednik jest alternatywą zdań*

```
(A or B) implies C        :: thesis = (A or B) implies C
 proof
 assume a1: A or B;       :: thesis = C
  per cases by a1;
   suppose A;             :: thesis = C
    ...
    thus thesis;          :: thesis = {}
   end;
   suppose B;             :: thesis = C
    ...
    thus thesis;          :: thesis = {}
   end;
 end;
```

Język MIZAR pozwala na pewną swobodę (wyznaczoną przez styl dowodzenia) doboru techniki dowodzenia. Podobnie jest z uzasadnianiem kolejnych kroków. Część faktów jest uzasadniana automatycznie (nie trzeba wskazywać wprost, z jakich poprzednio udowodnionych stwierdzeń wnioskuje się prawdziwość danego). Jednakże większą część zdań trzeba uzasadnić. Podstawowe metody to: uzasadnianie poprzez etykiety i linkowanie.

Jeśli można stwierdzić prawdziwość jakiegoś stwierdzenia w jednym kroku, odwołując się do faktu wcześniej uzasadnionego bądź założenia, to mamy wówczas prostą metodę uzasadniania: po zdaniu, które uzasadniamy, piszemy słówko `by` (czy `from` – jeśli odwołujemy się do schematu), a następnie wypisujemy etykiety zdań, na które się powołujemy:

```
a1:  stwierdzenie1;
a2:  stwierdzenie2;
a3:  stwierdzenie3;
     stwierdzenie4 by a1,a2,a3,TARSKI:2,BOOLE:def 3;
```

`a1,a2,a3` są etykietami zdań bieżącego dowodu, `TARSKI:2` oznacza odwołanie się do drugiego twierdzenia w artykule `TARSKI`, a `BOOLE:def 3` – do trzeciej definicji w artykule `BOOLE`.

Istnieje również możliwość odwołania się do stwierdzenia z poprzedniego kroku dowodu bez użycia etykiety. Mamy wówczas do czynienia z mechanizmem linkowania: po zdaniu, na które chcemy się powołać, piszemy słówko `then`. Popatrzmy, jak zmieni się powyższy przykład, gdy powołanie się na `stwierdzenie3` poprzez etykietę `a3` zamienimy na linkowanie:

```
a1:  stwierdzenie1;
a2:  stwierdzenie2;
     stwierdzenie3;
then stwierdzenie4 by a1,a2,TARSKI:2,BOOLE:def 3;
```

Mechanizmu linkowania można też używać wtedy, gdy chcemy stwierdzić konkluzję, powołując się na poprzednie zdanie. W tym wypadku słówko `thus` (stwierdzenie konkluzji) zastępujemy słówkiem `hence`, które oznacza tyle, co `then thus`. Warto wspomnieć, że w języku MIZAR nie ma wyrażenia `then thus` – zastępuje je właśnie słówko `hence`.

Trochę inaczej wygląda uzasadnienie stwierdzenia poprzez powołanie się na schemat. Chcąc na przykład powołać się na schemat indukcji naturalnej (zapisany w artykule `NAT_1`), musimy najpierw wypisać i uzasadnić wszystkie kroki indukcyjne, a potem użyć słówka `from` następująco:

```
a: pierwszy krok indukcyjny; b: drugi krok indukcyjny;
   stwierdzenie from NAT_1:sch 1(a,b);
```

Trzeba dodać, że etykiety `a` i `b` odwołujące do kroków indukcyjnych (czy innych założeń innego schematu) muszą być wypisane w nawiasach i w takiej kolejności, jak w definicji schematu.

1.3. Weryfikacja dowodu

Do weryfikacji tekstu zapisanego w języku MIZAR używa się programu `mizf`. Tekst ten jest wtedy przetwarzany przez dwa różne programy: `Acommodator` oraz `Verifier`. Najpierw `Acommodator` przetwarza *Deklaracje Środowiskowe* i tworzy lokalne środowisko, w którym przechowywane są informacje pożądane przez *Dyrektywy*, a pobrane z bazy. Potem `Verifier` sprawdza poprawność *Tekstu Właściwego*, używając jedynie informacji przechowywanych w lokalnym środowisku, bez komunikacji z bazą. Wynikiem jego działania są uwagi (w postaci numerów błędów) naniesione na niezaakceptowane fragmenty tekstu. Autor nanosi poprawki i uruchamia program `mizf` aż do momentu, gdy nie będzie zgłoszonych żadnych uwag (błędów). Oto przykładowy wynik działania programu `mizf`:

```
environ
 vocabularies ENUMSET, RELATION;
 notations ENUMSET, RELATION;
 constructors ENUMSET, RELATION;
 theorems ENUMSET, RELATION;
 definitions ENUMSET, RELATION;

begin

reserve P,Q,R,S,T for Relation, x,y,z,v for set;

 R is transitive iff R*R c= R
  proof
   thus R is transitive implies R*R c= R
    proof
     assume a1: R is transitive
::>                              *330
     let x,y; assume [x,y] in R*R;
::>                *52
     then consider z such that
      a2: [x,z] in R and a3: [z,y] in R by RELATION:def 7;
::>                                        *4
     hence [x,y] in R by a2,a3,a1,RELATION:def 12;
::>      *164
    end;
   thus R*R c= R implies R is transitive
    proof
      assume a1: R*R c= R;
      let x,y,z; assume [x,y] in R & [y,z] in R;
      then [x,z] in R*R by RELATION def 7;
::>                                  *384
      hence [x,z] in R by RELATION:def 9;
    end;
  end;
::> 4: This inference is not accepted
::> 52: Invalid assumption
::> 164: Nothing to link
::> 330: Unexpected end of an item (perhaps ";" missing)
::> 384: ":" expected
```

Jak widać, na końcu sprawdzanego przez MIZAR tekstu pojawiają się objaśnienia błędów, które zostały zgłoszone. Dzięki temu użytkownik może łatwo poprawić

swój tekst. Najczęściej występującymi błędami są 4 oraz 70. Ten pierwszy pojawił się w naszym przykładzie i oznacza, że fakt zapisany w linii, pod którą MIZAR wypisał 4, nie jest przez niego zaakceptowany (być może nie jest prawdziwy, bądź został źle uzasadniony). Błąd 70 oznacza brak dowodu. Błędy 330 i 384 to zwykłe literówki: brakuje odpowiednio `;` oraz `:`. Błąd 52 – nieprawidłowe założenie – jest w tym wypadku konsekwencją błędu 330, natomiast błąd 164 jest naniesiony pod słówkiem `hence` (`hence` to to samo, co `then thus`) z tego powodu, że po użytej w poprzednim zdaniu konstrukcji `consider` nie można napisać `then`. Po dopisaniu `;` oraz `:` w liniach, pod którymi są zgłoszone błędy 330 i 384 oraz po zamianie słówka `hence` na `thus` (błąd 164) powyższy tekst zawiera tylko jeden błąd:

```
environ
 vocabularies ENUMSET, RELATION;
 notations ENUMSET, RELATION;
 constructors ENUMSET, RELATION;
 theorems ENUMSET, RELATION;
 definitions ENUMSET, RELATION;

begin

reserve P,Q,R,S,T for Relation, x,y,z,v for set;

 R is transitive iff R*R c= R
  proof
   thus R is transitive implies R*R c= R
    proof
     assume a1: R is transitive;
     let x,y; assume [x,y] in R*R;
     then consider z such that
      a2: [x,z] in R and a3: [z,y] in R by RELATION:def 7;
     thus [x,y] in R by a2,a3,a1,RELATION:def 12;
    end;
   thus R*R c= R implies R is transitive
    proof
      assume a1: R*R c= R;
      let x,y,z; assume [x,y] in R & [y,z] in R;
      then [x,z] in R*R by RELATION:def 7;
      hence [x,z] in R by RELATION:def 9;
::>                       *4
    end;
  end;
::> 4: This inference is not accepted
```

Zgłaszając błąd 4, MIZAR zaznacza, że dana inferencja nie jest przez niego zaakceptowana. Aby pozbyć się tego błędu, a tym samym skończyć dowód, należy uzasadnić wskazaną tezę, powołując się na założenie `a1: R*R c= R`.

W jaki sposób MIZAR sprawdza poprawność przeprowadzonego wnioskowania?

Mamy do sprawdzenia inferencję:

$$\psi \texttt{ by } \phi_1, \phi_2, ..., \phi_k$$

tzn. inferencję postaci

$$\frac{\phi_1, ..., \phi_k}{\psi}$$

MIZAR (jest disproverem) neguje konkluzję, a następnie dołącza ją do przesłanek i próbuje otrzymać sprzeczność, czyli wyjściową inferencję sprowadza do następującej:

$$\frac{\phi_1, ..., \phi_k, \neg\psi}{\bot}.$$

W następnym kroku „tworzona" jest koniunkcja wszystkich przesłanek, która z kolei sprowadzana jest do postaci normalnej alternatywno-koniuncyjnej, a przez to wyjściowa inferencja jest równoważna poniższej:

$$\frac{[\neg]\phi_{1,1} \wedge ... \wedge [\neg]\phi_{1,k_1} \vee ... \vee [\neg]\phi_{n,1} \wedge ... \wedge [\neg]\phi_{n,k_n}}{\bot},$$

gdzie $\phi_{i,j}$ są zdaniami atomowymi bądź uniwersalnymi (zanegowanymi lub nie). Teraz MIZAR próbuje obalić prawdziwość każdego członu alternatywy.

Aby inferencja była zaakceptowna przez system, muszą zostać obalone wszystkie człony alternatywy. Stąd w rzeczywistości system weryfikuje n inferencji:

$$\frac{[\neg]\phi_{1,1} \wedge ... \wedge [\neg]\phi_{1,k_1}}{\bot}$$

$$\vdots$$

$$\frac{[\neg]\phi_{n,1} \wedge ... \wedge [\neg]\phi_{n,k_n}}{\bot}.$$

Wewnętrznie do przetwarzania formuł używa się systemu korelatów semantycznych (porównaj z systemem korelatów semantycznych zaprezentowanym przez R. Suszko (1968)). Formuły sprowadzane są do tzw. uproszczonych postaci normalnych – ich korelaty semantyczne wraz ze zdaniami atomowymi używają tylko `VERUM`, `not`, `&`, `for__holds__`. Wszystkie pozostałe spójniki logiczne ($\vee$, $\rightarrow$, $\leftrightarrow$) oraz kwantyfikator egzystencjalny $\exists$ wyeliminowane są dzięki użyciu następujących reguł:

1. Elementem neutralnym koniunkcji jest `VERUM`.
2. Koniunkcja jest łączna, ale nie jest przemienna.
3. `not` (`not` ϕ) jest równoważne ϕ (reguła podwójnej negacji).
4. Zachodzą prawa de'Morgana dla alternatywy oraz kwantyfikatora egzystencjalnego.
5. ϕ `implies` ψ jest równoważne `not` (ϕ `&` `not` ψ).
6. ϕ `iff` ψ jest równoważne `not` (ϕ `&` `not` ψ) `&` `not` (ψ `&` `not` ϕ).

Biorąc pod uwagę brak przemienności koniunkcji, inferencja

$$\frac{\forall_x \exists_y P[x,y] \wedge Q[x,y]}{\exists_y Q[a,y] \wedge P[a,y]}$$

nie będzie zaakceptowana przez system, ale

$$\frac{\forall_x \exists_y P[x,y] \wedge Q[x,y]}{\exists_y P[a,y] \wedge Q[a,y]}$$

oczywiście tak.

Więcej informacji o systemie można znaleźć w pracy „A Brief Overview of Mizar" A. Naumowicza, A. Korniłowicza (2009).

Jak wygląda sprawdzenie poprawności przeprowadzonego wnioskowania z punktu widzenia użytkownika MIZARa?

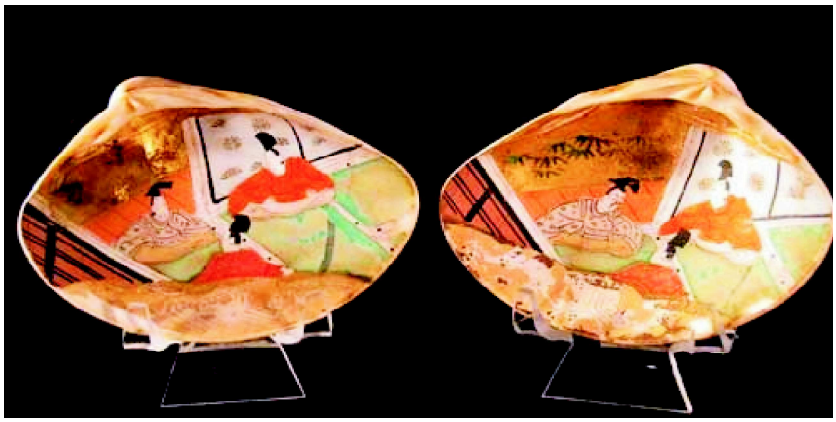
Zupełnie inaczej weryfikację poprawności dowodu widzi autorka artykułu. Sprawa jest bardzo prosta: wystarczy wywołać program `mizf` i po chwili na ekranie monitora pojawia się wynik; jeśli MIZAR nie zgłosił błędów, to przeprowadzone rozumowanie jest poprawne, w przeciwnym wypadku – nie. Można powiedzieć, że idea systemu MIZAR jest analogiczna do idei w starej japońskiej grze w muszle Kai-awase[3].

Gracz, chcąc odnaleźć połówkę muszli „jigai" pasującą do wyciągniętej przez niego z pudełka połówki „degai", korzysta tylko z obrazków na nich namalowanych; odwołując się jednocześnie do swojej wiedzy i doświadczenia, gracz może np. dopasować kolory czy odgadnąć historyjkę przedstawioną na muszlach. Jednakże fizyczne sprawdzenie, czy odnalezione połówki małży rzeczywiście do siebie pasują, polega na złożeniu ich ze sobą. Z powodu bardzo dużego zróżnicowania genetycznego muszli małży (nie ma dwóch identycznych par muszli), w ten jakże szybki sposób gracz uzyskuje odpowiedź.

Podobnie autor (m.in. student) piszący dowód w Mizarze, szukając dowodu, korzysta z różnorodnej wiedzy i umiejętności (np. dostrzega i wykorzystuje analogię z podobnymi dowodami, ilustruje problem np. grafem), a fizyczne sprawdzenie, czy dowód został poprawnie dopasowany do dowodzonej formuły następuje poprzez wywołanie programu `mizf`. Różnica między grą Kai-awase a Mizarem jest taka, że w grze w muszle gracz ma tylko jedną możliwość sprawdzenia poprawności rozwiązania, natomiast w Mizarze – wiele (interaktywna praca autora z komputerem).

[3]Gra ta była jedną z ulubionych rozrywek w dworskich kręgach w okresie Edo (1603-1868). Japońskie słowo *kai* to muszla, *awaseru* oznacza dopasować, zgrać, połączyć. Klasyczny zestaw gry Kai-awase zawierał 360 par muszli małży o podobnych wymiarach. Każda para miała wewnątrz namalowaną historyjkę (obrazki nie były identyczne) opartą na motywach klasycznej prozy, poezji bądź związaną z porą roku. Muszle na zewnątrz pozostawione były w stanie naturalnym. Podczas gry, gdy muszle „degai" (prawe połówki) były wyjmowane z pudełka, ich lewe połówki „jigai" były odnajdywane wśród rozłożonych muszli na podłodze malowaną stroną do dołu. Wygrywała osoba, która zebrała najwięcej jednakowych par. Muszle „degai" i „jigai" nie miały wewnątrz identycznych obrazków, ale dosłowne dopasowanie występowało w parze łączącej dwie połówki.

Na rysunku 1 umieszczona jest przykładowa para muszli do gry Kai-awase[4].



Rysunek 1. Para muszli do gry Kai-awase.

2. System MIZAR w edukacji

2.1. Rys historyczny

Historia kursów Mizarowych[5] jest dość długa. Pierwsze z nich pojawiły się w 1975 roku, już w dwa lata od momentu rozpoczęcia prac nad projektem. Były to kursy z logiki zdań prowadzone równolegle na dwóch uczelniach: pierwszy – przez Romana Matuszewskiego na Filii Politechniki Warszawskiej w Płocku, a drugi – przez Andrzeja Trybulca na Uniwersytecie Warszawskim.

MIZAR-MSE (Trybulec, 1982) – kolejna i bogatsza wersja systemu – wykorzystywany był jako narzędzie do wspomagania nauczania matematyki w ramach wielu kursów. W październiku 1983 roku rozpoczął się korespondencyjny kurs „Wstęp do logiki". Prowadzony był za pośrednictwem miesięcznika *Delta*, czasopisma skierowanego do uczniów szkół średnich. Materiał podzielony był na 10 części, a każda część była opracowywana przez K. Prażmowskiego oraz P. Rudnickiego i publikowana w *Delcie* (1983a; 1983b; 1983c; 1983d; 1984a; 1984b; 1984c; 1984d; 1984e; 1984f). Wraz z treściami kursu zamieszczano zestaw zadań do samodzielnego rozwiązania. Uczniów zachęcano do tego, aby swoje rozwiązania przesyłali drogą elektroniczną do redakcji *Delty* w celu sprawdzenia ich poprawności przez system MIZAR-MSE. Nadesłane prace domowe były zapisywane w języku MIZAR-MSE, weryfikowane i odsyłane do ich autorów. W ten sposób sprawdzono 241 rozwiązań.

Począwszy od roku akademickiego 1985-86, na różnych wydziałach Filii Uniwersytetu Warszawskiego w Białymstoku prowadzono wspomagane Mizarem-MSE jednosemestralne kursy z podstaw logiki (opisane w „An Application of Mizar-MSE in a Course in Logic" przez A. Zalewską (1987)).

[4]Zdjęcie pochodzi ze strony www.japonia.org.pl

[5]Przez wyrażenie kursy Mizarowe należy rozumieć kursy czy zajęcia wspomagane systemem MIZAR.

Tę wersję systemu wykorzystywano też eksperymentalnie do:

- nauczania logiki na Wydziale Informatyki Uniwersytetu Jagiellońskiego (Moszner, 1989). Zorganizowano wówczas trzy rodzaje kursów:
  1. szkolne koło informatyczne dla uczniów szkół średnich,
  2. seminarium z logiki dla studentów pierwszego roku informatyki,
  3. w ramach zajęć „Komputerowe wspomaganie nauczania" dla studentów starszych lat,
- nauczania podstaw geometrii na Wydziale Matematyki Uniwersytetu Warszawskiego (Szczerba, 1987). Zajęcia prowadzone były na podstawie podręcznika „Geometria dla nauczycieli" M. Kordosa, L. Szczerby (1976), jednakże systemu MIZAR-MSE można było użyć jedynie do części materiału zawartego w tymże podręczniku. Wynikało to z faktu, że w MIZARZE-MSE nie było możliwości zapisywania formuł z notacją funkcyjną. Udało się jednak stworzyć odpowiednie środowisko zawierające niezbędne fakty z geometrii. Wybrano też z podręcznika odpowiednie twierdzenia, które w przygotowanym środowisku można było udowodnić. Zadaniem studentów było zapisanie i udowodnienie tych twierdzeń w języku MIZAR, a następnie komputerowa weryfikacja ich poprawności. Dzięki temu studenci nabywali umiejętności dowodzenia (prostych) twierdzeń z geometrii.

W 1986 roku pojawiła się kolejna, udoskonalona wersja systemu – MIZAR-4. Tej wersji MIZARa używano w latach 1987-88 do wspomagania nauczania „Wstępu do matematyki" oraz „Teorii krat" na Filii Uniwersytetu Warszawskiego w Białymstoku. W tym czasie nastąpił również gwałtowny rozwój systemu i już pod koniec lat 80. pojawiał się PC-MIZAR.

W latach 90. w Instytucie Matematyki FUW próbowano zorganizować kurs Mizarowy z topologii. Nie było jednak w tym czasie możliwości technicznych i organizacyjnych, aby zajęcia te mogły odbywać się w laboratorium komputerowym. Napisano więc 5 skryptów zawierających ćwiczenia oraz ich rozwiązania zapisane w języku MIZAR i sprawdzone przez system, których intensywnie używano podczas tradycyjnie prowadzonych zajęć.

Warto zaznaczyć, że system MIZAR był i jest wykorzystywany do nauczania matematyki na wielu zagranicznych uniwesytetach, np. w USA, Kanadzie (Hoover, Rudnicki, 1994), Japonii, Belgii. Co więcej używany jest też do przygotowywania prac dyplomowych, magisterskich i doktorskich (np. prace: Schwarzwellera (2005), Grądzkiej (2000), Korniłowicza (2001), Naumowicza (2005)).

Pomimo że kursów Mizarowych było w historii projektu wiele, to były prowadzone albo w formie eksperymentów, albo jako kursy oferowane studentom starszych lat w formie wykładów monograficznych. Sytuacja ta diametralnie się zmieniła, kiedy w 2000 roku na Uniwersytecie w Białymstoku (dawna FUW) powstał Instytut Informatyki i wiele osób pracujących nad projektem zostało tam zatrudnionych. Zaistniała możliwość „stałego" stosowania MIZARa w trakcie obowiązkowych zajęć, oferowanych studentom informatyki w ramach programu nauczania. W następnej części tej sekcji omówimy te kursy.

2.2. Kursy Mizarowe w Instytucie Informatyki UwB

MIZAR jako narzędzie wspomagające nauczanie matematyki w Instytucie Informatyki UwB wprowadzany był stopniowo. Na początku, podczas zajęć z „Podstaw logiki i teorii mnogości" (PLiTM) zaczęto wprowadzać notację Mizarową i pisać dowody w języku MIZAR w trakcie tradycyjnie prowadzonych wykładów (30 godzin) i ćwiczeń (30 godzin, bez użycia komputerów). W semestrze zimowym roku akademickiego 2003-04 część ćwiczeń (15 godzin) odbywała się już w laboratorium komputerowym. Studenci tego rocznika kontynuowali naukę matematyki ze wsparciem MIZARa podczas kursu „Formalizacja matematyki" (FM) (Retel, Zalewska, 2005). Kurs FM był obowiązkowy dla wszystkich studentów pierwszego roku, a prowadzony w formie wykładów (30 godzin) i laboratoriów (30 godzin). Ukończyło go 57 osób.

Przez kolejne lata obydwa kursy Mizarowe – PLiTM i FM – obowiązywały dla wszystkich studentów pierwszego roku. W tym czasie nauczyciele akademiccy prowadzący te zajęcia zdobyli już pewne doświadczenie w prowadzeniu kursów z użyciem systemu MIZAR i opracowali metody nauczania, jakie powinne być stosowane podczas zajęć ze studentami[6]. Okazało się także, że wysiłek studentów włożony w roczną naukę języka i systemu MIZAR był opłacalny – studenci sprawnie posługiwali się językiem MIZAR do zapisywania i dowodzenia twierdzeń. Aby umożliwić studentom wykorzystywanie zdobytych przez nich umiejętności z zakresu formalizacji matematyki w Mizarze, wprowadzono w semestrze zimowym dla studentów wszystkich kierunków i lat Instytutu Informatyki.

Obecnie w instytucie prowadzone są następujące kursy z użyciem systemu:

1. Na studiach pierwszego stopnia:
   - Podstawy metod formalnych (PMF),
   - Metody konstruktywne w informatyce (MKwI),
   - Metody abstrakcyjne w informatyce (MAwI).
2. Na studiach drugiego stopnia:
   - Dowodzenie własności programów (DWP),
   - Abstrakcyjne typy danych (ATP).

Kursy na poziomie studiów pierwszego stopnia

„Podstawy metod formalnych" to kurs korespondujący z prowadzonym wcześniej – „Podstawy logiki i teorii mnogości" (organizacja i metodologia kursu PLiTM dokładnie opisane zostały w pracy Borak, Zalewskiej (2007)). To właśnie podczas tego kursu studenci mają zapoznać się z systemem MIZAR oraz nabrać umiejętności formalnego uzasadniania faktów, a przy tym zdobyć wiedzę i umiejętności z zakresu matematyki wykorzystywane jako baza dla innych przedmiotów, np. „Matematyki dyskretnej".

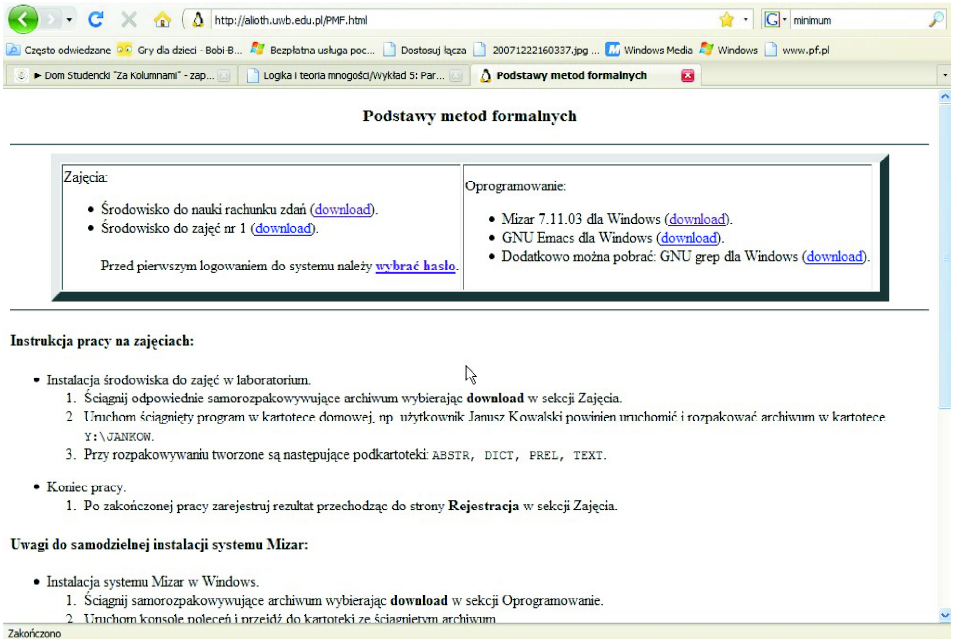
[6] Niektóre metody autorka prezentowała na konferencji MKM w Linzu (Austria) w 2007 roku.

„Podstawy metod formalnych” to kurs przeznaczony dla studentów, którzy nie znają systemu MIZAR, posiadają wiedzę i umiejętności matematyczne na poziomie szkoły średniej w zakresie podstawowym (tylko nieliczni zdawali maturę na poziomie rozszerzonym), a co za tym idzie nie są przygotowani do pracy z całą matematyczną bazą MML, ani nawet z jej fragmentem. Aby umożliwić studentom naukę ze wsparciem systemu MIZAR, tworzone jest pewnego rodzaju „lokalne” środowisko Mizarowe bazujące na części biblioteki MML. Środowisko to składa się z kilku części i tak kolejno budowane są:

1. środowisko do nauki rachunku zdań – PROPOSIT,
2. środowisko do nauki teorii zbiorów – ENUMSET,
3. środowisko do nauki teorii relacji – RELATION oraz RELAT_AB.

Tak przygotowane środowisko sukcesywnie udostępniane jest studentom do pobrania ze strony internetowej kursu (patrz rysunek 2).

Ze strony tej można też pobrać pozostałe oprogramowanie niezbędne do pracy, mianowicie:



Rysunek 2. Internetowa strona: `http://alioth.uwb.edu.pl`

- system MIZAR w wersji aktualnej na dzień 1 października danego roku; ze względu na dynamiczny rozwój systemu wersja ta jest „zamrażana” na czas zajęć ze studentami,
- edytor GNU Emacs; dzięki MizarMode'owi utworzonemu przez J. Urbana (zob. Urban, 2010) jest kompletnym, przyjaznym środowiskiem do pracy z systemem MIZAR,

- Grep, który jest wykorzystywany do przeszukiwania bazy MML dopiero podczas kolejnych kursów.

1. Po krótkich ćwiczeniach związanych z pisaniem i edycją tekstu w edytorze Emacs studenci próbują zweryfikować poprawność najkrótszego – pustego artykułu Mizarowego (tzn. artykułu, który składa się tylko z dwóch zarezerwowanych słów: **environ** i **begin**), po to, aby dowiedzieć się, jak używać MizarMode dla EMACSa i jak MIZAR raportuje błędy.
2. Studenci łączą się następnie ze stroną internetową kursu (rysunek 2), wybierają hasła dostępu do swoich indywidualnych kont na serwerze (na kontach tych rejestrowane są wyniki ich pracy) i ściągają pierwszą część środowiska (PROPOSIT).
3. Po zainstalowaniu na komputerach środowiska PROPOSIT (instalacja pozostałych części środowiska następuje później w taki sam sposób), automatycznie tworzą się 4 katalogi: `abstr`, `dict`, `prel` i `text`, z których tylko dwa – pierwszy i ostatni – są istotne z punktu widzenia studenta. Katalog `abstr` (z wyjątkiem PROPOSIT) zawiera pliki z notacją i definicjami danej części środowiska, a w katalogu `text` jest umieszczony plik `test.miz` z zapoczątkowanym artykułem Mizarowym, stosownie do danego środowiska.
4. Następnie studenci otwierają w EMACSie plik `test.miz` i próbują napisać, a także zweryfikować proste formuły rachunku zdań.
5. Pod koniec zajęć każdy student wysyła plik `test.miz` z rezultatami pracy na swoje indywidualne konto na serwerze *alioth* (tak kończą się wszystkie zajęcia).

Cały kurs PMF, składa się z 30 godzin wykładów oraz 30 godzin laboratoriów, podzielonych na 15 jednostek dwugodzinnych. Pozostałe 14 zajęć podzielonych jest na 3-4 sesje – każda sesja związana jest pracą w innym środowisku.

Główne cele kursu PMF to:

- rozwijanie umiejętności prowadzenia rozumowania dedukcyjnego z użyciem różnych technik dowodzenia, poprzez dobór odpowiednich przykładów z zakresu treści materiału realizowanego w ramach tego kursu,
- rozwijanie umiejętności uzasadniania swoich intuicyjnych rozwiązań w sposób formalny,
- nabycie wiedzy z wybranego zakresu matematyki; w sylabusie zawarte są następujące treści:
  - formuły logiczne,
  - struktura dowodu założeniowego,
  - własności Boole'owskie zbiorów,
  - rodziny zbiorów i ich własności,
  - relacje binarne (złożenie, relacja odwrotna, własności relacji),
  - relacja równoważności, częściowego porządku,
  - funkcje (dziedzina, przeciwdziedzina, obraz, przeciwobraz itd.).

Kolejny kurs na poziomie studiów I stopnia to „Metody konstruktywne w informatyce". Jest on przeznaczony dla studentów drugiego roku (trzeci semestr studiów), od których oczekuje się znajomości języka MIZAR, różnych technik dowodzenia oraz umiejętności pisania dowodów formalnych w edytorze Emacs. W odróżnieniu od kursu poprzedniego, studenci pracują już z całą bazą MML, a do wyszukiwania potrzebnych informacji zapisanych w bazie używają Grepa. W sylabusie tego przedmiotu są następujące treści matematyczne:

- arytmetyka Peano,
- różne schematy indukcji (indukcja naturalna o kroku $1, 2, \ldots, k$, zupełna, na ciągach i zbiorach skończonych, drzewach, liczbach całkowitych i in.),
- schemat minimum,
- dobór predykatu, dla którego prowadzimy indukcję,
- dobór zmiennej, dla której prowadzimy indukcję,
- zastosowanie indukcji w arytmetyce,
- sumy i iloczyny skończone liczb naturalnych,
- definiowanie przez indukcję,
- uzasadnianie za pomocą schematów Mizarowch (stwierdzeń ze zmiennymi wolnymi drugiego rzędu).

Głównym celem ostatniego kursu organizowanego na poziomie studiów licencjackich – „Metody abstrakcyjne w informatyce" – jest zapoznanie studentów z abstrakcyjnymi metodami opisywania różnych obiektów, występujących w informatyce a używanych do opisów i specyfikacji.
Poniżej podane są przykładowe zadania studentów.

ZADANIE 1

```
R is transitive & S is transitive & R*S=S*R implies R*S is transitive
 proof
  assume    E1: R is transitive & S is transitive & R*S=S*R;
  thus R*S is transitive
   proof
    let x,y,z;
    assume E2: [x,y] in R*S & [y,z] in R*S;
     then consider u such that E3: [x,u] in R & [u,y] in S by RELATION:def 7;
     consider v such that E4: [y,v] in R & [v,z] in S by E2,RELATION:def 7;
     [u,v] in S*R by E3,E4,RELATION:def 7;
     then [u,v] in R*S by E1;
     then consider w such that E5: [u,w] in R & [w,v] in S by RELATION:def 7;
     E6: [x,w] in R by E1,E3,E5,RELATION:def 12;
      [w,z] in S by E4,E5,E1,RELATION:def 12;
    hence[x,z] in R*S by E6,RELATION:def 7;
   end;
 end;
```

ZADANIE 2

```
reserve R,S,T for Relation, x for set;
 ex R, S, T st not (R*S)/\(R*T) c= R*(S/\T)
  proof
   reconsider R={[0,1], [0,2]} as Relation by RELATION:9;
```

```
  reconsider S={[1,3]} as Relation by RELATION:8;
  reconsider T={[2,3]} as Relation by RELATION:8;
   take R,S,T;
   assume z1: (R*S)/\(R*T) c= R*(S/\T);
   [0,1] in R & [1,3] in S & [0,2] in R & [2,3] in T by ENUMSET:def 3,def 4;
   then [0,3] in R*S & [0,3] in R*T by RELATION:def 7;
   then z2: [0,3] in (R*S)/\(R*T) by RELATION:def 5;
    not [0,3] in R*(S/\T)
   proof
     assume [0,3] in R*(S/\T);
     then consider x such that z3: [0,x] in R & [x,3] in S/\T by RELATION:def 7;
     [x,3] in S & [x,3] in T by z3,RELATION:def 5;
     then [x,3] = [1,3] & [x,3] = [2,3] by ENUMSET:def 3;
     then x=1 & x=2 by ENUMSET:2;
    hence contradiction;
   end;
  hence contradiction  by z1,z2,RELATION:def 9;
 end;
```

ZADANIE 3

```
ex A be set,Q be Relation of A,A st Q <> {} & Q is connected
 proof
  set A = {1};
  reconsider Q'={[1,1]} as Relation by RELATION:8;
  a: dom Q' c= A
   proof
    let x; assume x in dom Q';
     then consider y such that z: [x,y] in Q' by RELAT_AB:def 1;
      [x,y] = [1,1] by z,ENUMSET:def 3;
     then x=1 by ENUMSET:2;
    hence thesis by ENUMSET:def 3;
   end;
  rng Q' c= A
   proof
    let x; assume x in rng Q';
     then consider y such that z: [y,x] in Q' by RELAT_AB:def 2;
      [y,x] = [1,1] by z,ENUMSET:def 3;
     then x=1 by ENUMSET:2;
    hence thesis by ENUMSET:def 3;
   end;
  then reconsider Q=Q' as Relation of A,A by a,RELAT_AB:def 5;
  take A,Q;
  [1,1] in Q by ENUMSET:def 3;
  hence Q <> {} by RELATION:def 2;
  thus Q is connected
   proof
    let x,y; assume x in A & y in A;
     then x=1 & y=1 by ENUMSET:def 3;
     then [x,y] in Q by ENUMSET:def 3;
    hence [x,y] in Q or [y,x] in Q;
   end;
 end;
```

ZADANIE 4

```
reserve k,n,i for Nat;

scheme { k()-> Nat, P[Nat]}: for n holds P[n]
```

```
 provided
a: for n st n<k() holds P[n] and
b: for n st P[n] holds P[n+k()] and
c: k()>0
  proof
   defpred Q[Nat] means for i st i<k() holds  P[$1+i];
A:  Q[0] by a;
B:  Q[n] implies Q[n+1]
     proof
      assume Q[n];
      then f: for i st i<k() holds  P[n+i];
      thus Q[n+1]
     proof
       let i such that z1: i<k();
       i+1<=k() by z1,INT_1:20;
       then t2:i<=k()-1 by XREAL_1: 21;
     per cases by t2,XXREAL_0: 1;
      suppose e:i=k()-1;
        P[n+0] by f,c;
        then P[n+k()] by b;
        hence P[n+1+i] by e;
      end;
      suppose i<k()-1;
        then i+1<k() by XREAL_1: 22;
        then P[n+(i+1)] by f;
        hence P[n+1+i];
      end;
     end;
     end;
  t1:for n holds Q[n] from NAT_1: sch 2(A,B);
  let n;
  Q[n] by t1;
  P[n+0] by c,t1;
  hence P[n];
 end;
```

Kursy na poziomie studiów drugiego stopnia

Kurs „Dowodzenie własności programów" organizowany dla studentów pierwszego semestru studiów drugiego stopnia wykorzystuje system MIZAR jako narzędzie służące do weryfikacji oprogramowania. Studenci wykazują duże zainteresowanie tematyką kursu, gdyż sprawdzanie poprawności programów to duże wyzwanie dla informatyka. Sylabus DWP zawiera treści:

- operacyjna semantyka języków programowania,
- denotacyjna semantyka języków programowania,
- kryteria poprawności programów,
- matematyczne modele komputerów,
- weryfikacja instrukcji sekwencyjnych, warunkowych, pętli i skoków,
- weryfikacja procedur rekurencyjnych,
- weryfikacja przy użyciu techniki „describera",
- projekt WHY.

Od studenta ostatniego roku studiów drugiego stopnia oczekuje się znajomości systemu na takim poziomie, aby mógł samodzielnie formalizować pewien wybrany fragment teorii matematycznej. Formalizacja ta może stanowić podstawę pracy magisterskiej. Dlatego też głównym celem ostatniego kursu jest umożliwienie studentowi dokonania wyboru odpowiedniej dziedziny do samodzielnej formalizacji. Warto zaznaczyć, że już po serii kursów na poziomie studiów pierwszego stopnia student powinien posiadać wiedzę i umiejętności wystarczające do napisania pracy licencjackiej z użyciem MIZARa.

3. Podsumowanie

System MIZAR jest stosowany jako narzędzie do wspomagania nauczania matematyki w Instytucie Informatyki UwB od roku 2003, ale dopiero wprowadzenie sekwencji kursów Mizarowych pozwoliło na pełniejsze wykorzystanie jego możliwości. Podczas kolejnych kursów studenci zdobywają coraz większą wiedzę na temat samego systemu, przy czym stosuje się zasadę, aby na każdym etapie studiów przekazywać tylko te informacje o MIZARZE, które są niezbędne do realizacji celów nauczania określonych w sylabusach poszczególnych kursów. Dzięki temu nauczanie, a z drugiej strony uczenie się wspomagane systemem MIZAR nie jest wcale utrudnione. Co więcej, z przeprowadzonych przez autorkę niniejszego artkułu badań wstępnych nad oceną przydatności stosowania systemu MIZAR do wspomagania nauczania matematyki wynika jasno, że używanie MIZARa daje dużo korzyści. Hipotezy te wymagają jeszcze weryfikacji.

Literatura

Borak, E., Zalewska, A.: 2007, Mizar course in logic and set theory, w: M. Kauers (red.), *Towards Mechanized Mathematical Assistants, Proceedings of 14th Symposium Calculemus 2007 and 6th International Conference MKM 2007*, LNCS **4573**, 191-204.

Fitch, F. B.: 1952, *Symbolic Logic. An Introduction*, The Ronald Press Company, New York.

Grądzka, E.: 2000, *On the Order-consistent Topology of Complete and Uncomplete Lattices,* M. Sc. thesis, Białystok University, Białystok.

Hoover, H. J., Rudnicki, P.: 1994, *Introduction to Logic in Computing Science, CMPUT* **172**, University of Alberta, Edmonton.

Jaśkowski, S.: 1934, On the rules of supposition in formal logic, *Studia Logica* **1**, 5-32.

Kordos, M., Szczerba, L.: 1976, *Geometria dla nauczycieli*, PWN, Warszawa.

Korniłowicz, A.: 2001, *A Formal Theory of Abstract Computers,* PhD thesis, Shinshu University, Nagano.

Moszner, P.: 1989, Mizar as tool of CAT, *Bulletin of the Association for Computerised Logic Teaching* **2**(2), 16-18.

Muzalewski, M.: 1993, *An Outline of PC Mizar*, Fondation Philippe le Hodey, Brusseles.

Nakamura, Y., Watanabe, T., Tanaka, Y., Kawamoto, P.: 2002, *Mizar Lecture Notes (4^{th} Edition)*, Shinshu University, Departament of Information Engineering, Kiso Lab., Nagano.

Naumowicz, A.: 2005, Mizar *Codification of the Theory of Partial Linear Spaces as an Example of Formalizing Recent Mathematical Results,* PhD thesis, Shinshu University, Nagano.

Naumowicz, A., Korniłowicz, A.: 2009, A brief overview of Mizar, w: S. Berghofer (red.), *TPHOLs 2009, LNCS* **5674**, Springer-Verlag Berlin Heidelberg, 67-72.

Ono, K.: 1962, On practical way of describing formal deduction, *Nagoya Mathematical Journal* **21**, 115-121.

Prażmowski, K., Rudnicki, P.: 1983a, Kurs logiki w Mizarze-MSE (1), *Delta* **9**, 8-9.

Prażmowski, K., Rudnicki, P.: 1983b, Kurs logiki w Mizarze-MSE (2), *Delta* **10**, 13-14.

Prażmowski, K., Rudnicki, P.: 1983c, Kurs logiki w Mizarze-MSE (3), *Delta* **11**, 7.

Prażmowski, K., Rudnicki, P.: 1983d, Kurs logiki w Mizarze-MSE (4), *Delta* **12**, 6.

Prażmowski, K., Rudnicki, P.: 1984a, Kurs logiki w Mizarze-MSE (10), *Delta* **6**, 4.

Prażmowski, K., Rudnicki, P.: 1984b, Kurs logiki w Mizarze-MSE (5), *Delta* **1**, 15-16.

Prażmowski, K., Rudnicki, P.: 1984c, Kurs logiki w Mizarze-MSE (6), *Delta* **2**, 12.

Prażmowski, K., Rudnicki, P.: 1984d, Kurs logiki w Mizarze-MSE (7), *Delta* **3**, 13.

Prażmowski, K., Rudnicki, P.: 1984e, Kurs logiki w Mizarze-MSE (8), *Delta* **4**, 15.

Prażmowski, K., Rudnicki, P.: 1984f, Kurs logiki w Mizarze-MSE (9), *Delta* **5**, 13.

Retel, K., Zalewska, A.: 2005, Mizar as a tool for teaching mathematics, *Mechanized Mathematics and Its Applications* **4**, 25-33.

Schwarzweller, C.: 2005, Mizar *Verification of Generic Algebraic Algorithms,* PhD thesis, Wilhelm-Schicard-Institute for Computer Science, Tubingem.

Suszko, R.: 1968, Ontology in the tractatus of L. Wittgenstein, *Notre Dame Journal of Formal Logic* **9**, 7-33.

Szczerba, L. W.: 1987, The use of Mizar-MSE in a course in foundations of geometry, w: J. Srzednicki (red.), *Initiatives in Logic*, Nijhoff Publishers, Dordrecht, 231-232.

Trybulec, A.: 1982, *Logic Information Language MIZAR-MSE,* ICS PAS Reports, nr 465, Institute of Computer Science Polish Academy of Sciences, Warsaw.

Trybulec, A.: 1990, Tarski Grothendieck set theory, *Formalized Mathematics* **1**(1), 9-11.

Urban, J.: 2010, *Mizar Mode for Emacs*, http://alioth.uwb.edu.pl/twiki/bin/view/Mizar/MizarMode, data dostępu: 1 XII 2010 r.

Zalewska, A.: 1987, Application of Mizar-MSE in a course in logic, w: J. Srzednicki (red.), *Initiatives in Logic*, Nijhoff Publishers, Dordrecht, 224-230.

Instytut Informatyki
Uniwersytet w Białymstoku
ul. Sosnowa 64
15-887 Białystok
e-mail: ewag@ii.uwb.edu.pl