Formalisation of Nominal Equational Reasoning

Mauricio Ayala-Rincón

Formalisation of Nominal Equational Reasoning* (Invited Talk)

Mauricio Ayala-Rincón[1]

Universidade de Brasília - Brasília D.F., Brasil
ayala@unb.br

Abstract

In contrast to other methods to bind variables, the nominal approach uses atoms and the algebra of atom permutations bringing enlightening implications. This talk will discuss the lessons learned from our experience towards formalising nominal matching and nominal unification modulo C and AC using the proof assistant PVS (Prototype Verification System). Furthermore, the talk will dissect exciting issues that appear during the formalisation of nominal equational reasoning giving rise to surprises regarding the well-known properties of first-order unification and unification modulo C and AC.

1 Nominal reasoning

In nominal syntax [Pit13], terms include function symbols, abstractions, and two kinds of variables: *atoms* and *unknowns* (or simply variables). *Atoms* represent object-level variables whereas *unknowns* behave like first-order variables, except that they can have "suspended atom permutations" which act when variables are instantiated. Atoms can be abstracted over terms; the nominal term $[a]s$ represents the abstraction of a in s, and α-equivalence is axiomatised through a freshness relation $a\#t$ (read: a is fresh in t). Furthermore, name-swapping $(a\,b)$, seen as a transposition, implements atom permutation and, thus, atom renaming.

For example, the first-order logic formula $\exists a.a < 0$ can be written as the nominal term $\exists([a]\mathtt{lt}(a,0))$, using function symbols $\exists$ and $\mathtt{lt}$ and an abstracted atom a. Notice that $\exists([a]\mathtt{lt}(\mathtt{a},\mathtt{0})) \approx_\alpha^? \exists([b]\mathtt{lt}(Y,0))$ is a nominal unification problem whose solution should be such that a does not occur free in Y.

The next sections discuss the long-term project of formalising nominal equational reasoning in the proof assistant PVS [OS99].

2 Earlier formalisations of nominal reasoning

Nominal unification was proved and formalised decidable in Isabelle/HOL by Christian Urban et al. ([UPG04, Urb10]), and efficient nominal unification algorithms are available (e.g. [CF08, LV10]) that compute solutions consisting of freshness contexts (containing freshness constraints of the form $a\#X$) and substitutions. Nominal unification was also formalised in the proof assistant PVS [AFO15]. Interesting differences regarding Urban's Isabelle/HOL approach include that the specification of the nominal unification algorithm is given in a functional style and not as an inference-rule procedure. A byproduct of the PVS formalisation is related to essential observations on how nominal unification implies differences in rewriting properties, such as confluence [AFGO15], on nominal rewriting [FG07].

*Supported by FAPDF DE 00193.00001175/21-11, CNPq Universal 409003/21-2. Author partially funded by a CNPq research grant 313290/21-0.

Nominal equational unification was earlier investigated in [ARFNS16] and [SKLV16]. The former paper introduces nominal narrowing and a lifting result relating nominal narrowing and unification. The latter paper presents a nominal unification approach for higher-order expressions with recursive let. Further, [AdCSFN17a] proposed an inductive nominal C-unification algorithm based on a set of inference rules specified in Coq. For each solvable nominal C-unification problem, the simplification process generates a finite set of fixed-point equations of the form $\pi \cdot X \approx^{?}_{\alpha,C} X$, where π is a permutation, and X is a variable, with a set of *freshness constraints* and a *substitution*. In contrast, the output of the standard nominal unification algorithm consists only of substitutions and freshness constraints. Fixed-point equations can be easily eliminated in the standard unification algorithm (freshness constraints replace them), but this is not the case in the presence of commutative symbols. For instance, if $+$ is commutative, the fixed point equation $(a\,b) \cdot X \approx^{?}_{\alpha,C} X$ has infinite solutions $X/a+b, X/(a+b)+(a+b), \ldots$ (see [AdCSFN17b] for a procedure to generate solutions of fixed-point equations). A formalisation in Coq of the inductive inference rules, including correctness and completeness proofs, was later adapted to solve C-matching problems (see [ARdCSFNS19], [AdCSF$^+$19]). Moreover, a functional version of the nominal C unification algorithm was formalised and verified in the proof assistant PVS [AFSN19, AdCSF$^+$21]. Such fixed-point equations are recurrent in nominal syntax (e.g., [SKLV16, SKL$^+$22]).

The formalisation of nominal C equational reasoning gave rise to a crucial difference regarding syntactic C-unification; indeed, the type of unification is infinitary in contrast to the finitary type in the syntactic setting. Although this, the set of solutions may be characterised as a kind of pre-unifiers consisting of a set of freshness constraints, a substitution, and a set of fixed-point equations, giving rise to a provable countable set of unifiers. An alternative so-called nominal fixed-point approach to finitely represent solutions was introduced in [AFN18, AFN20].

3 Nominal AC reasoning

An earlier combinatorial approach for nominal AC unification was presented in UNIF 19, for which only soundness was formalised in PVS. Indeed, neither formal mechanisation of its completeness nor minimality was provided. After that, a significant formalisation effort was performed to obtain the first mechanisation of the Stikel-Fages' AC-unification algorithm [AFSS22]. The formalisation modifies the almost half a century old linear Diophantine-based approach by Stickel [Sti75, Sti81], and the correction on its termination further proposed by Fages [Fag84, Fag87]. Such mechanisation of AC unification in PVS is not routine-formalisation work; before it, only a Coq formalisation of AC-matching has been reported by Contejean [Con04].

The PVS specification of the algorithm is a functional modification of the original Stickel-Fages approach that avoids mutual recursion. Moreover, the verification involves formalisations of its termination, soundness, and completeness. The formalisation involves almost one thousand proved formulas (PVS type correctness obligations - TCCs, lemmas and theorems), from which, as expected, more than 70% are related to auxiliary lemmas on basic notions, properties of application of the AC unification rule, and termination. Further work was performed to adapt the Diophantine-based approach to the nominal setting in [AFS$^+$23]. A nominal AC-matching algorithm was introduced and verified in PVS. Furthermore, an interesting question remains open for nominal AC-unification. The following examples are taken from the last reference to explain the question.

3.1 First-Order AC-Unification

We give an example, adapted from [Sti81], of how we would solve the first-order AC-unification problem $\{f(X, X, Y, a, b, c) \approx^? f(b, b, b, c, Z)\}$, where f is an AC-function symbol. In a high-level view, this technique converts an AC-unification problem into a linear Diophantine equation. It uses a basis of solutions of the Diophantine equation to get a complete set of AC-unifiers to our original problem.

The first step is eliminating common arguments in the terms we are trying to unify. The problem is now $\{f(X, X, Y, a) \approx^? f(b, b, Z)\}$.

The second step, called variable abstraction, is to associate the unification problem with a linear Diophantine equation, where each argument of the terms corresponds to one variable in the equation, and the coefficient of this variable in the system is the number of occurrences of the argument. In our case, the linear Diophantine equation obtained is: $2X_1 + X_2 + X_3 = 2Y_1 + Y_2$ (X_1 is associated with argument X, X_2 with argument Y and so on; the coefficient of variable X_1 is two, since argument X occurs twice in $f(X, X, Y, a)$ and so on).

X_1	X_2	X_3	Y_1	Y_2	New vars
0	0	1	0	1	Z_1
0	1	0	0	1	Z_2
0	0	2	1	0	Z_3
0	1	1	1	0	Z_4
0	2	0	1	0	Z_5
1	0	0	0	2	Z_6
1	0	0	1	0	Z_7

(1)

The third step generates a basis of solutions to the equation and associates a new variable (the Z_is) to each solution. The result is shown in Table (1). As we will soon see, the unification problem $\{f(X, X, Y, a) \approx^? f(b, b, Z)\}$ may branch into (possibly) many unification problems and the new variables Z_is will be the building blocks for the right-hand side of these unification problems. Observing Table (1) we relate the "old variables" with the "new variables":

$$\begin{aligned} X_1 &= Z_6 + Z_7 \\ X_2 &= Z_2 + Z_4 + 2Z_5 \\ X_3 &= Z_1 + 2Z_3 + Z_4 \\ Y_1 &= Z_3 + Z_4 + Z_5 + Z_7 \\ Y_2 &= Z_1 + Z_2 + 2Z_6. \end{aligned} \tag{2}$$

To explore all possible solutions, we must consider whether we will include or not each solution on our basis. Since seven solutions compose our basis (one for each new variable), there are 2^7 cases to consider. Since including a solution of our basis means setting the corresponding variable Z_i to 1 and not including it means setting it to 0, we must respect the constraint that none of the original variables receives 0. Eliminating the cases that do not respect this constraint, we are left with 69 cases.

For example, if we decide to include only the solutions represented by the variables Z_1, Z_4 and Z_6, the corresponding unification problem, according to the Equations (2), becomes:

$$P = \{X_1 \approx^? Z_6, X_2 \approx^? Z_4, X_3 \approx^? f(Z_1, Z_4), Y_1 \approx^? Z_4, Y_2 \approx^? f(Z_1, Z_6, Z_6)\} \tag{3}$$

Furthermore, the cases where a variable that does not represent a variable term is paired with an AC-function application can be dropped. For instance, the unification problem P should be discarded since the variable X_3 represents the constant a, and we cannot unify a with $f(Z_1, Z_4)$. This constraint eliminates 63 of the 69 potential unifiers.

Finally, we replace the variables X_1, X_2, X_3, Y_1, Y_2 with the original arguments they substituted and proceed with the unification. Some unification problems that we will explore will be unsolvable and discarded later, as:

$$\{X \approx^? Z_6, Y \approx^? Z_4, a \approx^? Z_4, b \approx^? Z_4,\ Z \approx^? f(Z_6, Z_6)\}$$

We cannot unify Z_4 with a and b simultaneously. In the end, the solutions computed will be:

$$\begin{array}{ll} \sigma_1 = \{Y \mapsto f(b,b), Z \mapsto f(a,X,X)\} & \sigma_2 = \{Y \mapsto f(Z_2,b,b), Z \mapsto f(a,Z_2,X,X)\} \\ \sigma_3 = \{X \mapsto b, Z \mapsto f(a,Y)\} & \sigma_4 = \{X \mapsto f(Z_6,b), Z \mapsto f(a,Y,Z_6,Z_6)\} \end{array} \tag{4}$$

When using the technique to unify $f(X,X,Y,a,b,c)$ with $f(b,b,b,c,Z)$, we obtained unification problems that only contain the variables X_1, X_2, X_3, Y_1, Y_2 or AC-functions whose arguments are all variables (for instance P in Equation 3). However, this does not mean that the technique cannot be applied to general AC-unification problems since we eventually replace the variables X_1, X_2, X_3, Y_1, Y_2 by their corresponding arguments (X, Y, a, b, Z respectively) and proceed with unification.

3.2 Issues to Adapt the Algorithm to the Nominal Setting

The example describes the process of trying to unify two terms $t \equiv f(t_1, \ldots, t_m)$ and $s \equiv f(s_1, \ldots, s_n)$, where f is an AC-function symbol. Four modifications were necessary to adapt this process to the nominal setting.

The first is related to eliminating common arguments: we do not eliminate arguments t_i and s_j of t and s if they are equal modulo AC. They are eliminated if they are α-equivalent (modulo AC) under the context Γ we are working with, i.e. if $\Gamma \vdash t_i \approx_\alpha s_j$. If we have as a hypothesis that (Δ, δ) is the solution to the problem we are working with, the correctness of this step boils down to proving that from $\Gamma \vdash t_i \approx_\alpha s_j$ we have $\Delta \vdash \delta t_i \approx_\alpha \delta s_j$. This is possible to prove by using the fact that $\Delta \vdash \delta\Gamma$.

The second change is related to the new variables introduced and that in the nominal setting, a moderated variable $\pi \cdot X$ always has a permutation π suspended on the variable X. What should be the permutation π suspended on the new variables? Since the ultimate goal of these new variables is to outline the combinatory between the arguments of t and the arguments of s, we suspended the identity permutation on the new variables. For instance, in the example, we would have the moderated variables $id \cdot Z_1, \ldots, id \cdot Z_7$, which we would write simply as $Z_1, \ldots, Z_7$.

In the example, we have variables X_1, X_2, X_3, Y_1, Y_2 to represent the arguments X, Y, a, b, Z respectively, and we say that when generating the new unification problems, we can discard the ones "where a variable that does not represent a variable term is paired with an AC-function application". Here, we can also discard unification problems where a moderated variable $\pi \cdot X$, with $X \in \mathcal{X}$, where $\mathcal{X}$ is a set of "protected variables", is paired with an AC-function application. This is the third change to adapt to the nominal setting, which works for AC-matching selecting as protected variables those variables occurring in the right-hand side of the problem [AFS+23].

Finally, we must guarantee that the new variables Z_is introduced by the algorithm can be instantiated. Since those new variables are not in the set V, we ensure that by putting the restriction that $\mathcal{X} \subseteq V$ in the definition of allowed inputs.

Adapting the mechanism to deal with nominal AC-Unification gives rise to a circularity explained in the next example, which does not appear when using it to solve nominal AC-matching problems.

Suppose, under the empty context (i.e. $\Gamma = \emptyset$), we want to solve the equational constraint $f(X, W) \approx^? f(\pi X, \pi Y)$, with $\mathcal{X} = \emptyset$. The linear Diophantine equation associated with this problem is $U_1 + U_2 = V_1 + V_2$, where U_1 is associated with argument X, U_2 with argument W, V_1 with argument πX and V_2 with πY. A basis of solutions to this linear Diophantine equation is shown in Table (5).

U_1	U_2	V_1	V_2	$U_1 + U_2$	$V_1 + V_2$	New vars
0	1	0	1	1	1	Z_1
0	1	1	0	1	1	W_1
1	0	0	1	1	1	Y_1
1	0	1	0	1	1	X_1

(5)

We choose the names of the new variables to be Z_1, W_1, Y_1 and X_1 deliberately to make the loop in nominal AC-unification more transparent. Finally, we will branch into new equational constraints, using Table (5) to construct them. The algorithm bifurcates into seven branches, shown below, along with their corresponding equational constraints:

$$
\begin{aligned}
B_1 &= \{X \approx^? X_1, W \approx^? Z_1, \pi X \approx^? X_1, \pi Y \approx^? Z_1\}\\
B_2 &= \{X \approx^? Y_1, W \approx^? W_1, \pi X \approx^? W_1, \pi Y \approx^? Y_1\}\\
B_3 &= \{X \approx^? Y_1 + X_1, W \approx^? W_1, \pi X \approx^? W_1 + X_1, \pi Y \approx^? Y_1\}\\
B_4 &= \{X \approx^? Y_1 + X_1, W \approx^? Z_1, \pi X \approx^? X_1, \pi Y \approx^? Z_1 + Y_1\}\\
B_5 &= \{X \approx^? X_1, W \approx^? Z_1 + W_1, \pi X \approx^? W_1 + X_1, \pi Y \approx^? Z_1\}\\
B_6 &= \{X \approx^? Y_1, W \approx^? Z_1 + W_1, \pi X \approx^? W_1, \pi Y \approx^? Z_1 + Y_1\}\\
B_7 &= \{X \approx^? Y_1 + X_1, W \approx^? Z_1 + W_1, \pi X \approx^? W_1 + X_1, \pi Y \approx^? Z_1 + Y_1\}
\end{aligned}
$$

The next step is to instantiate moderated variables. We denote branch i by B_i, the substitution computed in this branch by σ_i and show the result after performing the instantiations. For brevity, when presenting σ_i, we omit the instantiation of variables X_1, W_1, Y_1, and Z_1 since they were not in the initial problem.

$$
\begin{array}{ll}
B_1 & \{\pi X \approx^? X\}, \sigma_1 = \{W \mapsto \pi Y\}\\
B_2 & \sigma_2 = \{W \mapsto \pi^2 Y, X \mapsto \pi Y\}\\
B_3 & \{f(\pi^2 Y, \pi X_1) \approx^? f(W, X_1)\}, \sigma_3 = \{X \mapsto f(\pi Y, X_1)\}\\
B_{4,5} & \textit{No solution}\\
B_6 & \sigma_6 = \{W \mapsto f(Z_1, \pi X), Y \mapsto f(\pi^{-1} Z_1, \pi^{-1} X)\}\\
B_7 & \{f(\pi Y_1, \pi X_1) \approx^? f(W_1, X_1)\},\\
 & \sigma_7 = \{X \mapsto f(Y_1, X_1),\ W \mapsto f(Z_1, W_1), Y \mapsto f(\pi^{-1} Z_1, \pi^{-1} Y_1)\}
\end{array}
$$

Branches 3 and 7 are a renaming of the original problem

$$f(X, W) \approx^? f(\pi X, \pi Y).$$

Regarding Branch 3, notice that if we rewrite $\sigma_3 = \{X \mapsto f(\pi Y, X_1)\}$ as $\sigma_3' = \{Y \mapsto \pi^{-1} Y_1, X \mapsto f(\pi Y, X_1), \}$, then the equational constraint of the mentioned branch is:

$$f(X_1, W_1) \approx^? f(\pi X_1, \pi Y_1).$$

Regarding Branch 7, it is even simpler to see the renaming, as the equational constraint is:

$$f(X_1, W_1) \approx^? f(\pi X_1, \pi Y_1),$$

The formalisation in PVS of a sound and complete nominal AC-matching algorithm proposed in [AFS+23] is possible because such circularity does not appear for AC-matching. Still, how to contour the circularity for the case of nominal AC unification remains an open question.

4 Acknowledgements

Progress on the nominal PVS library has been obtained through cooperation with our colleagues and students. In particular, the students, Ana Cristina Rocha Oliveira, Washington Luís de Carvalho Segundo, and Gabriel Ferreira Silva, provided the fine formalisations, which are described in detail in the references and in their PhD theses ([RO16], [dCS19]).

References

[AdCSF+19] Mauricio Ayala-Rincón, Washington de Carvalho Segundo, Maribel Fernández, Daniele Nantes-Sobrinho, and Ana Cristina Rocha Oliveira. A formalisation of nominal α-equivalence with A, C, and AC function symbols. *Theor. Comput. Sci.*, 781:3–23, 2019.

[AdCSF+21] Mauricio Ayala-Rincón, Washington de Carvalho Segundo, Maribel Fernández, Gabriel Ferreira Silva, and Daniele Nantes-Sobrinho. Formalising Nominal C-Unification Generalised with Protected Variables. *Math. Struct. Comput. Sci.*, 31(3):286–311, 2021.

[AdCSFN17a] Mauricio Ayala-Rincón, Washington de Carvalho Segundo, Maribel Fernández, and Daniele Nantes-Sobrinho. Nominal C-Unification. In *Logic-Based Program Synthesis and Transformation - 27th International Symposium, LOPSTR, Revised Selected Papers*, volume 10855 of *LNCS*, pages 235–251. Springer, 2017.

[AdCSFN17b] Mauricio Ayala-Rincón, Washington de Carvalho Segundo, Maribel Fernández, and Daniele Nantes-Sobrinho. On Solving Nominal Fixpoint Equations. In *Frontiers of Combining Systems - 11th International Symposium, FroCoS*, volume 10483 of *LNCS*, pages 209–226. Springer, 2017.

[AFGO15] Mauricio Ayala-Rincón, Maribel Fernández, Murdoch James Gabbay, and Ana Cristina Rocha Oliveira. Checking Overlaps of Nominal Rewriting Rules. In *Proc. of the 10th Workshop on Logical and Semantic Frameworks, with Applications, LSFA*, volume 323 of *ENTCS*, pages 39–56. Elsevier, 2015.

[AFN18] Mauricio Ayala-Rincón, Maribel Fernández, and Daniele Nantes-Sobrinho. Fixed-Point Constraints for Nominal Equational Unification. In *3rd International Conference on Formal Structures for Computation and Deduction, FSCD*, volume 108 of *LIPIcs*, pages 7:1–7:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[AFN20] Mauricio Ayala-Rincón, Maribel Fernández, and Daniele Nantes-Sobrinho. On Nominal Syntax and Permutation Fixed Points. *Log. Methods Comput. Sci.*, 16(1), 2020.

[AFO15] Mauricio Ayala-Rincón, Maribel Fernández, and Ana Cristina Rocha Oliveira. Completeness in PVS of a Nominal Unification Algorithm. In *Proc. of the 10th Workshop on Logical and Semantic Frameworks, with Applications, LSFA*, volume 323 of *ENTCS*, pages 57–74. Elsevier, 2015.

[AFS+23] Mauricio Ayala-Rincón, Maribel Fernández, Gabriel Ferreira Silva, Temur Kutsia, and Daniele Nantes-Sobrinho. Nominal ac-matching. In *To appear in Proc. of the 16th Conference on Intelligent Computer Mathematics, CICM*, 2023.

[AFSN19] Mauricio Ayala-Rincón, Maribel Fernández, Gabriel Ferreira Silva, and Daniele Nantes-Sobrinho. A Certified Functional Nominal C-Unification Algorithm. In *Logic-Based Program Synthesis and Transformation - 29th International Symposium, LOPSTR, Revised Selected Papers*, volume 12042 of *LNCS*, pages 123–138. Springer, 2019.

[AFSS22] Mauricio Ayala-Rincón, Maribel Fernández, Gabriel Ferreira Silva, and Daniele Nantes Sobrinho. A Certified Algorithm for AC-Unification. In *7th International Conference on Formal Structures for Computation and Deduction, FSCD*, volume 228 of *LIPIcs*, pages 8:1–8:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[ARdCSFNS19] Mauricio Ayala-Rincón, Washington de Carvalho-Segundo, Maribel Fernández, and Daniele Nantes-Sobrinho. A Formalisation of Nominal C-Matching through Unification with Protected Variables. *ENTCS*, 344:47 – 65, 2019.

[ARFNS16] Mauricio Ayala-Rincón, Maribel Fernández, and Daniele Nantes-Sobrinho. Nominal Narrowing. In *1st International Conference on Formal Structures for Computation and Deduction, FSCD*, page 11. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[CF08] Christophe Calvès and Maribel Fernández. A polynomial nominal unification algorithm. *Theor. Comput. Sci.*, 403(2-3):285–306, 2008.

[Con04] Evelyne Contejean. A Certified AC Matching Algorithm. In *Proc. of the 15th International Conference on Rewriting Techniques and Applications, RTA*, volume 3091 of *LNCS*, pages 70–84. Springer, 2004.

[dCS19] Washington Luís Ribeiro de Carvalho Segundo. *Nominal equational problems modulo associativity, commutativity and associativity-commutativity.* PhD thesis, Graduate Program in Informatics, Universidade de Brasília. Available as https://repositorio.unb.br/handle/10482/35474, 2019. https://repositorio.unb.br/handle/10482/35474.

[Fag84] François Fages. Associative-Commutative Unification. In *7th International Conference on Automated Deduction CADE*, volume 170 of *LNCS*, pages 194–208. Springer, 1984.

[Fag87] François Fages. Associative-Commutative Unification. *J. of Sym. Computation*, 3(3):257–275, 1987.

[FG07] Maribel Fernández and Murdoch James Gabbay. Nominal rewriting. *Information and Computation*, 205(6):917–965, 2007.

[LV10] Jordi Levy and Mateu Villaret. An Efficient Nominal Unification Algorithm. In *Proc. of the 21st International Conference on Rewriting Techniques and Applications, RTA*, volume 6 of *LIPIcs*, pages 209–226. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

[OS99] Sam Owre and Natarajan Shankar. The Formal Semantics of PVS. Technical Report 97-2R, SRI International Computer Science Laboratory, Menlo Park CA 94025 USA, 1997, revised 1999.

[Pit13] Andrew M Pitts. *Nominal Sets: Names and Symmetry in Computer Science.* Cambridge University Press, 2013.

[RO16] Ana Cristina Rocha Oliveira. *Unification, confluence, and intersection types for nominal rewriting systems.* PhD thesis, Graduate Program in Informatics, Universidade de Brasília. Available as https://repositorio.unb.br/handle/10482/22387, 2016.

[SKL+22] Manfred Schmidt-Schauß, Temur Kutsia, Jordi Levy, Mateu Villaret, and Yunus D. K. Kutz. Nominal Unification and Matching of Higher Order Expressions with Recursive Let. *Fundam. Informaticae*, 185(3):247–283, 2022.

[SKLV16] Manfred Schmidt-Schauß, Temur Kutsia, Jordi Levy, and Mateu Villaret. Nominal Unification of Higher Order Expressions with Recursive Let. In *Logic-Based Program Synthesis and Transformation - 26th International Symposium, LOPSTR, Revised Selected Papers*, volume 10184 of *LNCS*, pages 328–344. Springer, 2016.

[Sti75] Mark E. Stickel. A Complete Unification Algorithm for Associative-Commutative Functions. In *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence, IJCAI*, pages 71–76, 1975.

[Sti81] Mark E. Stickel. A Unification Algorithm for Associative-Commutative Functions. *J. of the ACM*, 28(3):423–434, 1981.

[UPG04] Christian Urban, Andrew M. Pitts, and Murdoch Gabbay. Nominal unification. *Theor. Comput. Sci.*, 323(1-3):473–497, 2004.

[Urb10] Christian Urban. Nominal Unification Revisited. In *Proc. of the 24th International Workshop on Unification, UNIF*, volume 42 of *EPTCS*, pages 1–11, 2010.