J Autom Reasoning (2013) 50:203–210
DOI 10.1007/s10817-012-9261-6

On Rewriting Rules in Mizar

Artur Korniłowicz

Received: 8 December 2011 / Accepted: 30 August 2012 / Published online: 14 September 2012

Abstract This paper presents some tentative experiments in using a special case of rewriting rules in Mizar (Mizar homepage: http://www.mizar.org/): rewriting a term as its subterm. A similar technique, but based on another Mizar mechanism called functor identification (Korniłowicz 2009) was used by Caminati, in his paper on basic first-order model theory in Mizar (Caminati, J Form Reason 3(1):49–77, 2010, Form Math 19(3):157–169, 2011). However for this purpose he was obligated to introduce some artificial functors. The mechanism presented in the present paper looks promising and fits the Mizar paradigm.

Keywords Proof assistant · Natural deduction · Computer algebra system · Term rewriting · Mizar

1 Equalizer

One of the main tasks of proof assistants is checking obviousness of proof steps [4, 12]. The ideal goal is that inferences obvious for the inference checker should be obvious for human readers and vice versa.

In Mizar context it comes down to checking obviousness of a single inference

$$\frac{\phi_1, \phi_2, \ldots, \phi_n}{\psi}$$

A. Korniłowicz (✉)
Institute of Informatics, University of Białystok,
Sosnowa 64, 15-887 Białystok, Poland
e-mail: arturk@mizar.org

where the formulas ϕ_i are premises and ψ is the conclusion. Since MIZAR is a disprover, the conclusion is negated and added to the premises, so such an inference is transformed to

$$\frac{\phi_1, \phi_2, \ldots, \phi_n, \neg\psi}{\lightning}$$

where $\lightning$ means falsehood, or rather to

$$\frac{\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n \wedge \neg\psi}{\lightning}$$

The premise is then translated into its disjunctive normal form (DNF), i.e. a disjunction of conjunctions of possibly negated *propositionally atomic formulas*. Here, a *propositionally atomic formula* is one whose the principal operator is not a logical connective. Propositionally atomic formulas include atomic formulas (among them equalities) and formulas whose the principal operator is a universal quantifier.

Since the acceptance of the original inference is equivalent to the refutation of the corresponding DNF, i.e. refutation of each disjunct separately, the task of the inference checker is to check inferences of the form

$$\frac{\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_k}{\lightning}$$

where the α_i are possibly negated propositionally atomic formulas.

The module EQUALIZER dealing with equational calculus [13] has to check such an inference. When some premises are equalities, computing their *congruence closure* can help in the checking process, where the congruence closure of a relation R defined on a set A is understood as a minimal *congruence relation* containing the original relation R. A relation S is called congruence, if it is an equivalence relation (it satisfies axioms of reflexivity, symmetry, and transitivity) and fulfills the axiom of *monotonicity* saying that for every n-ary operation f, if all pairs of arguments are in the relation S, then results are also in the relation, which we can express as

$$\left(\forall_{1\leq i\leq n} x_i S y_i\right) \rightarrow f(x_1, \ldots, x_n)\, S f(y_1, \ldots, y_n).$$

A congruence relation is then a monotonic equivalence relation.

A congruence is represented by its equivalence classes $[a]_R$, which are sets of all $b \in A$ such that aRb, that is

$$[a]_R = \{b : aRb\}.$$

The a is called the *representant* of the class. Two elements are called *congruent*, if they belong to the same equivalence class. Every equivalence relation defined on A divides it into disjoint nonempty equivalence classes, whose union gives the entire set A.

Effective algorithms computing congruence closure can be found in [5, 10, 11, 14].

Congruence closure can be utilized to check if an equality of two expressions is a consequence of a set of other equalities. An answer is based on the answer to the question, if the given equality belongs to the congruence closure. If it does, then the equality is a consequence, and if the equality is not a member of the congruence closure, it cannot be derived from the set of equalities.

For example, having equalities $y = z$, $f(y) = z$, and $f(z) = x$, one can infer that $x = y$. A possible scenario could be: since $y = z$ and $f(z) = x$, then $f(y) = x$.

Then, by symmetry and transitivity with $f(y) = z$ one knows that $x = z$ and again by symmetry and transitivity with $y = z$ one can conclude that $x = y$. In fact, all equalities among $x = y = z = f(y) = f(z)$ belong to the congruence closure.

Another use of the congruence closure used by CHECKER is to detect a contradiction. When the congruence closure is created and one of the following cases hold

- there are two premises of the form $P[x]$ and $\neg P[y]$ and x, y are congruent, or
- there is a premise of the form $x \neq y$ again when x, y are congruent

the contradiction is reported and the conjunction is refuted.

The input to EQUALIZER is a fixed collection of terms and all equalities accessible at the given stage of the proof. In principle, when the collection of terms has been built, it should never expand—EQUALIZER works on a closed set of terms named the *universe of discourse*. It protects the system from generating unexpected terms which could increase the amount of time needed to process an inference. For example, the associativity of multiplication $(x * y) * z = x * (y * z)$ applied to the term $a * (b * (c * d))$ would generate $(a * b) * (c * d)$ and $a * ((b * c) * d)$, and then $(a * (b * c)) * d$, etc.

The universe of discourse is internally represented as a collection of so called *cumulated terms*, that is a collection of equivalence classes initialized as trivial classes containing one term of the universe of discourse each. Complex terms are expressed in terms of representants of arguments, for example $a * b + c * d$ is stored as $E_1 + E_2$, where E_1 represents $a * b$ and E_2 represents $c * d$. EQUALIZER processes accessible equalities and when it finds an applicable one, it performs dedicated actions over adequate equivalence classes. The basic operation on equivalence classes is merging two classes. Roughly speaking, it joins two classes creating a new one, chooses its representant, and replaces occurrences of old representants of joined classes in all terms of the universe of discourse by the new one. The process is repeated until no merging can be done.

Mizar supports two kinds of proven sentences: theorems and *background knowledge* (called *properties* and *registrations*). To use a theorem one needs to refer to it explicitly (using `by`). Accessible background knowledge is used by CHECKER automatically. Only sentences of particular shapes (e.g. commutativity) qualify as background knowledge. Reading articles stored in the MIZAR MATHEMATICAL LIBRARY (MML) it was observed that quite many theorems contain statements about equality, and in the current version of MIZAR there is no way to promote them to become background knowledge.

Properties in MIZAR are special formulas, which can be registered while defining new unary or binary operations and used by EQUALIZER by default. Properties currently supported in the MIZAR language for unary operations are **involutiveness** and **projectivity** and for binary operations **commutativity** and **idempotence**, with their traditional meanings. Looking at structure of formulas corresponding to **involutiveness** $f(f(x)) = x$, **projectivity** $f(f(x)) = f(x)$, and **idempotence** $f(x, x) = x$, it can be seen that they may be thought of as rewriting rules [1] for which *the successor* (right hand side) is a subterm of *the predecessor* (left hand side).

This observation has led us to extending the MIZAR language and enhancing the CHECKER with a new mechanism working with terms called *reductions*, which increases the deductive power of the CHECKER.

The details of this mechanism are described in the next section.

2 Reductions

In the sequel, by a reduction we mean an equality for which the successor is a subterm of the predecessor. This section describes an experimental implementation inside Mizar of the proposed reduction mechanism.

2.1 Syntax

The syntax of reductions mimics the syntax of registrations with certain modifications.

Reductions can be introduced in Mizar texts as parts of registration blocks. They can be mixed with other constructions allowed in such blocks.

The grammar of reductions is:

```
Registration-Block =
 "registration"
  { Loci-Declaration | Cluster-Registration |
    Identify-Registration | Properties-Registration |
    Reduction-Registration | Canceled-Registration }
 "end" .

Reduction-Registration =
 "reduce" Term-Expression "to" Term-Expression ";"
 Correctness-Conditions .
```

The correctness condition, called `reducibility`, generated by the system is the equality of the two terms. In general a reduction (with pseudo-variables) could look like:

registration
 let x_1 **be** θ_1, x_2 **be** θ_2, ..., x_n **be** θ_n;
 reduce $\tau_1(x_1, x_2, \ldots, x_n)$ **to** $\tau_2(x_1, x_2, \ldots, x_n)$;
 reducibility
 proof
  thus $\tau_1(x_1, x_2, \ldots, x_n)$ **=** $\tau_2(x_1, x_2, \ldots, x_n)$;
 end;
end;

and a simple example taken from the Mizar Mathematical Library is:

```
registration
 let X be set, Y be Subset of X;
 reduce X /\ Y to Y;
 reducibility
 proof
  thus X /\ Y = Y;
 end;
end;
```

To preserve the rule that EQUALIZER only considers terms in the given universe of discourse, the successor has to be a subterm of the predecessor. A simple check of the structures of the two terms ensures that all components are already collected in the universe of discourse. As an example, the theorem

```
for F being Field, v being Vector of F holds v - v = 0.F;
```

cannot be presented as a reduction, because `0.F` is not a subterm of `v - v`.

Moreover, to avoid introducing trivial reductions, we require that the successor has to be a proper subterm of the predecessor.

In languages supporting hidden arguments of operations, the notion "subterm" can have two, slightly different, meanings, depending on whether or not hidden arguments count as subterms. Because the MIZAR language allows hidden arguments, we decided to regard hidden arguments as subterms. For example, the theorem saying that the image of the natural homomorphism generated by a normal subgroup of a given group is the quotient group, in the MML formulated as:

```
for G being Group, N being normal Subgroup of G holds
 Image nat_hom N = G./.N;
```

where `nat_hom N` is `Homomorphism of G,G./.N` can be presented as a reduction. In this case, `Image` is internally represented as *Image(G,N,G./.N, nat_hom(N))*, and can be simplified to `G./.N`.

The way to use reductions defined in other articles is to import them with the environment directive **registrations**.

2.2 Possible Errors

This section presents errors related to reductions reported by the MIZAR verifier. Standard errors related to the general syntax, accessibility of symbols, notions, loci, etc. are not listed here.

```
registration
 let X be set; let Y be Subset of X;
 reduce X /\ Y and Y;
::>                *404
 reducibility;
end;

::> 404: "to" expected
```

```
registration
 let X,Y be set;
 reduce X /\ Y to Y /\ X;
::>                   *257
 reducibility;
end;

::> 257 Right term must be a proper subterm of the left term
```

```
registration
 let S be empty 1-sorted;
 reduce the carrier of S to {};
::>                            *258
 reducibility;
end;
```

```
::> 258 Left term must be a standard term
```

In the above error explanation a standard term is a term of the form $F(\tau_1, \ldots, \tau_n)$, where F is a defined functor applied to terms, so, for example, selector terms like `the carrier of` are not allowed.

3 Experiments

The implemented software was tested on Mizar Version 7.12.01 working with MML Version 4.128.1067.

3.1 Detection

An important part of the reduction package is a tool (to be included in official distributions of the Mizar system to help authors to detect possible reductions during their work) which detects theorems stored in the MML, that could be rewritten as reductions. In the current version of the MML, 1579 cases were found. The Library Committee, who is responsible for management, developing and revisions of the MML, decided not to introduce all reductions detected as they are, but to analyze their complexity (how complex are predecessors and successors), usability, degree of obviousness, or whether there are better ways of formulating these facts available in Mizar.

3.2 Kinds of Reductions

Analyzing the list of detected theorems it was observed that they can be classified into several groups depending on different criteria based on the structure of terms involved in equalities.

3.2.1 Empty Set

The first group of theorems (106 cases) are reductions of some terms to the empty set. They are typically equalities where one of the arguments is the empty set, like `R | {} = {}` (a restriction of a relation to the empty set). An argument against introducing reductions for such facts is that they all can be formulated as functorial registrations—registrations saying that a term fulfills some properties—like `cluster R | {} -> empty`, which could also be processed by Analyzer (a modul of Checker responsible for type checking and reconstruction of hidden arguments) when processing adjectives in term types.

Table 1 Properties

Property	Number
`projectivity`	34
`involutiveness`	66
`idempotence`	39

3.2.2 Requirements-Like Reductions

Another type of possible reductions are facts processed by the mechanism called requirements, which are special modules implemented in the system for automatization of reasonings within particular theories, like arithmetic over complex numbers, [8]. An example of such reductions is complex multiplication by zero (`z * 0 = 0`, requirements `ARITHM`).

3.2.3 Properties-Like Reductions

In mathematics there are unary operations, say f, satisfying the property $f^m(x) = f^n(x)$. Such equalities fulfill the condition we imposed on reductions (one term is a subterm of another, when $m \neq n$). Some of them have commonly used names, like projectivity for $m = 2, n = 1$, that is $f(f(x)) = f(x)$; involutiveness when $m = 2, n = 0$, that is $f(f(x)) = x$; and automatic processing of these properties is already implemented in Mizar.

Table 1 presents numbers of theorems which could be rewritten as properties supported by Mizar.

In the MML there are also theorems about operations satisfying the property for other values of m and n, for example: `for x being Nat holds abs x = x`. They could be reductions, or it would be worth introducing other properties to the Mizar language, at least for small values of m and n.

4 Conclusions and Further Work

Analyzing possibilities offered by reductions, a couple of questions have arisen:

- Should Mizar still support properties `projectivity`, `involutiveness`, `idempotence`?
- Should other properties be introduced to the language?
- Which requirements should be rewritten as reductions, if any?
- How complex terms could be used in reductions?

Since reductions constitute a new construction, the Library Committee suggests not to remove existing features of the Mizar system and language, and postpone a decision giving authors the opportunity to use both and wait for the feedback from them.

Further development of reductions can go in different directions. Better integration with the congruence engine would allow application of more reductions. Furthermore development of other relations between predecessors and successors (other than being a subterm) would increase their usability.

Moreover, the Mizar Mathematical Library requires revisions to exploit reductions. Introducing new reductions to the already stored Mizar articles can possibly produce irrelevant references and unnecessary proof steps. Then, revisions of articles with tools like Relprem and Reliters [6, 9] should be performed.

Acknowledgement Special thanks to Andrzej Trybulec for his continuous support of my work.

References

1. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
2. Caminati, M.B.: Basic first-order model theory in Mizar. J. Form. Reason. **3**(1), 49–77 (2010)
3. Caminati, M.B.: Preliminaries to classical first-order model theory. Form. Math. **19**(3), 157–169 (2011)
4. Davis, M.: Obvious logical inferences. In: Proceedings of the Seventh International Joint Conference on Artificial Intelligence, pp. 530–531 (1981)
5. Downey, P.J., Sethi, R., Tarjan, R.E.: Variations on the common subexpression problem. J. ACM **27**, 758–771 (1980). doi:10.1145/322217.322228
6. Grabowski, A., Korniłowicz, A., Naumowicz, A.: Mizar in a nutshell. J. Form. Reason., Special Issue: User Tutorials I **3**(2), 153–245 (2010)
7. Korniłowicz, A.: How to define terms in Mizar effectively. In: Grabowski, A., Naumowicz, A. (eds.) Computer Reconstruction of the Body of Mathematics. Studies in Logic, Grammar and Rhetoric, vol. 18(31), pp. 67–77. University of Białystok (2009)
8. Naumowicz, A., Byliński, C.: Improving Mizar texts with properties and requirements. In: A. Asperti (ed.) MKM-2004. LNCS, vol. 3119, pp. 290–301. Springer, Berlin Heidelberg (2004)
9. Naumowicz, A., Korniłowicz, A.: A brief overview of Mizar. In: Berghofer, S., Nipkow, T., Urban, C., Wenzel, M. (eds.) Proc. 22nd International Conference, TPHOLs, Munich, Germany. LNCS, vol. 5674, pp. 67–72. Springer, Berlin Heidelberg (2009)
10. Nelson, G., Oppen, D.C.: Fast decision procedures based on congruence closure. J. ACM **27**, 356–364 (1980). doi:10.1145/322186.322198
11. Nieuwenhuis, R., Oliveras, A.: Proof-producing congruence closure. In: Giesl, J. (ed.) 16th International Conference on Term Rewriting and Applications, RTA'05. Lecture Notes in Computer Science, vol. 3467, pp. 453–468. Springer (2005)
12. Rudnicki, P.: Obvious inferences. J. Autom. Reasoning **3**(4), 383–393 (1987). doi:10.1007/BF00247436
13. Rudnicki, P., Trybulec, A.: Mathematical knowledge management in Mizar. In: Proc. of MKM 2001 (2001)
14. Shostak, R.E.: An algorithm for reasoning about equality. Commun. ACM **21**, 583–585 (1978). doi:10.1145/359545.359570