A Rule-Based Procedure for Equivariant Nominal Unification*

Takahito Aoto[1] and Kentaro Kikuchi[2]

[1] Faculty of Engineering, Niigata University
aoto@ie.niigata-u.ac.jp
[2] RIEC, Tohoku University
kentaro@nue.riec.tohoku.ac.jp

Abstract

Nominal rewriting is a rewriting formalism that deals with variable binding. An equivariant nominal unification is a basic ingredient of nominal rewriting for computing rewrite steps and critical pairs. We present a rule-based procedure for the equivariant nominal unification.

1 Introduction

Rewriting captures various computational aspects in equational reasoning. *Higher-order rewriting* deals with rewriting of expressions with higher-order functions and/or variable binding. The nominal approach [5, 6] is a novel approach to deal with variable binding and α-equivalence—unlike other approaches, it incorporates permutations and freshness conditions on variables (atoms) as basic ingredients. *Nominal rewriting* [3, 4] is a formalism of rewriting based on the nominal approach.

A basic ingredient of nominal rewriting is a computation of rewrite step, i.e. to compute a term t such that $\Delta \vdash s \to_{\mathcal{R}} t$ or even (representatives of) all t such that $\Delta \vdash s \to_{\mathcal{R}} t$, from a given nominal rewrite system $\mathcal{R}$, a freshness context Δ and a term s. The main challenge here is to find suitable π and σ such that $\Delta \vdash \nabla^{\pi}\sigma$ and $\Delta \vdash s|_p \approx_\alpha l^{\pi}\sigma$, when fixing $\nabla \vdash l \to r \in \mathcal{R}$ and a position p in s. The problem is known to be an instance of *equivariant nominal unification* [2]. The equivariant nominal unification is also necessary for computing critical pairs which is a basic component for developing the Knuth-Bendix completion procedure.

In [2], an equivariant nominal unification procedure is given. In this paper, we consider a framework of equivariant nominal unification which is simpler than [2], and present an alternative equivariant nominal unification procedure, which is fully presented in a rule-based form. In what follows, we will develop our procedure from a basic ingredient to a full procedure in a step by step manner. Our procedure will be completed in Section 5 and a discussion on correctness will be postponed until Section 6. We refer to [3, 4, 7] for basic notions and notations on nominal terms. Familiarity with nominal unification [8] is assumed from Section 5.

2 Atom Identity Solving

We fix a countably infinite set $\mathcal{A} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots\}$ of *atoms* ranged over by $a, b, c, \ldots$ and a countably infinite set $\mathcal{X}_A$ of *atom variables* ranged over by $A, B, \ldots$. Elements of $\mathcal{A} \cup \mathcal{X}_A$ are *atom expressions* ranged over by $\alpha, \beta, \ldots$. *Atom equations* are of the form $\alpha \approx \beta$ and *atom disequations* are of the form $\alpha \not\approx \beta$ where $\alpha \sim \beta$ and $\beta \sim \alpha$ ($\sim \in \{\approx, \not\approx\}$) are identified. An

*This work is partially supported by JSPS KAKENHI (Nos. 15K00003, 16K00091).

atom identity problem is a set of atom equations and atom disequations. The *atom identity solving procedure* $\mathtt{ProcAtomIdent}_0$ consists of the following derivation rules where a lower set is derived from an upper set.

$$\frac{\{\alpha \approx \alpha\} \uplus E}{E} \qquad \frac{\{a \not\approx b\} \uplus E}{E}\ a \neq b \qquad \frac{\{\alpha \not\approx \alpha\} \uplus E}{\bot} \qquad \frac{\{a \approx b\} \uplus E}{\bot}\ a \neq b$$

$$\frac{\{A \approx \beta\} \uplus E}{\{A \mapsto \beta\} \cup E[\beta/A]}\ A \neq \beta$$

Here, $\uplus$ denotes the disjoint union, and $E[\beta/A]$ denotes the constraints obtained by replacing all occurrences of A in E by β. A solved form E is either $\bot$ or a set of disequations of the form $A \not\approx \beta$, and assignments of the form $A \mapsto \alpha$ where A does not occur elsewhere in E. The intended meaning is that $\approx$ ($\not\approx$) is an equality (disequality) on atoms and that atom variables are instantiated by atoms.

We now also consider a countably infinite set $\mathcal{X}_P$ of *permutation variables* ranged over by $P, Q \ldots$. An expression of the form $P \cdot \alpha$ is a *primitive atomic expression*. We now extend the atom identity problems to allow also the form $P \cdot \alpha \sim \beta$ ($\sim \in \{\approx, \not\approx\}$). The following derivation rules are added to $\mathtt{ProcAtomIdent}_0$:

$$\frac{\{P{\cdot}\alpha \approx \beta\} \uplus E}{\{P : \alpha \mapsto \beta\} \cup E} \qquad \frac{\{P{\cdot}\alpha \not\approx \beta\} \uplus E}{\{P : \alpha \mapsto A, A \not\approx \beta\} \uplus E}$$

$$\frac{\{P : \alpha \mapsto \beta'\} \uplus E}{\{\beta \approx \beta'\} \cup E}\ P : \alpha \mapsto \beta \in E \qquad \frac{\{P : \alpha' \mapsto \beta\} \uplus E}{\{\alpha \approx \alpha'\} \cup E}\ P : \alpha \mapsto \beta \in E$$

$$\frac{E}{\{\beta \not\approx \beta'\} \cup E}\ \begin{array}{l} P : \alpha \mapsto \beta, P : \alpha' \mapsto \beta' \in E \\ \alpha \not\approx \alpha' \in E, \beta \not\approx \beta' \notin E \end{array} \qquad \frac{E}{\{\alpha \not\approx \alpha'\} \cup E}\ \begin{array}{l} P : \alpha \mapsto \beta, P : \alpha' \mapsto \beta' \in E \\ \beta \not\approx \beta' \in E, \alpha \not\approx \alpha' \notin E \end{array}$$

where in the second derivation rule, A is a fresh atom variable. The result is the (full) atom identity solving procedure $\mathtt{ProcAtomIdent}$. Then a solved form E ($\neq \bot$) can also contain constraints of the form $P : \alpha \mapsto \beta$. The intended meaning is that a permutation is a (finite) bijection on $\mathcal{A}$, $P \cdot \alpha$ is an application, and $P : \alpha \mapsto \beta$ is a constraint on P such that P maps α to β.

3 Atomic Equality Solving Procedure

Atomic expressions and *permutation expressions* are generated by the following grammar in a mutually recursive way:

$$\begin{array}{llcll} \textit{atomic expression:} & v, w \in \mathcal{E}_A & := & \Pi{\cdot}\alpha & \text{(moderated atom expression)} \\ \textit{permutation expression:} & \Pi, \Psi \in \mathcal{E}_P & := & P & \text{(permutation-variables)} \\ & & & \mid \mathsf{Id} & \text{(identity)} \\ & & & \mid (v\ w) & \text{(swap)} \\ & & & \mid \Pi \circ \Psi & \text{(composition)} \\ & & & \mid \Pi^{-1} & \text{(inverse)} \end{array}$$

Here we have the following new constructs "Id", "(,)", "$\circ$" and "${}^{-1}$". The names of the construction suggests the indented meaning. For example, we have $(((P \circ Q)^{-1} \cdot A)\ B) \in \mathcal{E}_P$ and

$(((P \circ Q)^{-1} \cdot A)\ B) \cdot \mathsf{c} \in \mathcal{E}_A$. Clearly, a primitive atomic expression is an atomic expression. An *atomic (dis)equality* is a (dis)equation of the form $\Pi{\cdot}\alpha \sim \Pi'{\cdot}\beta$, and a set of atomic (dis)equalities is an *atomic equality problem*. The *atomic equality solving procedure* `ProcAtomEq` consists of the following derivation rules:

$$\frac{\{\Pi{\cdot}\alpha \sim \Pi'{\cdot}\beta\} \uplus E}{\{\Pi{\cdot}\alpha \approx A, \Pi'{\cdot}\beta \sim A\} \cup E} \qquad \frac{\{P{\cdot}v \sim \beta\} \uplus E}{\{v \approx A, P{\cdot}A \sim \beta\} \cup E}\ v \notin \mathcal{X}_A \cup \mathcal{A} \qquad \frac{\{\mathsf{Id}{\cdot}v \sim \beta\} \uplus E}{\{v \sim \beta\} \cup E}$$

$$\frac{\{(v\ v'){\cdot}w \sim \beta\} \uplus E}{\{v \not\approx w, v' \not\approx w, w \sim \beta\} \cup E} \qquad \frac{\{(v\ v'){\cdot}w \sim \beta\} \uplus E}{\{v \approx w, v' \sim \beta\} \cup E} \qquad \frac{\{(v\ v'){\cdot}w \sim \beta\} \uplus E}{\{v' \approx w, v \sim \beta\} \cup E}$$

$$\frac{\{(\Pi \circ \Pi'){\cdot}w \sim \beta\} \uplus E}{\{\Pi'{\cdot}w \approx A, \Pi{\cdot}A \sim \beta\} \cup E} \qquad \frac{\{\Pi^{-1}{\cdot}w \sim \beta\} \uplus E}{\{\Pi{\cdot}\beta \approx A, w \sim A\} \cup E}$$

In the first two and the last two derivation rules, A is a fresh atom variable. This procedure *non-deterministically* reduces an atomic equality problem to an atom identity problem, which in turn given to the procedure `ProcAtomIdent`.

4 Freshness Constraint Solving Procedure

We now fix a countably infinite set $\mathcal{X}$ of *term variables* ranged over by $X, Y, \ldots$. A *nominal signature* Σ is a set of *function symbols* ranged over by $f, g, \ldots$. *Term expressions* are given by the following grammar:

term expression:	$S, T \in \mathcal{E}_T$	$:=$	v	(atomic expressions)
			$\mid \Pi{\cdot}X$	(moderated term-variables)
			$\mid [v]T$	(abstraction)
			$\mid (f\ T)$	(function applications)
			$\mid \langle T_1, \ldots, T_n\rangle$	(tuples)

A *freshness constraint expression* is a pair $v\#T$ of $v \in \mathcal{E}_A$ and $T \in \mathcal{E}_T$. An *freshness constraint problem* is a finite set of freshness constraint expressions. The *freshness constraint solving procedure* `ProcFreshCnstr` consists of the following derivation rules:

$$\frac{\{v\#w\} \uplus P}{\{v \not\approx w\} \cup P} \qquad \frac{\{v\#\Pi{\cdot}X\} \uplus P}{\{\Pi^{-1}{\cdot}v \approx A, A\#X\} \cup P} \qquad \frac{\{v\#[w]T\} \uplus P}{\{v \approx w\} \cup P}$$

$$\frac{\{v\#[w]T\} \uplus P}{\{v\#T\} \cup \{v \not\approx w\} \cup P} \qquad \frac{\{v\#f\ T\} \uplus P}{\{v\#T\} \cup P} \qquad \frac{\{v\#\langle T_1, \ldots, T_n\rangle\} \uplus P}{\{v\#T_1, \ldots, v\#T_n\} \cup P}$$

In the second rule, A is a fresh atom variable. Permutation action $\Pi{\cdot}v \in \mathcal{E}_A$ on an atomic expression v by permutation Π used in the second rule is given by $\Pi{\cdot}(\Pi'{\cdot}\alpha) = (\Pi \circ \Pi'){\cdot}\alpha$. This procedure *non-deterministically* reduces a freshness constraint problem to an atomic equality problem supplemented with primitive freshness constraints of the form $A\#X$. The result is given to the procedure `ProcAtomEq`, where any primitive freshness constraint is omitted except that any atom variable A in $A\#X$ needs to be updated accordingly by the operation $[\beta/A]$.

5 Equivariant Unification Procedure

A permutation action $\Pi{\cdot}T \in \mathcal{E}_T$ on a term expression T by Π is given by the following rules:

$$\begin{array}{rclrcl} \Pi{\cdot}v & = & \Pi{\cdot}v & \Pi{\cdot}([v]T) & = & [\Pi{\cdot}v](\Pi{\cdot}T) \\ \Pi{\cdot}(\Pi'{\cdot}X) & = & (\Pi \circ \Pi'){\cdot}X & \Pi{\cdot}(f\ T) & = & f\ \Pi{\cdot}T \\ \Pi{\cdot}\langle T_1,\dots,T_n\rangle & = & \langle \Pi{\cdot}T_1,\dots,\Pi{\cdot}T_n\rangle & & & \end{array}$$

Here, the rhs of the first rule is given by the permutation action on an atomic expression.

A *substitution* is a mapping $\sigma : \mathcal{X} \to \mathcal{E}_T$ with a finite domain $\{X \in \mathcal{X} \mid \sigma(X) \neq X\}$. A substitution σ such that $\sigma(X) = T$ with domain $\{X\}$ is written as $\{X \mapsto T\}$. The application of a substitution on term expressions is given by

$$\begin{array}{rclrcl} v\sigma & = & v & ([v]T)\sigma & = & [v](T\sigma) \\ (\Pi{\cdot}X)\sigma & = & \Pi{\cdot}\sigma(X) & (f\ T)\sigma & = & f\ (T\sigma) \\ \langle T_1,\dots,T_n\rangle\sigma & = & \langle T_1\sigma,\dots,T_n\sigma\rangle & & & \end{array}$$

Here, rhs of the second rule is given by the permutation action (on a term expression).

An ***α**-equivalence constraint* is a pair $S \approx_\alpha T$ of $S, T \in \mathcal{E}_T$. An *equivariant unification problem* (EUP) is a finite set of **α**-equivalence constraints. The *equivariant unification procedure* `ProcEqvUnif` consists of the following derivation rules:

$$\frac{\langle\{\Pi{\cdot}X \approx_\alpha \Pi'{\cdot}X\} \uplus E, \sigma\rangle}{\langle\{\#(X,\Pi,\Pi')\} \cup E, \sigma\rangle} \qquad \frac{\langle\{T \approx_\alpha \pi{\cdot}X\} \uplus E, \sigma\rangle}{\langle E\{X \mapsto \pi^{-1}{\cdot}T\}, \{X \mapsto \pi^{-1}{\cdot}T\} \circ \sigma\rangle}\ X \notin \mathcal{X}(T)$$

$$\frac{\langle\{[v]S \approx_\alpha [w]T\} \uplus E, \sigma\rangle}{\langle\{v \approx w, S \approx_\alpha T\} \cup E, \sigma\rangle} \qquad \frac{\langle\{[v]S \approx_\alpha [w]T\} \uplus E, \sigma\rangle}{\langle\{v \not\approx w, S \approx_\alpha (v\ w){\cdot}T, v\#T\} \cup E, \sigma\rangle}$$

$$\frac{\langle\{f\ S \approx_\alpha f\ T\} \uplus E, \sigma\rangle}{\langle\{S \approx_\alpha T\} \cup E, \sigma\rangle} \qquad \frac{\langle\{\langle S_1,\dots,s_n\rangle \approx_\alpha \langle T_1,\dots,T_n\rangle\} \uplus E, \sigma\rangle}{\langle\{S_1 \approx_\alpha T_1,\dots,S_n \approx_\alpha T_n\} \cup E, \sigma\rangle}$$

Here $\mathcal{X}(T)$ denotes the set of term variables in a term expression T. This procedure starts with a pair of an EUP and the identity substitution. This procedure *non-deterministically* generates a pair of the union of a freshness constraint problem and an atomic equality problem supplemented with additional constraints of the form $\#(X,\Pi,\Pi')$, and a substitution. The former is given to the procedure `ProcFreshCnstr`, where freshness constraints are reduced to atomic equality problem. The constraints of the form $\#(X,\Pi,\Pi')$ is untouched except that any atom variable A in Π, Π' needs to be updated accordingly by the operation $[\beta/A]$. All in all, the procedure `ProcEqvUnif` *non-deterministically* generates an *answer constraint*, which is a finite set of expressions of the following forms:

$$A \mapsto v \mid P : \alpha \mapsto \beta \mid \alpha \not\approx \beta \mid X \mapsto T \mid \alpha\#X \mid \#(X,\Pi,\Pi')$$

where constraints of form $X \mapsto T$ is obtained from the substitution part of `ProcEqvUnif`. The set of all answer constraints generated by `ProcEqvUnif` from given EUP $\mathcal{C}$ is a solution of the EUP, which is denoted by $\mathsf{Sol}(\mathcal{C})$.

6 Correctness of the Procedure

The sets of *nominal terms* and *permutations* are denoted by $\mathcal{T}$ and $\mathcal{P}$, respectively. An *instantiation* is a pair $\theta = \langle\theta_A, \theta_P\rangle$ of mappings $\theta_A : \mathcal{X}_A \to \mathcal{A}$ and $\theta_P : \mathcal{X}_P \to \mathcal{P}$. For each $\Pi \in \mathcal{E}_P$,

$v \in \mathcal{E}_A$, $S \in \mathcal{E}_T$, their interpretations $[\![\Pi]\!]_\theta \in \mathcal{P}$, $[\![v]\!]_\theta \in \mathcal{A}$, $[\![S]\!]_\theta \in \mathcal{T}$ by an instantiation θ are defined by the following:

$$\begin{array}{rclrclrcl} [\![P]\!]_\theta &=& \theta_P(P) & [\![\Pi\cdot\alpha]\!]_\theta &=& [\![\Pi]\!]_\theta\cdot[\![\alpha]\!]_\theta & [\![a]\!]_\theta &=& a \\ [\![\mathsf{Id}]\!]_\theta &=& Id & [\![\Pi\cdot X]\!]_\theta &=& [\![\Pi]\!]_\theta\cdot X & [\![A]\!]_\theta &=& \theta_A(A) \\ [\![(v\ w)]\!]_\theta &=& ([\![v]\!]_\theta\ [\![w]\!]_\theta) & [\![[v]T]\!]_\theta &=& [[\![v]\!]_\theta][\![T]\!]_\theta & && \\ [\![\Pi\circ\Psi]\!]_\theta &=& [\![\Pi]\!]_\theta\circ[\![\Psi]\!]_\theta & [\![f\ T]\!]_\theta &=& f\ [\![T]\!]_\theta & && \\ [\![\Pi^{-1}]\!]_\theta &=& [\![\Pi]\!]_\theta^{-1} & [\![\langle T_1,\ldots,T_n\rangle]\!]_\theta &=& \langle[\![T_1]\!]_\theta,\ldots,[\![T_n]\!]_\theta\rangle & && \end{array}$$

Note here that "*Id*" etc. in the rhs's of the definitions are not constructs but meta-operations. For example, if we take $\theta_P(P) = (\mathsf{a}\ \mathsf{b})$, $\theta_P(Q) = (\mathsf{b}\ \mathsf{c})$ and $\theta_A(A) = \mathsf{a}$, $\theta_A(B) = \mathsf{b}$ then we have $[\![(((P\circ Q)^{-1}\cdot A)\ B)]\!]_\theta = (\mathsf{c}\ \mathsf{b}) \in \mathcal{P}$, $[\![(((P\circ Q)^{-1}\cdot A)\ B)\cdot\mathsf{c}]\!]_\theta = \mathsf{b} \in \mathcal{A}$ and $[\![[(((P\circ Q)^{-1}\cdot A)\ B)\cdot\mathsf{c}](\mathtt{f}\ \langle P^{-1}\cdot X, Q^{-1}\cdot\mathsf{c}\rangle)]\!]_\theta = [\mathsf{b}](\mathtt{f}\ \langle(\mathsf{a}\ \mathsf{b})\cdot X,\mathsf{b}\rangle) \in \mathcal{T}$.

We put $[\![v\#T]\!]_\theta = [\![v]\!]_\theta\#[\![T]\!]_\theta$ and $[\![S \approx_\alpha T]\!]_\theta = [\![S]\!]_\theta \approx_\alpha [\![T]\!]_\theta$. A *model* of an EUP $\mathcal{C} = \{\gamma_1,\ldots,\gamma_n\}$ is a triple $\langle\theta,\sigma,\Delta\rangle$ of an instantiation θ, a substitution σ and a freshness context Δ such that $\Delta \vdash [\![\gamma_i]\!]_\theta\sigma$ for all $1 \le i \le n$. We write $\langle\theta,\sigma,\Delta\rangle \models \mathcal{C}$ if $\langle\theta,\sigma,\Delta\rangle$ is a model of $\mathcal{C}$. A triple $\langle\theta,\sigma,\Delta\rangle$ is a model of an answer constraint $\mathcal{S}$, written as $\langle\theta,\sigma,\Delta\rangle \models \mathcal{S}$, if $\theta_A(A) = [\![v]\!]_\theta$ for any $A \mapsto v \in \mathcal{S}$, $\theta_P(P)([\![\alpha]\!]_\theta) = [\![\beta]\!]_\theta$ for any $P : \alpha \mapsto \beta \in \mathcal{S}$, $[\![\alpha]\!]_\theta \neq [\![\beta]\!]_\theta$ for any $\alpha \not\approx \beta \in \mathcal{S}$, $\sigma(X) = [\![T]\!]_\theta$ for all $X \mapsto T \in \mathcal{S}$, $\Delta \vdash [\![\alpha]\!]_\theta\#X\sigma$ for all $\alpha\#X \in \mathcal{S}$, and $\Delta \vdash a\#X\sigma$ for any $a \in ds([\![\Pi]\!]_\theta, [\![\Pi']\!]_\theta)$ and $\#(X,\Pi,\Pi') \in \mathcal{S}$.

Now, the correctness of our equivariant unification procedure is stated as follows.

Theorem 1. *For a given EUP $\mathcal{C}$,* `ProcEqvUnif` *computes a finite set $\mathcal{M} = \mathsf{Sol}(\mathcal{C})$ of answer constraints such that, for any model $\langle\theta,\sigma,\Delta\rangle$, $\langle\theta,\sigma,\Delta\rangle \models \mathcal{C}$ iff $\exists\mathcal{S} \in \mathcal{M}.\ \langle\theta,\sigma,\Delta\rangle \models \mathcal{S}$.*

7 Conclusion

In this paper, we have given a rule-based equivariant nominal unification procedure. To the best of our knowledge, such a rule-based procedure has not been reported for the equivariant nominal unification. We anticipate that our rule-based procedure is helpful to give a correctness proof easy to understand and suitable for formal verification. Our procedure has been used to implement a confluence prover for nominal rewriting [1].

References

[1] T. Aoto and K. Kikuchi. Nominal confluence tool. In *Proc. 8th IJCAR*, LNCS. Springer-Verlag, 2016. To appear.

[2] J. Cheney. Equivariant unification. *J. of Automated Reasoning*, 45:267–300, 2010.

[3] M. Fernández and M. J. Gabbay. Nominal rewriting. *Inform. and Comput.*, 205:917–965, 2007.

[4] M. Fernández, M. J. Gabbay, and I. Mackie. Nominal rewriting systems. In *Proc. 6th PPDP*, pages 108–119. ACM Press, 2004.

[5] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Comput.*, 13:341–363, 2002.

[6] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Inform. and Comput.*, 186:165–193, 2003.

[7] T. Suzuki, K. Kikuchi, T. Aoto, and Y. Toyama. Confluence of orthogonal nominal rewriting systems revisited. In *Proc. 26th RTA*, volume 36 of *LIPIcs*, pages 301–317, 2015.

[8] C. Urban, A. M. Pitts, and M. J. Gabbay. Nominal unification. *Theoret. Comput. Sci.*, 323:473–497, 2004.