Computer Languages, Systems & Structures 44 (2015) 238–250

Contents lists available at ScienceDirect

Computer Languages, Systems & Structures

journal homepage: www.elsevier.com/locate/cl

Flexary connectives in Mizar

Artur Korniłowicz

Institute of Informatics, University of Białystok, Konstantego Ciołkowskiego 1M, 15-245 Białystok, Poland

ARTICLE INFO

Article history:
Received 5 December 2014
Received in revised form
1 June 2015
Accepted 9 July 2015
Available online 21 July 2015

MSC:
68T15
03B35

Keywords:
Proof assistant
Ellipsis
Flexary conjunction
Flexary disjunction
Mizar

ABSTRACT

One of the main components of the Mizar project is the Mizar language, a computer language invented to reflect the natural language of mathematics. From the very beginning various linguistic constructions and grammar rules which enable us to write texts which resemble classical mathematical papers have been developed and implemented in the language.

The Mizar Mathematical Library is a repository of computer-verified mathematical texts written in the Mizar language. Besides well-known and important theorems, the library contains series of some quite technical lemmas describing some properties formulated for different values of numbers. For example the sequence of lemmas

```
for n being Nat st n <= 1 holds n = 0 or n = 1;
for n being Nat st n <= 2 holds n = 0 or n = 1 or n = 2;
for n being Nat st n <= 3 holds n = 0 or n = 1 or n = 2 or n = 3;
```

which for a long time contained 13 such formulae.

In this paper, we present an extension of the Mizar language – an ellipsis that is used to define flexary logical connectives. We define flexary conjunction and flexary disjunction, which can be understood as generalization of classical conjunction and classical disjunction, respectively. The proposed extension enables us to get rid of lists of such lemmas and to formulate them as single theorems, e.g.

```
for m,n being Nat st n <= m holds n = 0 or...or n = m;
```

covering all cases between the bounds `0` and `m` in this case.

Moreover, a specific inference rule to process flexary formulae, formulae using flexary connectives, is introduced. We describe how ellipses are incorporated into the Mizar language and how they are verified by the Mizar proof checker.

1. Introduction

In mathematics, an ellipsis is used in various contexts and with different meanings. It may mean “and so forth” when patterns of expressions are predictable (e.g. to express consequent odd natural numbers one may write $1, 3, 5, \ldots$); and when patterns are unknown (e.g. to express approximate value of the constant $e \approx 2.71828\ldots$). A very common usage of an ellipsis is to write finite terms without listing all their components (e.g. factorial of the number n as the product $1*2*\ldots*n$). An ellipsis is also used to express “long” vectors, matrices, etc.

E-mail address: arturk@math.uwb.edu.pl
URL: http://math.uwb.edu.pl/~arturk

http://dx.doi.org/10.1016/j.cl.2015.07.002
1477-8424/© 2015 Elsevier Ltd. All rights reserved.

In the paper we focus on another usage of ellipses, namely used to formulate flexary conjunctions and flexary disjunctions, which can be understood as generalizations of classical conjunctions and classical disjunctions. We present how ellipses are incorporated into the Mizar language and are verified by the Mizar proof checker.

The paper is structured as follows. In Section 2 we present basic information and features of the Mizar system required to explain the main track of the paper. In Section 3 we present motivations for using flexary connectives. Experimental results how ellipses influence the content of the Mizar library are shown. In Section 4 we describe how ellipses are implemented in Mizar. Moreover, the way how they are proceeded in the Mizar framework is described.

2. About Mizar

The Mizar project started in 1973 under the leadership of Andrzej Trybulec at the Płock Scientific Society in Poland. Currently, since 1976, it is maintained at the University of Białystok, Poland. The principal goal of the project is to design a computer environment that supports writing traditional mathematical papers under strict control of computer programs that check syntactical, semantical and logical correctness of texts. Mizar is a common name of different components of the Mizar project. It is the name of the project itself. It is the name of a language invented to write mathematical texts to be processed by computers. It is also the name of a bunch of computer programs designed and implemented for processing texts written in the Mizar language, with its core program, a proof checker named Verifier, suitable for formal verification [3,54,66].

Currently there are available dozens of proof assistants usable for proving very advanced mathematical theorems, with leading role of the HOL Light theorem prover,[1] Isabelle,[2] the Coq proof assistant,[3] Metamath,[4] ProofPower,[5] Nqthm/ACL2,[6] the PVS specification and verification system,[7] the Nuprl/PRL project,[8] and Mizar.[9] Mizar differs from mentioned systems by adopted logical and theoretical foundations which are briefly presented in the following sections.

2.1. Language

The Mizar language is a declarative language that allows writing texts resembling traditional mathematical papers. It was invented by Andrzej Trybulec on one hand to enable computers to process mathematical texts written in the language efficiently and on the other hand developed to reflect the natural language of mathematics. It implements rules for writing predicates; terms in prefix, infix and suffix forms; adjectives [36]; types [6]; formulae of various kinds (universal, existential, predicative, attributive, qualifying); definitions; theorems; local lemmas; reasoning methods (including straightforward reasoning, diffused reasoning, proof by exhaustion); proof steps (including generalizations, assumptions, exemplifications, conclusions, linking, references); different ways of introducing variables; syntactic constructions to launch distinguished algorithms for processing particular mechanisms (e.g. term identifications, term reductions [32], properties of predicates and functors [40]); etc.

Over decades many versions of the Mizar language have been issued [35]. Some of them inspired other proof assistants to incorporate the so-called Mizar modes to the systems. The most notable are the Mizar mode for HOL [26], Mizar-light for HOL Light [64], the Mizar mode for Coq [16], the declarative proof language (DPL) for Coq [12], and the Isar proof language for the Isabelle theorem prover [63]. The Mizar way of writing proofs was also the model for the notion of 'formal proof sketches' [65].

For the purposes of this paper we recall some basic information about formulae supported by the Mizar language. We can distinguish three kinds of atomic formulae: *predicative formulae*, *adjective formulae* and *qualifying formulae*. Predicative formulae express relations between objects and use predicates symbols as main operators of the formulae, e.g. `2 in {1,2,3}`. Adjective formulae express that objects satisfy some adjectives and use attributes symbols as main symbols of the formulae, e.g. {`x`} is finite. Qualifying formulae declare that objects are of some types and use mode symbols, e.g. `(#INT,addint#) is Group`. Atomic formulae can be combined into complex formulae using traditional logical connectives (negation `not`, conjunction `&`, disjunction `or`, implication `implies`, and equivalence `iff`) and quantifiers (general quantifier `for` ... `holds` ... and existential quantifier `ex` ... `st` ...). In spite of the fact that in Mizar texts one can use all these logical operators, internally all formulae are represented using general quantifier, negation and conjunction only. Formulae are transformed according to standard logical laws (de Morgan's laws and double negation elimination):

$$\begin{aligned}
&\neg\neg\alpha \quad \rightsquigarrow \quad \alpha\\
&\alpha \vee \beta \quad \rightsquigarrow \quad \neg(\neg\alpha \wedge \neg\beta)\\
&\alpha \Rightarrow \beta \quad \rightsquigarrow \quad \neg(\alpha \wedge \neg\beta)\\
&\alpha \Longleftrightarrow \beta \quad \rightsquigarrow \quad \neg(\alpha \wedge \neg\beta) \wedge \neg(\beta \wedge \neg\alpha)\\
&\exists_x : \alpha(x) \quad \rightsquigarrow \quad \neg\forall_x : \neg\alpha(x)
\end{aligned}$$

1 http://www.cl.cam.ac.uk/~jrh13/hol-light/
2 http://www.cl.cam.ac.uk/research/hvg/Isabelle/
3 http://coq.inria.fr
4 http://us.metamath.org
5 http://www.lemma-one.com/ProofPower/index/
6 http://www.cs.utexas.edu/users/moore/acl2/
7 http://pvs.csl.sri.com
8 http://www.nuprl.org
9 http://mizar.org