Formal verification of the correctness of chosen algorithms in Mizar

Adrian Jaszczak

Institute of Informatics
University of Bialystok, Poland
a.jaszczak@uwb.edu.pl

Abstract

In this paper we continue development of formal verification tools for algorithms using the framework of nominative data and Floyd-Hoare logic with partial pre- and postconditions in the Mizar proof assistant. We define operations of sequential composition of several programs, formalize and show soundness of new inference rules which can be used to prove partial correctness of programs involving these operations in the context of partial pre- and postconditions. A process of verification of the partial correctness of exemplary algorithms is described.

1 Introduction

We live in times of very fast changes. Every day we can see how computer programs and any kind of applications affect our life more and more. Architects of buildings, designers of cars decide to take part of control from humans and give it to machines and A.I. Human can make a mistake and computer – assuming that its algorithms are written correctly – can not. Therefore researchers in computer science are interested in creating tools allowing us to verify the correctness of those algorithms.

There are three major formal semantics which can be helpful in creating abstract models of algorithms: operational, denotational and axiomatic [NN92]. Each of them have different fundamental ideas. First one focuses on *how* to produce effect of a computation. Second one leaves behind *how* and describes only the effect, and the third one describes specific properties of the execution in *assertions*. Everything is done on a high level of abstraction, in isolation from any specific computer model.

Many approaches to formal verification of properties of programs and algorithms based on different logics and algebra systems have been developed. Researchers were trying to achieve the best formal model – some of them wanted to invent logic that would allow partiality of programs, which is very common, and others were trying to eliminate it and consider only total functions and predicates.

As an example, we can consider a logical framework for formal verification of programs using an extended Floyd-Hoare-style rules with pre- and postconditions and loop invariants defined by partial predicates [KNS13]. This framework consists of: a) a two-sorted program algebra (an extension of Glushkov algebra [Glu65]) over partial predicates and binominative functions on nominative data [SNI14] (programs and assertions are written as terms of the algebra); and b) an inference rules system based on classical Floyd-Hoare logic [Flo67, Hoa69] with new rules that allow reasoning over partial pre- and postconditions.

In: C. Kaliszyk, E. Brady, J. Davenport, W.M. Farmer, A. Kohlhase, M. Kohlhase, D. Müller, K. Pąk, and C. Sacerdoti Coen (eds.): Joint Proceedings of the FMM and LML Workshops, Doctoral Program and Work in Progress at the Conference on Intelligent Computer Mathematics 2019 co-located with the 12th Conference on Intelligent Computer Mathematics (CICM 2019), Prague, Czech Republic, July 8–12, 2019, published at http://ceur-ws.org

The framework is intended to be used for practical verification of the correctness of programs. One of its implementations is done in the Mizar proof assistant [BBG+15, GKN15]. Nominative data with simple names and complex values and basic operations on them are defined in [INKK17]. An algebra of programs over partial predicates and binominative functions on nominative data including among others operators of a programming language (e.g. *assignment*, *skip*, conditional *if* statement and a *while* loop) are defined in [KIN18, IKN18b, IKN18c]. An inference system of an extension of Floyd-Hoare logic for partial predicates and soundness of the rules are formalized in [IKN18a]. An example – the subtraction-based version of Euclid's algorithm computing the greatest common divisor of natural numbers – how to use all this stuff is presented in [IKN18d].

In this paper we report on our continuation [KKNI17a, KKNI17b, KKNI17c] of formalization of the framework in Mizar including three more inference rules for unconditional composition of 3, 4, and 5 programs and proofs of the correctness of two algorithms: computing the natural power of a complex number and factorial of a natural number.

2 Mizar Formalization

Having mathematical tools mentioned above we have proved in Mizar the correctness of two algorithms: the natural power of a complex number [Jas19] and factorial of a natural number [JK19] with the following pseudocodes:

```
Pseudocode of factorial algorithm:
i := val.1; j := val.2;
n := val.3; s := val.4;
while ( i <> n )
  i := i + j;
  s := s * i;
return s;
```

```
Pseudocode of power algorithm:
i := val.1; j := val.2;
b := val.3; n := val.4; s := val.5;
while ( i <> n )
  i := i + j;
  s := s * b;
return s;
```

Both algorithms consist of three parts: the initialization of variables, the main `while` loop, and the result returning statement. Moreover, in factorial we have only one input value and in power two input values: base and exponent. At the end we return `s` as the results of algorithms.

Formalization of the algorithms in Mizar has been divided into few separate parts. Because both algorithms are similar, we can take a closer look into just one of them, let us choose the power algorithm. To formulate it we introduced several Mizar functors representing various components of the algorithm: the initialization of variables:

```
definition let V,A,loc,val;
  func power_var_init(A,loc,val) -> SCBinominativeFunction of V,A equals
   PP_composition(
      SC_assignment(denaming(V,A,val.1), loc/.1), SC_assignment(denaming(V,A,val.2), loc/.2),
      SC_assignment(denaming(V,A,val.3), loc/.3), SC_assignment(denaming(V,A,val.4), loc/.4),
      SC_assignment(denaming(V,A,val.5), loc/.5) );
end;
```

the loop body:

```
definition let V,A,loc;
  func power_loop_body(A,loc) -> SCBinominativeFunction of V,A equals
    PP_composition(
      SC_assignment(addition(A,loc/.1,loc/.2),loc/.1),
      SC_assignment(multiplication(A,loc/.5,loc/.3),loc/.5) );
end;
```

the entire loop:

```
definition let V,A,loc;
  func power_main_loop(A,loc) -> SCBinominativeFunction of V,A equals
  PP_while(PP_not(Equality(A,loc/.1,loc/.4)),power_loop_body(A,loc));
end;
```

the initialization composed with the loop:

```
definition let V,A,loc,val;
  func power_main_part(A,loc,val) -> SCBinominativeFunction of V,A equals
    PP_composition(power_var_init(A,loc,val),power_main_loop(A,loc));
end;
```

and finally the entire program where the returning statement is added:

```
definition let V,A,loc,val,z;
  func power_program(A,loc,val,z) -> SCBinominativeFunction of V,A equals
    PP_composition(power_main_part(A,loc,val), SC_assignment(denaming(V,A,loc/.5),z));
end;
```

Because we work in the paradigm of nominative data those definitions take sets `V` and `A` as names and values, respectively. Moreover, `V-valued` functions `loc` represents formally locations in memory where variables are stored, and functions `val` keep values of variables.

The partial correctness of programs in our framework is expressed as the validity of semantic Floyd-Hoare triples [IKN18a] of the form `<*p,f,r*> is SFHT of D`, where `p` represents a precondition, `f` represents a program, and `r` represents a postcondition, all defined over a set `D`. In the case of the power algorithm it takes the form:

```
<* valid_power_input(V,A,val,b0,n0),
   power_program(A,loc,val,z),
   valid_power_output(A,z,b0,n0) *> is SFHT of ND(V,A)
```

where `valid_power_input` is the precondition representing the valid input and `valid_power_output` is the postcondition representing the valid output, both involving the complex base `b0` and the natural exponent `n0`. All components of the triple are defined over the set `ND(V,A)` of all nominative data over sets `V` and `A`.

The discussed algorithm contains a `while` loop, which verification requires an invariant. For this purpose we defined the predicate:

```
definition let V,A,loc,b0,n0,d;
  pred power_inv_pred A,loc,b0,n0,d means
  ex d1 being NonatomicND of V,A st
   d = d1 & { loc/.1, loc/.2, loc/.3, loc/.4, loc/.5 } c= dom d1 &
   d1.(loc/.2) = 1 & d1.(loc/.3) = b0 & d1.(loc/.4) = n0 &
   ex S being Complex, I being Nat st I = d1.(loc/.1) & S = d1.(loc/.5) & S = b0|^I;
end;
```

which describes that every state `d1` includes all required memory locations, initial values 1, `b0` and `n0` are always in the same locations, and `S` is always equal to $\mathtt{b0}^{\mathtt{I}}$. It is used to prove that initialization of variables and each iteration of the loop fulfill the condition.

With this structure we could start proving the correctness of the algorithms. Detailed proofs are available in the Mizar Mathematical Library [BBG+18, Jas19, JK19].

Apart from the formal verification of the correctness of mentioned algorithms we also defined operations for composition of 3, 4 and 5 instructions. Moreover, we defined inference rules describing how to prove the correctness of programs involving these operations and proved their soundness. In the case of the composition of 3 instructions the rule is:

$$\frac{\{p\}\ f_1\ \{q\},\ \{q\}\ f_2\ \{r\},\ \{\sim q\}\ f_2\ \{r\},\ \{r\}\ f_3\ \{s\},\ \{\sim r\}\ f_3\ \{s\}}{\{p\}\ f_1 \bullet f_2 \bullet f_3\ \{s\}}$$

As one can notice, the rule above is different than in the standard Floyd-Hoare logic would be. The reason of that is because we also consider a case where results of programs f_1 and f_2 do not belong to the domains of the partial conditions q and r, respectively. In the standard case all pre- and postconditions would be total.

The operations and rules allowed us to use one composition instead of several binary compositions.

3 Conclusions and Future Work

In this paper we have shown how to verify the correctness of algorithms in the Mizar proof assistant using nominative data and a variant of Floyd-Hoare logic on the example of algorithms computing the natural power of a complex number and factorial of a natural number. On these simple examples it can be observed that various algorithms have almost same structure and in many cases there are very similar formalization steps. In the future we want to detect and define more general structures of algorithms written in our environment and to use particular Mizar constructions, like structures and schemes, to formulate the algorithms and make reasoning on them. For example, we can define a Mizar structure containing the input, output, constant values, the main program, and possibly other components of programs as separate fields of the structure.

Another way of extension of our framework is to define new instructions in the language and new inference rules about the instructions within our variant of logic. For example, we can introduce an instruction for the composition of arbitrary n instructions instead of compositions of two, three, four and more instructions separately, and `for` loop instruction and adequate inference rules. It will make easier writing algorithms with sequences of compositions and make shorter proofs of properties of the algorithms.

The ultimate goal of the project is to build a complete formal tool for verifying the correctness of complex algorithms. These algorithms could be then implemented, for example, in some safety-critical software. Soon, almost every aspect of our life will be connected with computer programs, so it is important to make sure that algorithms that they will be using are designed and implemented correctly.

References

[BBG+15] Grzegorz Bancerek, Czesław Byliński, Adam Grabowski, Artur Korniłowicz, Roman Matuszewski, Adam Naumowicz, Karol Pąk, and Josef Urban. Mizar: State-of-the-art and beyond. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge, editors, *Intelligent Computer Mathematics*, volume 9150 of *Lecture Notes in Computer Science*, pages 261–279. Springer International Publishing, 2015. `http://dx.doi.org/10.1007/978-3-319-20615-8_17`

[BBG+18] Grzegorz Bancerek, Czesław Byliński, Adam Grabowski, Artur Korniłowicz, Roman Matuszewski, Adam Naumowicz, and Karol Pąk. The role of the Mizar Mathematical Library for interactive proof development in Mizar. *Journal of Automated Reasoning*, 61(1):9–32, 2018. `https://doi.org/10.1007/s10817-017-9440-6`

[Flo67] R.W. Floyd. Assigning meanings to programs. *Mathematical aspects of computer science*, 19(19–32), 1967.

[GKN15] Adam Grabowski, Artur Korniłowicz, and Adam Naumowicz. Four decades of Mizar. *Journal of Automated Reasoning*, 55(3):191–198, 2015. `https://doi.org/10.1007/s10817-015-9345-1`

[Glu65] V. M. Glushkov. Automata theory and formal microprogram transformations. *Cybernetics*, 1(5):1–8, Sep 1965.

[Hoa69] C.A.R. Hoare. An axiomatic basis for computer programming. *Commun. ACM*, 12(10):576–580, 1969.

[IKN18a] Ievgen Ivanov, Artur Korniłowicz, and Mykola Nikitchenko. An inference system of an extension of Floyd-Hoare logic for partial predicates. *Formalized Mathematics*, 26(**2**):159–164, 2018. `https://doi.org/10.2478/forma-2018-0013`

[IKN18b] Ievgen Ivanov, Artur Korniłowicz, and Mykola Nikitchenko. On algebras of algorithms and specifications over uninterpreted data. *Formalized Mathematics*, 26(**2**):141–147, 2018. `https://doi.org/10.2478/forma-2018-0011`

[IKN18c] Ievgen Ivanov, Artur Korniłowicz, and Mykola Nikitchenko. On an algorithmic algebra over simple-named complex-valued nominative data. *Formalized Mathematics*, 26(**2**):149–158, 2018. `https://doi.org/10.2478/forma-2018-0012`

[IKN18d] Ievgen Ivanov, Artur Korniłowicz, and Mykola Nikitchenko. Partial correctness of GCD algorithm. *Formalized Mathematics*, 26(**2**):165–173, 2018. `https://doi.org/10.2478/forma-2018-0014`

[INKK17] Ievgen Ivanov, Mykola Nikitchenko, Andrii Kryvolap, and Artur Korniłowicz. Simple-named complex-valued nominative data – definition and basic operations. *Formalized Mathematics*, 25(**3**):205–216, 2017. `https://doi.org/10.1515/forma-2017-0020`

[Jas19] Adrian Jaszczak. Partial correctness of a power algorithm. *Formalized Mathematics*, 27(**2**), 2019. Accepted for publication.

[JK19] Adrian Jaszczak and Artur Korniłowicz. Partial correctness of a factorial algorithm. *Formalized Mathematics*, 27(**2**), 2019. Accepted for publication.

[KIN18] Artur Korniłowicz, Ievgen Ivanov, and Mykola Nikitchenko. Kleene algebra of partial predicates. *Formalized Mathematics*, 26(**1**):11–20, 2018. `https://doi.org/10.2478/forma-2018-0002`

[KKNI17a] Artur Korniłowicz, Andrii Kryvolap, Mykola Nikitchenko, and Ievgen Ivanov. An approach to formalization of an extension of Floyd-Hoare logic. In Vadim Ermolayev, Nick Bassiliades, Hans-Georg Fill, Vitaliy Yakovyna, Heinrich C. Mayr, Vyacheslav Kharchenko, Vladimir Peschanenko, Mariya Shyshkina, Mykola Nikitchenko, and Aleksander Spivakovsky, editors, *Proceedings of the 13th International Conference on ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer, Kyiv, Ukraine, May 15–18, 2017*, volume 1844 of *CEUR Workshop Proceedings*, pages 504–523. CEUR-WS.org, 2017. `http://ceur-ws.org/Vol-1844/10000504.pdf`

[KKNI17b] Artur Korniłowicz, Andrii Kryvolap, Mykola Nikitchenko, and Ievgen Ivanov. Formalization of the algebra of nominative data in Mizar. In Maria Ganzha, Leszek A. Maciaszek, and Marcin Paprzycki, editors, *Proceedings of the 2017 Federated Conference on Computer Science and Information Systems, FedCSIS 2017, Prague, Czech Republic, September 3–6, 2017*, pages 237–244, 2017. `https://doi.org/10.15439/2017F301`

[KKNI17c] Artur Korniłowicz, Andrii Kryvolap, Mykola Nikitchenko, and Ievgen Ivanov. Formalization of the nominative algorithmic algebra in Mizar. In Leszek Borzemski, Jerzy Świątek, and Zofia Wilimowska, editors, *Information Systems Architecture and Technology: Proceedings of 38th International Conference on Information Systems Architecture and Technology – ISAT 2017 – Part II, Szklarska Poręba, Poland, September 17–19, 2017*, volume 656 of *Advances in Intelligent Systems and Computing*, pages 176–186. Springer, 2017. `https://doi.org/10.1007/978-3-319-67229-8_16`

[KNS13] Andrii Kryvolap, Mykola Nikitchenko, and Wolfgang Schreiner. Extending Floyd-Hoare logic for partial pre- and postconditions. In Vadim Ermolayev, Heinrich C. Mayr, Mykola Nikitchenko, Aleksander Spivakovsky, and Grygoriy Zholtkevych, editors, *Information and Communication Technologies in Education, Research, and Industrial Applications: 9th International Conference, ICTERI 2013, Kherson, Ukraine, June 19–22, 2013, Revised Selected Papers*, pages 355–378. Springer International Publishing, 2013. `https://doi.org/10.1007/978-3-319-03998-5_18`

[NN92] Hanne Riis Nielson and Flemming Nielson. *Semantics with Applications: A Formal Introduction.* John Wiley & Sons, Inc., New York, NY, USA, 1992.

[SNI14] Volodymyr G. Skobelev, Mykola Nikitchenko, and Ievgen Ivanov. On algebraic properties of nominative data and functions. In Vadim Ermolayev, Heinrich C. Mayr, Mykola Nikitchenko, Aleksander Spivakovsky, and Grygoriy Zholtkevych, editors, *Information and Communication Technologies in Education, Research, and Industrial Applications – 10th International Conference, ICTERI 2014, Kherson, Ukraine, June 9–12, 2014, Revised Selected Papers*, volume 469 of *Communications in Computer and Information Science*, pages 117–138. Springer, 2014. `https://doi.org/10.1007/978-3-319-13206-8_6`