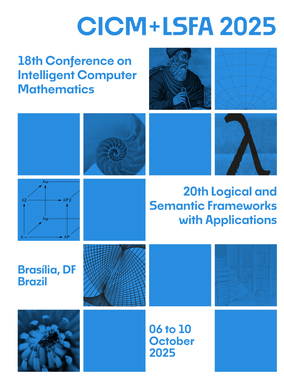
Formalizing an Algorithm in PVS for Anti-unification modulo $(\mathtt{aC})(\mathtt{a})(\mathtt{C})$

Marcos Mercandeli-Rodrigues

Universidade de Brasília

m.m.rodrigues@mat.unb.br



Introduction

An anti-unification problem for a family of objects of a certain type in a certain context consists in finding commonalities between those objects in order to construct generalizers that uniformly abstract over their differences subject to a preference relation for comparing them according to what constitutes a good generalization relative to the context (Cerna and Kutsia [2023]). Anti-unification is a vital component in mathematical formalisms regarding different deductive frameworks such as proof assistants, systems for numerical, symbolic and algebraic computations, automatic deduction, and automation (Cerna and Kutsia [2023]), to cite a few.

Considering those applications and the development of the field of equational reasoning, it is pertinent to formalize, in interactive theorem provers, the mathematical properties ensuring the termination, soundness, and completeness of computational solutions for the anti-unification problem modulo various algebraic theories. The only formalization of an anti-unification algorithm known to date is due to Ayala-Rincón et al. [2025] for syntactic anti-unification. The algorithm `Antiunify` (Ayala-Rincón et al. [2025]) presented in Figure 1, together with its termination and soundness, was formalized and verified in PVS (Prototype Verification System, Owre et al. [1992]).

$$\text{(Decompose-Function)}\ \frac{\langle \{f\, s \underset{X}{\triangleq} f\, t\} \cup U' \mid S \mid \sigma\rangle}{\langle \{s \underset{Y}{\triangleq} t\} \cup U' \mid S \mid (X \mapsto f\, Y)\,\sigma\rangle}$$

$$\text{(Decompose-Pair)}\ \frac{\langle \{(s,u) \underset{X}{\triangleq} (t,v)\} \cup U' \mid S \mid \sigma\rangle}{\langle \{s \underset{Y}{\triangleq} t, u \underset{Z}{\triangleq} v\} \cup U' \mid S \mid (X \mapsto (Y,Z))\,\sigma\rangle}$$

$$\text{(Solve-Repeated)}\ \frac{\langle \{s \underset{X}{\triangleq} t\} \cup U' \mid S \mid \sigma\rangle}{\langle U' \mid S \mid (X \mapsto X')\,\sigma\rangle}\ \text{if } s \underset{X}{\triangleq} t \text{ is solved and } s \underset{X'}{\triangleq} t \in S$$

$$\text{(Solve-Non-Repeated)}\ \frac{\langle \{s \underset{X}{\triangleq} t\} \cup U' \mid S \mid \sigma\rangle}{\langle U' \mid \{s \underset{X}{\triangleq} t\} \cup S \mid \sigma\rangle}\ \text{if } s \underset{X}{\triangleq} t \text{ is solved and non-repeated in } S$$

$$\text{(Syntactic)}\ \frac{\langle \{s \underset{X}{\triangleq} s\} \cup U' \mid S \mid \sigma\rangle}{\langle U' \mid S \mid (X \mapsto s)\,\sigma\rangle}\ \text{if } s \underset{X}{\triangleq} s \text{ is trivial}$$

Figure 1: Rules of the algorithm Antiunify (Ayala-Rincón et al. [2025]).

Although useful, algorithms for syntactic anti-unification alone cannot provide all the preferred solutions for the instances present in equational theories such as the commutative (`C`), associative (`A`) (Alpuente et al. [2014]), idempotent (`I`) (Cerna and Kutsia [2020b]), unital (`U`) (Cerna and Kutsia [2020a]), absorptive ($\mathtt{a}$) (Ayala-Rincón et al. [2024]) theories as well as their combinations. A very favourable aspect of the algorithm's construction and specification in PVS developed by Ayala-Rincón et al. [2025] is that the work done so far provides a very solid ground upon which algorithms for solving the anti-unification problems in first-order equational theories of extensive use can also be specified and formally mechanized.

Aims and Questions

Since March 2025, I have been working with Professor Mauricio Ayala-Rincón, Professor Temur Kutsia, Dr. Mariano Moscato, Professor Thaynara Arielly de Lima, and Maria Julia Dias Lima on the completeness proof of the Antiunify algorithm. Such work dealt also with certain conjectures proved for substitutions and renamings. For the PhD thesis, the main goal is the extension of the algorithm Antiunify to a complete and sound algorithm for dealing with anti-unificatiom modulo $(\mathtt{aC})(\mathtt{a})(\mathtt{C})$. That extension and its mathematical properties will be also specified and verified in PVS. Important issues regarding the extension of the current specification in PVS are the following:

1. The specifications for `first-order terms` (Figure 2) and `first-order substitution` must be extended in order to provide specifications for terms and substitutions modulo commutativity and absorption;
2. The notion of `configuration` must be extended in order to deal with delayed sets of AUTs (Ayala-Rincón et al. [2024]);
3. The algorithm Antiunify for syntactic anti-unification must be extended to a sound and complete algorithm for anti-unification modulo $(\mathtt{aC})(\mathtt{a})(\mathtt{C})$.

Regarding those issues, some important questions are the following:

a. How can such extensions be constructed in PVS in a way that preserves the previously developed proofs of results that are independent of the axioms of that equational theory?

b. Regarding algorithms, how can such extensions be constructed in PVS in order to preserve certain branches of previously developed proofs for rule applications that are independent of the axioms of that equational theory?

Example

For a concrete example, let us consider the specifications for `first-order terms` presented in Figure 2. In order to deal with absorptive-commutative theories, we must deal with the axioms governing such theories.

```
first_order_term[constant:TYPE, variable:TYPE+, f_symbol:TYPE]: DATATYPE
    BEGIN
        const (a: constant): const?
        variable (V: variable): var?
        unit: unit?
        pair (term1: first_order_term, term2: first_order_term): pair?
        app (f_sym: f_symbol, arg: first_order_term): app?
    END first_order_term
```

Figure 2: Abstract datatype for first order terms (Ayala-Rincón et al. [2025]).

If we add more structure to the abstract data type, then new proof obligations will occur inside proofs that do not depend on such algebraic properties (for instance, proofs of results concerning the action of substitutions on terms as syntactic objects). That is undesirable.

```
AUT: TYPE = [# lhs, rhs : Term , label : variable #]

Configuration: TYPE = [# unsolved, solved: List_AUT, substitution: (nice?) #]
```

Figure 3: Specifications of configurations and anti-unification triples in PVS (Ayala-Rincón et al. [2025]).

Another important change will happen in the specification of configuration in order to accommodate delayed sets of AUTs (Ayala-Rincón et al. [2024]), which play an important role in the case of anti-unification in absorptive theories. See Figure 3 for their specifications in PVS. Considering the approach used by Ayala-Rincón et al. [2024] for solving the problem in absorptive theories, the AUTs must also be adapted in order to accommodate wild cards, which in turn will require further modifications on the specifications of first-order terms.

Partial Results

We are working on the proof of completeness for the algorithm Antiunify (see Figure 4). The proof of completeness is divided into four lemmata, each dealing with a rule:

- The branch dealing with the rule `Decompose-Function` is already closed;
- The branch dealing with the rule `Decompose-Pair` is similar to the previous one;
- We are currently proving the `Solve-Repeated` and `Solve-Non-Repeated` branches.

The proof of completeness for the syntactic algorithm is an important step for the main goal, not just because of completeness itself, but also for the insights on how to deal with rules that will not be changed during the course of the specification of an algorithm for dealing with absorptive-commutative theories (for instance, the rule `Syntactic`).

```
antiunify(c: (validConfiguration?)): recursive (validConfiguration?)
 =  IF cons?(c`unsolved)
    THEN
      IF match_DecF_conf?(c)
      THEN      % Apply "Decompose-Function" rule
           antiunify(DecF(c))
      ELSIF match_DecP_conf?(c)
      THEN      % Apply "Decompose-Pairs" rule
           antiunify(DecP(c))
      ELSIF match_Synt_conf?(c)
      THEN      % Apply "Syntactic" rule
           antiunify(Synt(c))
      ELSE      % Apply "Solve" rule
           antiunify(Solve(c))
      ENDIF
    ELSE c
    ENDIF
 MEASURE size(c`unsolved)
```

Figure 4: Algorithm Antiunify in PVS (Ayala-Rincón et al. [2025]).

Conclusion and Future Work

- We are currently working on the proof of completeness for the algorithm Antiunify. We closed the branches dealing with decomposition rules;
- We must extend the specifications of first-order terms, first-order substitution, AUTs, and configurations in a conservative way in order to avoid undesirable proof obligations.
- Rules must be extended in order to deal with absorptive-commutative theories: For instance, the rules Solve-Repeated and Solve-Non-Repeated must be able do deal with equality modulo absorptive-commutative theories.

References

M. Alpuente, S. Escobar, J. Espert, and J. Meseguer. A modular order-sorted equational generalization algorithm. *Inf. Comput.*, 235:98–136, 2014.

M. Ayala-Rincón, D. M. Cerna, A. F. G. Barragán, and T. Kutsia. Equational anti-unification over absorption theories. In C. Benzmüller, M. J.H. Heule, and R. A. Schmidt, editors, *Automated Reasoning*, pages 317–337, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-63501-4.

M. Ayala-Rincón, T. A. de Lima, M. J. D. Lima, M. M. Moscato, and T. Kutsia. Verification of an anti-unification algorithm in PVS. In A. Dutle, L. Humphrey, and L. Titolo, editors, *NASA Formal Methods*, pages 54–71, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-93706-4.

D. M. Cerna and T. Kutsia. Unital anti-unification: Type algorithms. *5th International Conference on Formal Structures for Computation and Deduction, FSCD*, 167(6):26:1–26:20, 2020a. URL `https://doi.org/10.1017/S0960129520000110`.

D. M. Cerna and T. Kutsia. Idempotent anti-unification. *ACM Trans. Comput. Log.*, 21 (2):10:1–10:32, 2020b.

D. M. Cerna and T. Kutsia. Anti-unification and generalization: A survey. In *Proceedings of the 32nd Int. Joint Conference on Artificial Intelligence, IJCAI*, pages 6563–6573. ijcai.org, 2023. URL `https://doi.org/10.24963/ijcai.2023/736`.

S. Owre, J. M. Rushby, and N. Shankar. PVS: A prototype verification system. In D. Kapur, editor, *11thInt. Conference on Automated Deduction (CADE)*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752, Saratoga, NY, June 1992. Springer.

Acknowledgements

This study was financed in part by the Coordenação de Aprefeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

About the Author

Marcos Mercandeli Rodrigues is a PhD candidate in Mathematics at Universidade de Brasília working under the supervision of Professor Mauricio Ayala-Rincón (supervisor, UnB) and Professor Temur Kutsia (cosupervisor, RISC/JKU).