Formalizing a Fragment of Combinatorics on Words

Štěpán Holub[1(✉)] and Robert Veroff[2]

[1] Department of Algebra, Charles University, Prague, Czech Republic
holub@karlin.mff.cuni.cz
[2] Computer Science Department, University of New Mexico, Albuquerque, USA
veroff@cs.unm.edu

Abstract. We describe an attempt to formalize some tasks in combinatorics on words using the assistance of Prover9, an automated theorem prover for first-order and equational logic.

Keywords: Formalization · Periodicity · Combinatorics on words · Automated theorem proving

1 Motivation

In this paper we discuss a formalized approach to some tasks in combinatorics on finite words. Formalization of mathematical knowledge classically has two rather different motivations. One is Automated Theorem Proving, where one hopes to develop methods to find (possibly difficult, or just tedious) proofs automatically. The second motivation is Formalization of Mathematics, that aims at human-assisted computer verification of (human originated) parts of mathematics.

A prominent example of the formalization approach has evolved around the proof of Kepler's conjecture announced by Thomas Hales in 1998 and subsequently reviewed by 13 reviewers of *Annals of Mathematics* for three years without a conclusive verdict [10]. The situation in combinatorics on words is certainly less dramatic, but there are some similar features. As an illustrative example, we are looking at the classification of binary equality words [2,3,5,7,8]. In addition to important concepts, this project requires a lot of detailed case analysis which is arduous to make, tedious and unrewarding to read and check, and therefore possibly unreliable as to its correctness and/or completeness. Many of the arguments are repetitious. All of this leads to a conclusion that formalization might be a good idea. Moreover, the project is not yet completed, including more than two hundred undecided cases that may have to be dealt with separately. In view of this example, we want to keep in mind both possible goals of the formalization. One also can point out that artificial intelligence may blur the sharp distinction between them [14].

A third, tangential interest in this paper is to see what assumptions are needed in order to prove certain results. Looking at words through the lens of a limited set of tools yields interesting insights.

Š. Holub—Supported by the Czech Science Foundation grant number 13-01832S.

J. Kari et al. (Eds.): CiE 2017, LNCS 10307, pp. 24–31, 2017.
DOI: 10.1007/978-3-319-58741-7_3

2 Formalization

The proofs presented in this paper were found with the assistance of Prover9 [12], an automated theorem prover for first-order and equational logic. Problems are represented with a set of first-order formulas in clause form [1] that includes a set of axioms and a problem statement posed for proof by contradiction. Prover9 searches for a proof by applying inference rules to clauses until either a contradiction is found or some processing limit is reached. The search is guided by heuristics for selecting clauses, applying inference rules, and managing the growing set of derived clauses. Full details for the computation are accessible on the support web page [9].

Our formalization is based on several decisions that are motivated partly by theoretical considerations and partly by features of Prover9.

2.1 Semigroup

We consider words as *semigroup elements*. There is no explicit use of the fact that the semigroup is free; in particular, words are not seen as sequences. This is partly motivated by the fact that formal representations of lists typically are not handled well by theorem provers such as Prover9. Moreover, we are mainly interested in proofs showing that a certain relation on words forces periodicity, which is a property rather algebraic in nature. Nevertheless, the combinatorial complexity reappears in a nontrivial use of associativity; there are an exponential number of ways to associate any given expression.

We do not allow the empty word. Existence of the empty word has both advantages and disadvantages, and we decided to avoid it to simplify the language and theory.

2.2 No Arithmetic

We do not use natural numbers; in particular, we have no strong concept of length. The main reason for this is that we want to avoid computation and reasoning about inductively defined objects. Here too, the motivation is to avoid weaknesses of the theorem prover. An important consequence is that there is no uniform way to deal with arguments that typically would be inductive in nature. Length is partly substituted with a weaker *length comparison* compatible with the semigroup operation.

2.3 Equidivisibility

We assume the *equidivisibility* property of the semigroup: if $xy = uv$, then there is an element w such that either $x = uw$ and $v = wy$, or $u = xw$ and $y = wv$. This is a property of a free semigroup, also called Levi's lemma. Levi [11] proved that an equidivisible semigroup S is free if and only if it is graded, that is, if it is endowed with a semigroup homomorphism $\varphi : S \to (\mathbb{N}_+, +)$ (see also [13, p. 26]). Levi's lemma is thus a kind of measure of the distance of our axioms from the free semigroup.

2.4 Power

In addition to semigroup multiplication and length comparison, we use *power* as a primitive concept. The choice stresses that the main feature of words we are interested in is periodicity. Note that the concept of power becomes a nontrivial extension of multiplication precisely in the absence of natural numbers, since the expression w^n, understood as $w \cdot w \cdot \cdots \cdot w$, is an expression in a meta-language.

The properties of power that we formalized can be seen in the list of axioms below. We want to stress the axiom claiming that if both y and uyv are powers of x, then all the words u, y and v are powers of some common z. This is in a sense the only nontrivial fact about powers that we are using for now. (cf. Sect. 3).

Axioms

The above decisions lead to the following formal theory. The logical symbols in Prover9 notation are `&` for the logical *and*, `|` for the logical *or*, the minus sign for *negation*, and `!=` for *non-equal*. The existential quantifier is explicitly stated as ∃ (it has a verbal form `exists` in the computer code); non-quantified variables are implicitly universally quantified.

The non-logical symbols are

- binary operation `*`
- binary relations `Power` and `Shorter`

The standard interpretation of `Power(y,x)` is that y is a power of x; that is, $y \in x^+$. The standard interpretation of `Shorter(x,y)` is that x is shorter than y; that is, $|x| < |y|$.

The axioms are as follows:

```
(x * y) * z = x * (y * z).                          (associativity)
(x * y = x * z) -> y = z.                           (left cancellation)
(x * y = z * y) -> x = z.                           (right cancellation)
 x * y != x.                                        (no right unit)
 x * y != y.                                        (no left unit)
 x * y = u * v -> (x = u | ∃w (x * w = u | w * y = v)).
                                                    (equidivisibility)

-Shorter(x,x).                                      (non-reflexive)
Shorter(x,y) -> -Shorter(y,x).                      (anti-symmetric)
Shorter(x,y) & Shorter(y,z) -> Shorter(x,z).        (transitive)
Shorter(x,x * y).                                   (compatible with *)
Shorter(x,y * x).
Shorter(x,y) <-> Shorter(x * z,y * z).              (cancelation of length)
Shorter(x,y) <-> Shorter(z * x,z * y).
Shorter(x,y) <-> Shorter(x * z,z * y).
Shorter(x,y) <-> Shorter(z * x,y * z).
```

```
Power(x,x).                                          (reflexivity of power)
Power(x,y) & Power(y,z) -> Power(x,z).               (transitivity of power)
Power(y,x) & Power(z,x) -> Power(y * z,x).           (compatibility with *)
Power(y,x) & Power(z,x) -> ( y = z |  ∃u ( Power(u,x) &
 ( (y = z * u & y = u * z)|(z = y * u & z = u * y)) ) )
                                                     (cancellation of powers)
Power(y * z,x) -> (Power(y,x) & Power(z,x)) |
 (∃u ∃v (u * v = x &
       (y = u | ∃y1 (y = y1 * u & Power(y1,x)) &
       (z = v | ∃z1 (z = v * z1 & Power(z1,x)) ) )  (breaking a power)
(Power(y,x) & Power(u * (y * v), x)) ->
    ∃z (Power(y,z) & Power(u,z) & Power(v,z)).       (no nontrivial shift)
```

3 Commutation

In combinatorics on words, the fact that two words commute if and only if they are powers of the same root is probably the most elementary fact (up to considering the existence of the common root to be the very definition of commutation). In our formalization, however, the formula

```
x * y = y * x ->∃z(Power(x,z)&Power(y,z)).
```

is not an axiom but the first fact we would like to prove (let us call it the "Commutation lemma"). It turns out that this is an easy task for Prover9, hence we witness maybe the very first theorem in this field ever obtained by Automated Theorem Proving. Its "difficulty" at best corresponds to the characterization offered by Thomas Hales in 2008 [6, p. 1377]:

> Overall, the level today of fully automated computer proof [...] remains that of undergraduate homework exercises...

The proof output from Prover9 is given in Fig. 1. The constants `c1` and `c2` appearing in the proof are Skolem constants [1] coming from the existentially quantified variables in the negation of the theorem (for proof by contradiction).

The proof consists of a unique (humanly) nontrivial observation implied by the commutativity of x and y: $(xy)(xy) = x(yx)y = x(xy)y$. Now the consequent follows from the (no nontrivial shift) axiom.

4 Conjugation and Missing Induction

An obvious candidate claim to be proven next is the following theorem, actually just a part of a well known characterization of conjugate words.

Theorem 1. *If $xz = zy$, then there are words u and v such that $x = uv$ and $y = vu$.*

```
============================== PROOF =================================

% Proof 1 at 0.03 (+ 0.00) seconds.
% Length of proof is 21.
% Level of proof is 8.
% Maximum clause weight is 23.000.
% Given clauses 69 (8.625 givens/level).

5 Power(y,x) & Power(z,x) -> Power(y * z,x) # label(non_clause).  [assumption].
8 Power(y,x) & Power(u * (y * v),x) -> (exists z (Power(y,z) & Power(u,z) & Power(v,
z))) # label(non_clause).  [assumption].
29 x * y = y * x -> (exists z (Power(x,z) & Power(y,z))) # label(non_clause) #
label(goal).  [goal].
39 (x * y) * z = x * (y * z).  [assumption].
45 Power(x,x).  [assumption].
47 -Power(x,y) | -Power(z,y) | Power(x * z,y) # label(non_clause).  [clausify(5)].
64 -Power(x,y) | -Power(z * (x * u),y) | Power(z,f7(y,x,z,u)).  [clausify(8)].
65 -Power(x,y) | -Power(z * (x * u),y) | Power(u,f7(y,x,z,u)).  [clausify(8)].
89 c2 * c1 = c1 * c2.  [deny(29)].
90 c1 * c2 = c2 * c1 # label("Goal 1").  [copy(89),flip(a)].
91 -Power(c1,x) | -Power(c2,x).  [deny(29)].
97 -Power(x,y) | Power(x * x,y).  [factor(47,a,b)].
143 -Power(x * y,z) | -Power(u * (x * (y * w)),z) | Power(u,f7(z,x * y,u,w)).
[para(39(a,1),64(b,1,2))].
146 -Power(x * y,z) | -Power(u * (x * (y * w)),z) | Power(w,f7(z,x * y,u,w)).
[para(39(a,1),65(b,1,2))].
264 c1 * (c2 * x) = c2 * (c1 * x).  [para(90(a,1),39(a,1,1)),rewrite([39(4)]),flip(a)].
302 Power(x * x,x).  [resolve(97,a,45,a)].
325 Power(x * (y * (x * y)),x * y).  [para(39(a,1),302(a,1))].
398 Power(c2 * (c2 * (c1 * c1)),c2 * c1).  [para(90(a,1),325(a,1,2,2)),rewrite([264(7),
264(6),90(10)])].
431 Power(c2,f7(c2 * c1,c2 * c1,c2,c1)).  [resolve(143,b,398,a),unit_del(a,45)].
461 -Power(c1,f7(c2 * c1,c2 * c1,c2,c1)).  [ur(91,b,431,a)].
474 $F.  [resolve(146,b,398,a),unit_del(a,45),unit_del(b,461)].

============================== end of proof ==========================
```

Fig. 1. Proof of the Commutation lemma

Consider the following simple classical proof.

Proof. Proceed by induction on $|z|$. If $|z| \leq |x|$, then there is a word v such that $x = zv$ and $y = vz$. Therefore, we are done with $u = z$.

Assume that $|z| > |x|$. Then $z = xz' = z'y$ with $|z'| < |z|$, and the proof is completed by induction.

This proof cannot be formalized with our current axioms, since we do not have induction. Specifically, `Shorter` is not a well-founded relation.

In fact, the problem is not just with *this* proof, since the formula

$$\texttt{x * z = z * y -> (x = y) | } \exists \texttt{u} \exists \texttt{v (x = u * v \& y = v * u)} \quad \text{(conj.)}$$

cannot be proven with our axioms. This follows from the following semigroup in which all axioms hold, but Theorem 1 does not.

Let $\langle A \rangle$ be a semigroup generated by $A = \{a, b\} \cup \{c_i \mid i \in \mathbb{N}\}$ and defined by the set of relations $c_i = ac_{i+1} = c_{i+1}b$, $i \in \mathbb{N}$. The `Power` relation is interpreted in the natural way as in A^*. We define the "length" semigroup homomorphism $\ell : \langle A \rangle \to (\mathbb{Q}, +)$ by $\ell(a) = \ell(b) = 1$ and $\ell(c_i) = 2^{-i}$. The relation `Shorter(x,y)` is interpreted as $\ell(x) < \ell(y)$. Note that by Levi's lemma (see above), there cannot exist any semigroup homomorphism $\langle A \rangle \to (\mathbb{N}, +)$.

This example shows that our rudimentary axiomatic system must be extended by the (conj.) formula if we wish to prove anything about conjugate words.

5 Periodicity Lemma

The Periodicity lemma, often called the Fine and Wilf theorem [4], is a fundamental tool when dealing with periodicity. It states when a word can have two different periods p and q in a nontrivial way, where nontrivial means not having a period dividing both p and q. This formulation of the claim apparently depends strongly on arithmetical properties of periods, namely divisibility. Nevertheless, in this case we are able to prove the following version of the Periodicity lemma that is only slightly weaker than the full version:

Theorem 2. *Let u with $|u| \geq |xy|$ be a prefix of both x^ω and y^ω. Then x and y commute.*

For the sake of completeness, we recall that the full version of the Periodicity lemma has a weaker assumption $|u| \geq |xy| - \gcd(|x|, |y|)$, and moreover, it claims that the bound is optimal.

We have formulated Theorem 2 in a way that fits our formalization. Namely, periods of u are defined by its *periodic roots* x and y. Moreover, the infinite power is used, reminding us that we do not care about the exponent (since we haven't got the means needed). In order to make our formulas more intuitive, it may be convenient to enrich our language with binary relations `Prefix` and `Period`, defined by axioms

```
Prefix(x,y) <-> ∃z  (x * z = y).
Period(x,y) <-> ∃z  (Power(z,x)  &  Prefix(y,z)).
```

Theorem 2 now has the following simple form:

```
(Period (x,u) & Period(y,u) & -Shorter(u,x * y)) -> x * y = y * x.
```

An "informal" proof of Theorem 2 using only accepted axioms is the following.

Proof. Let uu_1 be a power of x and let uu_2 be a power of y. Then $uu_1 = xx_1$ and $uu_2 = yy_1$ with $x_1 \in x^+$ and $y_1 \in y^+$. Then $u = xu_3 = yu_4$, where $u_3u_1 \in x^+$ and $u_4u_2 \in y^+$. Since $uu_1, u_3u_1x \in x^+$ and $|uu_1| = |u_3u_1x|$, we deduce $uu_1 = u_3u_1x$. Now,

$$uu_1x = xu_3u_1x = xuu_1 = xyu_4u_1,$$

and xy is a prefix of u. Similarly, we obtain that yx is a prefix of u, which concludes the proof.

This is a typical example of a very simple proof which is at the same time quite unpleasant to read and verify. Of course, the same argument can be made with an appeal to the intuition of the character of the periodicity. However, such an intuition is hardly preserved throughout more complex proofs.

To date, Prover9 has not found a proof entirely on its own. To get a fully formalized proof, we split the argument into several steps. Specifically, we first proved four auxiliary lemmas:

```
(x * y = u * v) -> (Prefix(x,u) | Prefix(u,x) | x = u).          (L1)
Prefix(x,y) -> Shorter(x,y) .                                    (L2)
-Shorter(u, x * y) -> (u * z != x * y) .                         (L3)
((u * u1 = x * x1) & -Shorter(u,x * y)) -> (∃u3   (u = x * u3)) .  (L4)
```

They are just tiny, humanly natural reformulations of existing axioms and definitions which nevertheless help to point the Prover9 search in the right direction. The use of (L4) is clear from the reformulation of Theorem 2 below. Note that the lemma says: if u and x are prefixes of the same word and $|u| \geq |xy|$ (for an arbitrary y), then x is a prefix of u. The lemma would be more natural if $|x| < |u|$, that is `Shorter(x,u)`, were used instead of $|u| \geq |xy|$ (that is `-Shorter(u,x * y)`). However, the latter being an explicit assumption of Theorem 2, the present form is one more little hint for the automated proof.

We then reformulated the task as

```
(Power(u * u1, x) & u = x * u3 &
 Power(u * u2, y) & u = y * u4 &
 -Shorter(u,x * y) )
    -> x * y = y * x.
```

Here `∃u1 Power(u * u1, x)` can be proved, or it can be considered as a different definition of `Period(x,u)` (similarly for `Period(y,u)`). The claims `∃u3 u=x*u3` and `∃u4 u=y*u4` were proved separately.

The proof of Theorem 2 now splits into two cases. (1) $u = xy$ or $u = yx$; (2) $u \neq xy$ and $u \neq yx$. By symmetry of x and y, the first case can be reduced to $u = xy$. Note that this is a meta-argument sparing us one of two formal proofs identical up to exchange of x and y.

The case $u = xy$ was proved automatically when we suggested (L1) and (L2) to Prover9. For the case $u \neq xy$ and $u \neq yx$, we let Prover9 first prove intermediate conclusions (identical up to symmetry of x and y):

```
∃u5 (u=(x*y)*u5).
∃u5 (u=(y*x)*u6).
```

6 Conclusion

A text in combinatorics on words usually contains three dots (like $a_1 \cdots a_n$) somewhere on the first few lines. Experts on automated theorem proving quickly become skeptical when seeing those dots, since computers refuse to understand what they mean. The original intention of our research was therefore to break this skepticism and to show the very possibility of a formal approach to words. As in practically all other areas of mathematics, there is little hope (or fear) that computers will replace mathematicians in the near future. From our recent experience reported in this text, the realm of fully automated proving ends somewhere between the Commutation lemma and the Periodicity lemma. On the other hand, a vision of a computer assisted proof verification or search for individual steps in proofs seems more realistic. We wish to leave open for a further enquiry whether this or some modified formalization attempt can bring about something substantial.

References

1. Chang, C.-L., Lee, R.C.-T.: Symbolic Logic and Mechanical Theorem Proving. Academic Press, New York (1973)
2. Culik II, K., Karhumäki, J.: On the equality sets for homomorphisms on free monoids with two generators. RAIRO ITA 14(4), 349–369 (1980)
3. Czeizler, E., Holub, Š., Karhumäki, J., Laine, M.: Intricacies of simple word equations: An example. Int. J. Found. Comput. Sci. **18**(6), 1167–1175 (2007)
4. Fine, N.J., Wilf, H.S.: Uniqueness theorems for periodic functions. Proc. Am. Math. Soc. **16**(1), 109–109 (1965)
5. Hadravová, J.: Structure of equality sets. Ph.D. thesis, Charles University (2007)
6. Hales, T.C.: Formal proof. Not. AMS **55**(11), 1370–1380 (2008)
7. Holub, Š.: Binary equality sets are generated by two words. J. Algebra **259**(1), 1–42 (2003)
8. Holub, Š.: A unique structure of two-generated binary equality sets. In: Ito, M., Toyama, M. (eds.) DLT 2002. LNCS, vol. 2450, pp. 245–257. Springer, Heidelberg (2003). doi:10.1007/3-540-45005-X_21
9. Holub, Š., Veroff, R.: Formalizing a fragment of combinatorics on words (web support) (2017). http://www.cs.unm.edu/veroff/CiE2017/
10. Lagarias, J.C.: The Kepler Conjecture: The Hales-Ferguson Proof. Springer, New York (2011)
11. Levi, F.W.: On semigroups. Bull. Calcutta Math. Soc. **36**, 141–146 (1944)
12. McCune, W.: Prover9, version 02a (2009). http://www.cs.unm.edu/mccune/prover9/
13. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press, New York (2009)
14. Urban, J., Vyskočil, J.: Theorem proving in large formal mathematics as an emerging AI field. In: Bonacina, M.P., Stickel, M.E. (eds.) Automated Reasoning and Mathematics. LNCS, vol. 7788, pp. 240–257. Springer, Heidelberg (2013). doi:10.1007/978-3-642-36675-8_13