Nominal Confluence Tool*

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 framework of higher-order rewriting introduced in (Fernández, Gabbay & Mackie, 2004; Fernández & Gabbay, 2007). Recently, (Suzuki et al., 2015) revisited confluence of nominal rewriting in the light of feasibility. We report on an implementation of a confluence tool for (non-closed) nominal rewriting, based on (Suzuki et al., 2015) and succeeding studies.

Keywords: confluence, nominal rewriting, automation, variable binding, higher-order rewriting

1 Introduction

Rewriting captures various computational aspects in equational reasoning [4]. *Higher-order rewriting* deals with rewriting of expressions with higher-order functions and variable binding. Various formalisms for higher-order rewriting have been considered e.g. [12, 14]. *Nominal rewriting* [6, 7] is a formalism of higher-order rewriting, based on the nominal approach for terms and unification [9, 16, 21].

Confluence is a central property in rewriting [4]. Confluence tools for various rewriting formalisms have been developed [2, 10, 17, 22], and a yearly competition for confluence tools has emerged from 2012 [1]. Some basic confluence results for nominal rewriting have been mentioned in [6]. Recently, these results have been revisited and extended by the authors [11, 19, 20] in the light of feasibility and more-in-depth analysis. In this paper, we report on a confluence tool for nominal rewriting based on those confluence studies.

2 Preliminaries

In this section, we recall basic notions and fix notations on nominal terms and rewriting. We refer to [6, 7, 19] for omitted definitions and intuitive explanations.

A *nominal signature* Σ is a set of *function symbols* ranged over by $f, g, \ldots$. We fix a countably infinite set $\mathcal{X}$ of *term variables* ranged over by $X, Y, \ldots$, and a countably infinite set $\mathcal{A} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots\}$ of *atoms* ranged over by $a, b, c \ldots$ (i.e.

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

$\mathsf{a}, \mathsf{b}, \mathsf{c}, \ldots$ stand for objects and $a, b, c, \ldots$ stand for meta-variables). A *swapping* is a pair $(a\ b)$ of atoms. *Permutations* π are bijections on $\mathcal{A}$ with finite $\mathit{support}(\pi) = \{a \in \mathcal{A} \mid a \neq \pi(a)\}$; permutations are represented by compositions of swappings. $\mathcal{P}$ stands for the set of permutations. We put $\mathit{ds}(\pi, \pi') = \{a \in \mathcal{A} \mid \pi{\cdot}a \neq \pi'{\cdot}a\}$ for any $\pi, \pi' \in \mathcal{P}$. *Terms* are generated by the grammar

$$s, t \in \mathcal{T} ::= a \mid \pi{\cdot}X \mid [a]t \mid f\ t \mid \langle t_1, \ldots, t_n \rangle$$

A term of form $\pi{\cdot}X$ is called a *suspension*. A suspension $\mathit{Id}{\cdot}X$ is abbreviated as X, where Id denotes the identity. We write $\mathcal{A}(t)$ and $\mathcal{X}(t)$ for the sets of atoms and term variables occurring in a term t (or any expression t, in general) where the former includes the atoms in abstractions $[a]$ and in $\mathit{support}(\pi)$ of suspensions $\pi{\cdot}X$. The *subterm* of t at a position p is written as $t|_p$. The term obtained from a term s by replacing the subterm at position p by a term t is written as $s[t]_p$. *Action* $\pi{\cdot}t$ and *meta-action* t^π are defined as follows:

$$\begin{array}{rcl@{\qquad}rcl}
\pi{\cdot}a & = & \pi(a) & a^\pi & = & \pi(a) \\
\pi{\cdot}(\pi'{\cdot}X) & = & (\pi \circ \pi'){\cdot}X & (\pi'{\cdot}X)^\pi & = & (\pi \circ \pi' \circ \pi^{-1}){\cdot}X \\
\pi{\cdot}([a]t) & = & [\pi{\cdot}a](\pi{\cdot}t) & ([a]t)^\pi & = & [a^\pi]t^\pi \\
\pi{\cdot}(f\ t) & = & f\ \pi{\cdot}t & (f\ t)^\pi & = & f\ t^\pi \\
\pi{\cdot}\langle t_1, \ldots, t_n \rangle & = & \langle \pi{\cdot}t_1, \ldots, \pi{\cdot}t_n \rangle & \langle t_1, \ldots, t_n \rangle^\pi & = & \langle t_1^\pi, \ldots, t_n^\pi \rangle
\end{array}$$

A *substitution* is a map $\sigma : \mathcal{X} \to \mathcal{T}$ with finite $\mathit{dom}(\sigma) = \{X \in \mathcal{X} \mid \sigma(X) \neq X\}$. The application of a substitution σ on a term t is written as $t\sigma$.

A finite set of pairs $a\#X$ of $a \in \mathcal{A}$ and $X \in \mathcal{X}$ is called a *freshness context*. For a freshness context ∇, $a \in \mathcal{A}$ and $s, t \in \mathcal{T}$, the relations $\nabla \vdash a\#t$ and $\nabla \vdash s \approx_\alpha t$ are defined as follows:

$$\frac{}{\nabla \vdash a\#b}\ a \neq b \qquad \frac{\nabla \vdash a\#t}{\nabla \vdash a\#f\ t} \qquad \frac{\nabla \vdash a\#t_1 \quad \cdots \quad \nabla \vdash a\#t_n}{\nabla \vdash a\#\langle t_1, \ldots, t_n \rangle}$$

$$\frac{}{\nabla \vdash a\#[a]t} \qquad \frac{\nabla \vdash a\#t}{\nabla \vdash a\#[b]t}\ a \neq b \qquad \frac{\pi^{-1}{\cdot}a\#X \in \nabla}{\nabla \vdash a\#\pi{\cdot}X}$$

$$\frac{}{\nabla \vdash a \approx_\alpha a} \qquad \frac{\nabla \vdash t_1 \approx_\alpha s_1 \quad \cdots \quad \nabla \vdash t_n \approx_\alpha s_n}{\nabla \vdash \langle t_1, \ldots, t_n \rangle \approx_\alpha \langle s_1, \ldots, s_n \rangle}$$

$$\frac{\nabla \vdash t \approx_\alpha s}{\nabla \vdash f\ t \approx_\alpha f\ s} \qquad \frac{\nabla \vdash t \approx_\alpha (a\ b){\cdot}s \quad \nabla \vdash a\#s}{\nabla \vdash [a]t \approx_\alpha [b]s}\ a \neq b$$

$$\frac{\nabla \vdash t \approx_\alpha s}{\nabla \vdash [a]t \approx_\alpha [a]s} \qquad \frac{\forall a \in \mathit{ds}(\pi, \pi').\ a\#X \in \nabla}{\nabla \vdash \pi{\cdot}X \approx_\alpha \pi'{\cdot}X}$$

Here $a\#t$ is called a *freshness constraint*, and $s \approx_\alpha t$ an $\boldsymbol{\alpha}$*-equivalence constraint*. For (freshness or $\boldsymbol{\alpha}$-equivalence) constraints $\gamma_1, \ldots, \gamma_n$, we write $\Delta \vdash \gamma_1, \ldots, \gamma_n$ if $\Delta \vdash \gamma_i$ for all $1 \leq i \leq n$. We put $(a\#t)\sigma = a\#t\sigma$ and $(s \approx_\alpha t)\sigma = s\sigma \approx_\alpha t\sigma$.

Nominal unification finds a pair $\langle \Delta, \sigma \rangle$ of a freshness context Δ and a substitution σ such that $\Delta \vdash \gamma_1\sigma, \ldots, \gamma_n\sigma$ from $\mathcal{C} = \{\gamma_1, \ldots, \gamma_n\}$; a most general such pair is an *mgu* of $\mathcal{C}$ [21].

A triple $\nabla \vdash l \to r$ of a freshness context ∇ and $l, r \in \mathcal{T}$ such that l is not a suspension and $\mathcal{X}(\nabla) \cup \mathcal{X}(r) \subseteq \mathcal{X}(l)$ is called a *nominal rewrite rule*, or simply *rewrite rule*. Rewrite rules are identified modulo renaming of term variables. A *nominal rewriting system* (*NRS* for short) is a finite set of rewrite rules. Let $R = \nabla \vdash l \to r$ be a rewrite rule. For a freshness context Δ and $s, t \in \mathcal{T}$, the *rewrite relation* is defined by

$$\Delta \vdash s \to_{\langle R,\pi,p,\sigma\rangle} t \overset{\text{def}}{\iff} \Delta \vdash \nabla^\pi\sigma,\ \Delta \vdash s|_p \approx_\alpha l^\pi\sigma,\ t = s[r^\pi\sigma]_p$$

where $\mathcal{X}(l) \cap (\mathcal{X}(\Delta) \cup \mathcal{X}(s)) = \emptyset$. Here, $\nabla^\pi = \{\pi(a)\#X \mid a\#X \in \nabla\}$. For an NRS $\mathcal{R}$, we write $\Delta \vdash s \to_{\mathcal{R}} t$ if there exist $R \in \mathcal{R}$, π, p and σ such that $\Delta \vdash s \to_{\langle R,\pi,p,\sigma\rangle} t$. We define $\Delta \vdash s_1 \bowtie_1 s_2 \bowtie_2 \cdots (\bowtie_{n-1} s_n)$ $(\bowtie_i \in \{\to_{\mathcal{R}}, \approx_\alpha, \ldots\})$ in the obvious way. $\Delta \vdash s \to_{\mathcal{R}}^* t$ stands for $\Delta \vdash s \to_{\mathcal{R}} \cdots \to_{\mathcal{R}} t$, and $\Delta \vdash s \downarrow_{\approx_\alpha} t$ stands for $\Delta \vdash s \to_{\mathcal{R}}^* \circ \approx_\alpha \circ \leftarrow_{\mathcal{R}}^* t$. An NRS $\mathcal{R}$ is *Church-Rosser modulo* $\approx_\alpha$ if $\Delta \vdash s\ (\leftarrow_{\mathcal{R}} \cup \to_{\mathcal{R}} \cup \approx_\alpha)^*\ t$ implies $\Delta \vdash s \downarrow_{\approx_\alpha} t$. An NRS $\mathcal{R}$ is *terminating* if there is no infinite rewrite sequence $\Delta \vdash s_1 \to_{\mathcal{R}} s_2 \to_{\mathcal{R}} \cdots$.

3 Computing Rewrite Steps and Basic Critical Pairs

A most fundamental ingredient in automation of confluence checking is the computation of rewrite steps, that is, 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 NRS $\mathcal{R}$, 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. Another key ingredient is the computation of basic critical pairs:

Definition 3.1 (Basic critical pair [20]). *Let $R_i = \nabla_i \vdash l_i \to r_i$ $(i = 1, 2)$ be rewrite rules. We assume w.l.o.g. $\mathcal{X}(l_1) \cap \mathcal{X}(l_2) = \emptyset$. Let $\nabla_1 \cup \nabla_2^\pi \cup \{l_1 \approx l_2^\pi|_p\}$ be unifiable for some permutation π and a non-variable position p and let $\langle \Gamma, \sigma \rangle$ be an mgu. Then, $\Gamma \vdash \langle l_2^\pi\sigma[r_1\sigma]_p, r_2^\pi\sigma \rangle$ is called a* basic critical pair *(*BCP *for short) of R_1 and R_2. The set of BCP of rules in $\mathcal{R}$ is denoted by* $\mathrm{BCP}(\mathcal{R})$.

Again, the main challenge for the computation of (representatives of) all BCPs is to find suitable π and σ when fixing $R_1, R_2 \in \mathcal{R}$ and a position p.

Since π is not fixed here, these problems are not computed by nominal unification but by *equivariant nominal unification* [5]. In what follows, we present our formalization of equivariant nominal unification and then explain how BCPs are computed. (The computation of rewrite steps is done by replacing equivariant unification by *equivariant matching*, obtained by adding constraints on instantiation.)

3.1 Equivariant Unification

We extend our language by countably infinite sets $\mathcal{X}_A$ and $\mathcal{X}_P$ of *atom variables* ranged over by $A, B, \ldots$ and *permutation variables* ranged over by $P, Q \ldots$. Elements of $\mathcal{A} \cup \mathcal{X}_A$ are ranged over by $\alpha, \beta, \ldots$ and called *atom expressions*. *Permutation/atomic/term expressions* ($\mathcal{E}_P/\mathcal{E}_A/\mathcal{E}_T$) are generated by the grammar:

$$\begin{array}{l} \Pi, \Psi \in \mathcal{E}_P := P \mid \mathsf{Id} \mid (v\ w) \mid \Pi \circ \Psi \mid \Pi^{-1} \\ v, w \in \mathcal{E}_A \ := \Pi \cdot \alpha \\ S, T \in \mathcal{E}_T \ := v \mid \Pi \cdot X \mid [v]T \mid f\ T \mid \langle T_1, \ldots, T_n \rangle \end{array}$$

Note here that "Id" etc. are not meta-operations but new constructs. For example, we have $(((P \circ Q)^{-1} \cdot A)\ B) \in \mathcal{E}_P$, $(((P \circ Q)^{-1} \cdot A)\ B) \cdot \mathsf{c} \in \mathcal{E}_A$ and $[(((P \circ Q)^{-1} \cdot A)\ B) \cdot \mathsf{c}](\mathtt{f}\ \langle P^{-1} \cdot X, Q^{-1} \cdot \mathsf{c} \rangle) \in \mathcal{E}_T$.

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:

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

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 $[\![[(((P \circ Q)^{-1} \cdot A)\ B) \cdot \mathsf{c}](\mathtt{f}\ \langle P^{-1} \cdot X, Q^{-1} \cdot \mathsf{c} \rangle)]\!]_\theta = [\mathsf{b}](\mathtt{f}\ \langle (\mathsf{a}\ \mathsf{b}) \cdot X, \mathsf{b} \rangle) \in \mathcal{T}$. For a permutation expression $\Pi \in \mathcal{E}_P$ and a term expression $T \in \mathcal{E}_T$, we define action $\Pi \cdot T \in \mathcal{E}_T$ and meta-action $T^\Pi \in \mathcal{E}_T$ as follows:

$$\begin{array}{rclrcl} \Pi \cdot (\Pi' \cdot \alpha) & = & (\Pi \circ \Pi') \cdot \alpha & (\Pi' \cdot \alpha)^\Pi & = & (\Pi \circ \Pi') \cdot \alpha \\ \Pi \cdot (\Pi' \cdot X) & = & (\Pi \circ \Pi') \cdot X & (\Pi' \cdot X)^\Pi & = & (\Pi \circ \Pi' \circ \Pi^{-1}) \cdot X \\ \Pi \cdot ([v]T) & = & [\Pi \cdot v](\Pi \cdot T) & ([v]T)^\Pi & = & [v^\Pi]T^\Pi \\ \Pi \cdot (f\ T) & = & f\ \Pi \cdot T & (f\ T)^\Pi & = & f\ T^\Pi \\ \Pi \cdot \langle T_1, \ldots, T_n \rangle & = & \langle \Pi \cdot T_1, \ldots, \Pi \cdot T_n \rangle & \langle T_1, \ldots, T_n \rangle^\Pi & = & \langle T_1^\Pi, \ldots, T_n^\Pi \rangle \end{array}$$

A *freshness constraint expression* is a pair $v \# T$ of $v \in \mathcal{E}_A$ and $T \in \mathcal{E}_T$ and an $\boldsymbol{\alpha}$*-equivalence constraint expression* is a pair $S \approx T$ of $S, T \in \mathcal{E}_T$. An *equivariant unification problem (EUP)* is a finite set of (freshness or $\boldsymbol{\alpha}$-equivalence) constraint expressions. We put $[\![v \# T]\!]_\theta = [\![v]\!]_\theta \# [\![T]\!]_\theta$ and $[\![S \approx T]\!]_\theta = [\![S]\!]_\theta \approx_\alpha [\![T]\!]_\theta$. 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 $1 \leq i \leq n$. We write $\langle \theta, \sigma, \Delta \rangle \models \mathcal{C}$ if $\langle \theta, \sigma, \Delta \rangle$ is a model of $\mathcal{C}$.

An *answer constraint* is a finite set of expressions of the following forms:

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

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 $\#(X,\Pi,\Pi') \in \mathcal{S}$. For a given EUP $\mathcal{C}$, *equivariant unification* [5] computes a finite set $\mathcal{M} = \mathsf{Sol}(\mathcal{C})$ of answer constraints such that, for any triple $\langle\theta,\sigma,\Delta\rangle$, $\langle\theta,\sigma,\Delta\rangle \models \mathcal{C}$ iff $\exists \mathcal{S} \in \mathcal{M}.\ \langle\theta,\sigma,\Delta\rangle \models \mathcal{S}$.

3.2 Computing Basic Critical Pairs

We now proceed to explain how the representative set of BCPs are computed using equivariant unification, from two given rewrite rules $R_i = \nabla_i \vdash l_i \to r_i$ $(i = 1, 2)$ and a position p. The procedure consists of the following two steps.

1. *Equivariant Unification.* We solve the following EUP:

$$\mathcal{C} = \nabla_1 \cup \nabla_2^P \cup \{l_1 \approx l_2^P|_p\} \cup \{P \cdot a_i \approx A_i \mid a_i \in \mathcal{A}(l_2[\,]_p) \cup \mathcal{A}(r_1) \cup \mathcal{A}(r_2)\}$$

   where $P \in \mathcal{X}_P$, and each A_i is a fresh atom variable. The last component of the union is added to specify $P(a)$ for all a required to construct $l_2^\pi\sigma[r_1\sigma]_p$ and $r_2^\pi\sigma$. If $\mathsf{Sol}(\mathcal{C}) = \emptyset$ then we return the empty set of BCPs.
2. *Instantiation.* For each $\mathcal{S} \in \mathsf{Sol}(\mathcal{C})$, we compute all (representative of) BCPs obtained by models of answer constraints $\mathcal{S} \in \mathsf{Sol}(\mathcal{C})$, more formally, a finite set $T_\mathcal{S}$ representing $\{\Gamma \vdash \langle l_2^{\theta_P(P)}\sigma[r_1\sigma]_p, r_2^{\theta_P(P)}\sigma\rangle \mid \langle\theta,\sigma,\Gamma\rangle \models \mathcal{S}\}$. We obtain a set $\mathrm{BCP}_\mathcal{S}$ of BCPs from $\mathcal{S}$, l_2, r_1 and r_2 by successively instantiating each atom variable and atomic expression $P{\cdot}\alpha$ in $\mathcal{S}$ by all atoms already used and one new fresh atom (as the representative of all other non-used atoms), where any instantiation must satisfy $\Gamma \vdash \gamma$ for all freshness constraints γ obtained from $\alpha \# X \in \mathcal{S}$ and $\#(X,\Pi,\Psi) \in \mathcal{S}$. Note also that due to the form of the input, all occurrences of P in $\#(X,\Pi,\Psi) \in \mathcal{S}$ have the form $P{\cdot}\alpha$. Therefore, any $\#(X,\Pi,\Psi)$ can be replaced with $\{a \# X \mid a \in ds(\Pi,\Psi)\}$ when instantiations are completed. (This is not always possible for general equivariant unification problems e.g. consider $\#(X, (\mathsf{a}\ \mathsf{b}), P)$.) Finally, we put $\mathrm{BCP}_\mathcal{C} = \bigcup_{\mathcal{S}\in\mathsf{Sol}(\mathcal{C})} \mathrm{BCP}_\mathcal{S}$.

Example 3.2. Let $\mathtt{forall} \in \Sigma$ and consider the following NRS:

$$\mathcal{R}_{\mathsf{com}\forall} = \{\ \vdash \mathtt{forall}\ [\mathsf{a}]\mathtt{forall}\ [\mathsf{b}]X \to \mathtt{forall}\ [\mathsf{b}]\mathtt{forall}\ [\mathsf{a}]X\}$$

Consider the overlap at position 11. In the first step, we solve an EUP:

$$\mathcal{C} = \{\mathtt{forall}\ [\mathsf{a}]\mathtt{forall}\ [\mathsf{b}]X \approx (\mathtt{forall}\ [P{\cdot}\mathsf{b}]Y)\} \cup \{P \cdot \mathsf{a} \approx A\}$$

Then we obtain $\mathsf{Sol}(\mathcal{C}) =$

$$\left\{\begin{array}{l} \{Y \mapsto (\mathtt{forall}\ [\mathsf{b}]X), P : \mathsf{a} \mapsto A, P : \mathsf{b} \mapsto \mathsf{a}\}, \\ \{Y \mapsto (\mathtt{forall}\ [\mathsf{a}][(\mathsf{a}\ \mathsf{b})]X), P : \mathsf{a} \mapsto A, P : \mathsf{b} \mapsto \mathsf{b}\}, \\ \{Y \mapsto (\mathtt{forall}\ [(\mathsf{a}\ C){\cdot}\mathsf{b}][(\mathsf{a}\ C)]X), C\#X, C \not\approx \mathsf{a}, C \not\approx \mathsf{b}, P : \mathsf{a} \mapsto A, P : \mathsf{b} \mapsto C\} \end{array}\right\}$$

By instantiating A (by a, b and c) and C (by $\mathsf{a}, \mathsf{b}, \mathsf{c}$ and d) successively, we obtain the following seven BCPs from this overlap: $\mathrm{BCP}_{\mathcal{C}} =$

$$\left\{\begin{array}{l}
\vdash \langle \mathtt{forall[b]forall[b]forall[a]}X,\ \mathtt{forall[a]forall[b]forall[b]}X\rangle \\
\vdash \langle \mathtt{forall[c]forall[b]forall[a]}X,\ \mathtt{forall[a]forall[c]forall[b]}X\rangle \\
\vdash \langle \mathtt{forall[a]forall[b]forall[a]}X,\ \mathtt{forall[b]forall[a]forall[a][(a\ b)]}X\rangle \\
\vdash \langle \mathtt{forall[c]forall[b]forall[a]}X,\ \mathtt{forall[b]forall[c]forall[a][(a\ b)]}X\rangle \\
\mathsf{c}\#X \vdash \langle \mathtt{forall[b]forall[b]forall[a]}X,\ \mathtt{forall[c]forall[b]forall[b](a\ c)\cdot}X\rangle \\
\mathsf{c}\#X \vdash \langle \mathtt{forall[a]forall[b]forall[a]}X,\ \mathtt{forall[c]forall[a]forall[b](a\ c)\cdot}X\rangle \\
\mathsf{d}\#X \vdash \langle \mathtt{forall[c]forall[b]forall[a]}X,\ \mathtt{forall[d]forall[c]forall[b](a\ d)\cdot}X\rangle
\end{array}\right\}$$

4 Proving Confluence Automatically

4.1 Confluence Criteria

We prove (non-)confluence based on the following confluence criteria.

Proposition 4.1 ([19]). *Let $\mathcal{R}$ be an orthogonal NRS that is abstract skeleton preserving (ASP). Then, $\mathcal{R}$ is Church-Rosser modulo $\approx_\alpha$.*

Proposition 4.2 ([20]). *Let $\mathcal{R}$ be a linear uniform NRS. Then $\mathcal{R}$ is Church-Rosser modulo $\approx_\alpha$ if $\Gamma \vdash u \to^{=} \circ \approx_\alpha \circ \leftarrow^{*} v$ and $\Gamma \vdash u \to^{*} \circ \approx_\alpha \circ \leftarrow^{=} v$ for any $\Gamma \vdash \langle u, v\rangle \in \mathrm{BCP}(\mathcal{R})$.*

Proposition 4.3 ([20]). *Let $\mathcal{R}$ be a terminating uniform NRS. Then $\mathcal{R}$ is Church-Rosser modulo $\approx_\alpha$ if and only if $\Gamma \vdash u \downarrow_{\approx_\alpha} v$ for any $\Gamma \vdash \langle u, v\rangle \in \mathrm{BCP}(\mathcal{R})$.*

Proposition 4.4 ([11]). *Let $\mathcal{R}$ be a left-linear uniform NRS. Then $\mathcal{R}$ is Church-Rosser modulo $\approx_\alpha$ if $\Gamma \vdash u \mathrel{-\!\!\!\twoheadrightarrow} \circ \approx_\alpha v$ for any $\Gamma \vdash \langle u, v\rangle \in \mathrm{BCP}_{in}(\mathcal{R})$ and $\Gamma \vdash u \mathrel{-\!\!\!\twoheadrightarrow} \circ \approx_\alpha \circ \leftarrow^{*} v$ for any $\Gamma \vdash \langle u, v\rangle \in \mathrm{BCP}_{out}(\mathcal{R})$.*

Here, an NRS is *orthogonal* if it is left-linear and has no proper BCPs [19]; $\Gamma \vdash s \to^{=} t$ stands for $\Gamma \vdash s \to t$ or $s = t$; $\Gamma \vdash s \mathrel{-\!\!\!\twoheadrightarrow} t$ stands for the parallel rewrite relation [19]; and $\mathrm{BCP}_{in}(\mathcal{R})$ and $\mathrm{BCP}_{out}(\mathcal{R})$ denote the sets of inner and outer BCPs [11], respectively.

The ASP condition and uniformness of NRSs are decidable [6, 19]. To check the joinability conditions in Propositions 4.2 and 4.4, sets $\{w \mid \Gamma \vdash u \to^{=} w\}$ and $\{w \mid \Gamma \vdash u \mathrel{-\!\!\!\twoheadrightarrow} w\}$ are computed using the procedure for computing rewrite steps. For checking confluence criteria of Proposition 4.3, termination checking is required, which we explain in the next subsection.

4.2 Proving Termination

In this subsection, we present a simple technique to show termination of NRSs.

Definition 4.5. *Let Σ be a nominal signature, and $\mathcal{F}$ a arity-fixed first-order signature given by $\mathcal{F} = \{f \mid f \in \Sigma\} \cup \{\diamond, \lambda\} \cup \{\mathsf{pair}_n \mid n \geq 0\}$, where $\diamond$ is of arity 0, λ and all $f \in \Sigma$ are of arity 1, and pair_n is of arity n for each n. We define a translation Φ from nominal terms over Σ to first-order terms over $\mathcal{F}$ (with the set $\mathcal{X}$ of variables) as follows:*

$$\Phi(a) = \diamond \qquad \Phi(\pi{\cdot}X) = X \qquad \Phi([a]t) = \lambda(\Phi(t))$$
$$\Phi(f\ t) = f(\Phi(t)) \qquad \Phi(\langle t_1, \ldots, t_n \rangle) = \mathsf{pair}_n(\Phi(t_1), \ldots, \Phi(t_n))$$

For an NRS $\mathcal{R}$, we define a first-order term rewriting system $\Phi(\mathcal{R})$ by: $\Phi(\mathcal{R}) = \{\Phi(l) \to \Phi(r) \mid \nabla \vdash l \to r \in \mathcal{R}\}$.

Theorem 4.6. *If $\Phi(\mathcal{R})$ is terminating then $\mathcal{R}$ is terminating modulo $\approx_\alpha$.*

Proof. The claim follows from the fact that for any Δ, s, t, (i) $\Delta \vdash s \approx_\alpha t$ implies $\Phi(s) = \Phi(t)$ and (ii) $\Delta \vdash s \to_{\mathcal{R}} t$ implies $\Phi(s) \to_{\Phi(\mathcal{R})} \Phi(t)$. □

Remark 4.7. In [8], nominal terms are given by the following grammar:

$$t, s ::= a \mid \pi{\cdot}X \mid [a]t \mid f(t_1, \ldots, t_n)$$

It is easy to modify the translation Φ to adapt to this definition. In [8, Definition 6], recursive path order on nominal terms for proving termination of "closed rewriting" has been given. It is easy to see that the order can be obtained by combining the translation Φ and recursive path order on first-order terms.

5 Implementation and Experiments

Our tool nrbox (**n**ominal **r**ewriting tool**box**) is implemented in Standard ML of New Jersey[1]. It reads an NRS $\mathcal{R}$ from the input and tries to prove whether it is Church-Rosser modulo $\approx_\alpha$ or not—it prints out "`YES`" ("`NO`") if it successfully proves that $\mathcal{R}$ is (resp. is not) Church-Rosser modulo $\approx_\alpha$ and "`MAYBE`" if it fails to prove or disprove that $\mathcal{R}$ is Church-Rosser modulo $\approx_\alpha$.

The source code of the tool is obtained from `http://www.nue.ie.niigata-u.ac.jp/tools/nrbox/`. It consists of about 4500 lines of code, and roughly one third of the code is devoted to equivariant unification. The format of input NRSs follows a specification bundled in the distribution. To prove the termination of NRSs by the method described in Section 4.2, the tool requires an external termination prover for first-order term rewriting systems.

We have tested our confluence prover with 30 NRSs, collected from the literature [3, 6, 8] and constructed during our studies [11, 18–20]. All tests have been performed in a PC with one 2.50GHz CPU and 4G memory. We have used TTT [13] with 20 sec. timeout as the external termination prover for first-order term rewriting systems.

[1] `http://www.smlnj.org/`

Table 1. Summary of experiments

	NRS	Orth.	Strong	K.-B.	Parallel
1	α-reduction rule ([6] Intro.)	MAYBE	YES	MAYBE	YES
2	Eta: η-reduction rule ([6] Intro.)	YES	YES	YES	YES
3	η-expansion rule ([6] Intro.)	MAYBE	MAYBE	MAYBE	MAYBE
4	$\mathcal{R}^*_\sigma$: subst. for λ with σ_ε (Ex. 43 [6])	MAYBE	MAYBE	YES	YES
5	β-reduction $\{\mathsf{Beta}\} \cup \mathcal{R}^*_\sigma$ (Ex. 43 [6])	MAYBE	MAYBE	MAYBE	MAYBE
6	a fragment of ML (Ex. 43 [6])	MAYBE	MAYBE	MAYBE	MAYBE
7	PNF of FOF (Ex. 44 [6])	MAYBE	MAYBE	NO	MAYBE
8	PNF of FOF with addition (Ex. 44 [6])	MAYBE	MAYBE	NO	MAYBE
9	non-joinable trivial CP (Lem. 56 [6])	MAYBE	MAYBE	MAYBE	MAYBE
10	$\{a\#X \vdash X \to [a]X\}$ (Lem. 56 [6])	MAYBE	MAYBE	MAYBE	MAYBE
11	$\{\mathsf{Eta}, \bot\}$ (Ex. 5 [8])	MAYBE	MAYBE	NO	MAYBE
12	$\{\mathsf{Eta}, \bot\}$ with CP (Ex. 5 [8])	MAYBE	YES	YES	YES
13	summation (Ex. 6 [8])	MAYBE	MAYBE	NO	MAYBE
14	summation with CP (Ex. 6 [8])	MAYBE	MAYBE	YES	MAYBE
15	$\{\vdash f(X) \to [a]X\}$ (Ex. 1.2 [18])	MAYBE	MAYBE	NO	MAYBE
16	$\{a\#X \vdash f(X) \to [a]X\}$ (Ex. 4.7 [18])	YES	YES	YES	YES
17	$\mathcal{R}_\sigma$: subst. for λ with $\sigma_{\mathtt{var}\varepsilon}$ (Ex. 8 [19])	YES	MAYBE	YES	YES
18	β-reduction $\{\mathsf{Beta}\} \cup \mathcal{R}_\sigma$	MAYBE	MAYBE	MAYBE	MAYBE
19	$\beta\eta$-reduction $\{\mathsf{Beta}\} \cup \{\mathsf{Eta}\} \cup \mathcal{R}_\sigma$	MAYBE	MAYBE	MAYBE	MAYBE
20	$\mathcal{R}_{\mathsf{uc}\text{-}\eta}$ (Ex. 17 [19])	MAYBE	MAYBE	MAYBE	MAYBE
21	$\mathcal{R}_{\mathsf{uc}\text{-}\eta\text{-}\mathsf{exp}}$ (Ex. 19 [19])	MAYBE	MAYBE	NO	MAYBE
22	μ-substitution for $\lambda\mu$-term ([15])	YES	MAYBE	YES	YES
23	$\{\vdash f(X) \to f([a]X)\}$ (Ex. 4.3 [3])	MAYBE	MAYBE	MAYBE	MAYBE
24	NNF of $\{\neg, \forall, \wedge\}$-form. with swap (Ex. 5.5 [3])	MAYBE	YES	YES	YES
25	$\mathsf{Com}_\forall$: com. rule for $\forall$ (Ex. 5 [20])	MAYBE	YES	MAYBE	MAYBE
26	PNF of $\{\forall, \wedge\}$-form. (Ex. 7 [20])	MAYBE	MAYBE	NO	MAYBE
27	PNF of $\{\forall, \wedge\}$-form. + $\mathsf{Com}_\forall$ (Ex. 12 [20])	MAYBE	MAYBE	MAYBE	MAYBE
28	NNF of $\{\neg, \forall, \exists\}$-form. (Ex. 29 [20])	MAYBE	MAYBE	YES	MAYBE
29	NNF of FOF	MAYBE	MAYBE	YES	MAYBE
30	NNF of FOF without DNE	YES	YES	YES	YES
	(♯YES, ♯NO)	(5,0)	(7,0)	(11,7)	(9,0)
	$\sum$ time (msec.)	611	1367	4377	2217

Summary of experiments is shown in Table 1. The column below "NRS" shows descriptions of the input NRSs. The columns below "Orth.", "Strong", "K.-B." and "Parallel" show the results of applying the confluence proving methods from Propositions 4.1, 4.2 (with an approximation of $\to^*$ by $\to^=$), 4.3 and 4.4 (with an approximation of $\to^*$ by $\to\!\!\!\!\to$), respectively—YES denotes for the success for proving, NO denotes for the success of disproving, and MAYBE denotes failure. For each method, the last two lines of the table show the number of successes for proving/disproving confluence and the total time for checking all of the examples.

Using the combination of all the methods, our prover succeeded in proving confluence of 13 examples and non-confluence of 7 examples. All details of the

experiments are available on the webpage `http://www.nue.ie.niigata-u.ac.jp/tools/nrbox/experiments/ijcar16/`.

References

1. Confluence competition, `http://coco.nue.riec.tohoku.ac.jp/`
2. Aoto, T., Yoshida, Y., Toyama, Y.: Proving confluence of term rewriting systems automatically. In: Proc. 20th RTA. LNCS, vol. 5595, pp. 93–102. Springer-Verlag (2009)
3. Ayala-Rincón, M., Fernández, M., Gabbay, M.J., Rocha-Oliveira, A.C.: Checking overlaps of nominal rewrite rules. In: Preproc. 10th LSFA. pp. 199–214 (2015)
4. Baader, F., Nipkow, T.: Term Rewriting and All That. Cambridge University Press (1998)
5. Cheney, J.: Equivariant unification. J. of Automated Reasoning 45, 267–300 (2010)
6. Fernández, M., Gabbay, M.J.: Nominal rewriting. Inform. and Comput. 205, 917–965 (2007)
7. Fernández, M., Gabbay, M.J., Mackie, I.: Nominal rewriting systems. In: Proc. 6th PPDP. pp. 108–119. ACM Press (2004)
8. Fernández, M., Rubio, A.: Nominal completion for rewriting systems with binders. In: Proc. ICALP'12. LNCS, vol. 7392, pp. 201–213. Springer-Verlag (2012)
9. Gabbay, M.J., Pitts, A.M.: A new approach to abstract syntax with variable binding. Formal Aspects of Comput. 13, 341–363 (2002)
10. Hirokawa, N., Klein, D.: Saigawa: A confluence tool. In: Proc. of 1st IWC. p. 49 (2012)
11. Kikuchi, K., Aoto, T., Toyama, Y.: Parallel closure theorem for left-linear nominal rewriting systems, `http://www.nue.riec.tohoku.ac.jp/user/kentaro/cr-nominal/pct.pdf`
12. Klop, J.W., van Oostrom, V., van Raamsdonk, F.: Combinatory reduction systems: introduction and survey. Theoret. Comput. Sci. 121, 279–308 (1993)
13. Korp, M., Sternagel, C., Zankl, H., Middeldorp, A.: Tyrolean termination tool 2. In: Proc. 20th RTA. LNCS, vol. 5595, pp. 295–304. Springer-Verlag (2009)
14. Mayr, R., Nipkow, T.: Higher-order rewrite systems and their confluence. Theoret. Comput. Sci. 192, 3–29 (1998)
15. Parigot, M.: $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In: Proc. 3rd LPAR. LNAI, vol. 624, pp. 190–201. Springer-Verlag (1992)
16. Pitts, A.M.: Nominal logic, a first order theory of names and binding. Inform. and Comput. 186, 165–193 (2003)
17. Sternagel, T., Middeldorp, A.: Conditional confluence (system description). In: Proc. Joint 25th RTA and 12th TLCA. LNCS, vol. 8560, pp. 456–465. Springer-Verlag (2014)
18. Suzuki, T., Kikuchi, K., Aoto, T., Toyama, Y.: On confluence of nominal rewriting systems. In: Proc. 16th PPL (2014), in Japanese.
19. Suzuki, T., Kikuchi, K., Aoto, T., Toyama, Y.: Confluence of orthogonal nominal rewriting systems revisited. In: Proc. 26th RTA. LIPIcs, vol. 36, pp. 301–317 (2015)
20. Suzuki, T., Kikuchi, K., Aoto, T., Toyama, Y.: Critical pair analysis in nominal rewriting. In: Proc. 7th SCSS. EPiC, vol. 39, pp. 156–168. EasyChair (2016)
21. Urban, C., Pitts, A.M., Gabbay, M.J.: Nominal unification. Theoret. Comput. Sci. 323, 473–497 (2004)
22. Zankl, H., Felgenhauer, B., Middeldorp, A.: CSI – A confluence tool. In: Proc. 23rd CADE. LNAI, vol. 6803, pp. 499–505. Springer-Verlag (2011)