# Nominal C-Unification

**Mauricio Ayala-Rincón[1], Washington de Carvalho-Segundo[1], Maribel Fernández[2], and Daniele Nantes-Sobrinho[1]**

1 **Departamentos de Matemática e Ciência da Computação**
**Universidade de Brasília, Brazil**
ayala@unb.br,dnantes@mat.unb.br,wtonribeiro@gmail.com
2 **Department of Informatics**
**King's College London, UK**
maribel.fernandez@kcl.ac.uk

## Abstract

Nominal unification is an extension of first-order unification that takes into account the $\alpha$-equivalence relation generated by binding operators, following the nominal approach. We propose a sound and complete procedure for nominal unification with commutative operators, or nominal C-unification for short, which has been formalised in Coq. The procedure transforms nominal C-unification problems into simpler problems whose solutions are generated by algebraic techniques on combinatorics of permutations.

**1998 ACM Subject Classification** D.3.1 Formal Definitions and Theory; F.4.3 Formal Languages

**Keywords and phrases** Nominal logic; Nominal Unification, Unification Modulo Commutativity

## 1 Introduction

Unification, where the goal is to solve equations between first-order terms, is a key notion in logic programming systems, type inference algorithms, protocol analysis tools, theorem provers, etc. Solutions to unification problems are represented by substitutions that map variables $(X, Y, \dots)$ to terms.

When terms include binding operators, a more general notion of unification is needed: unification modulo $\alpha$-equivalence. In this paper, we follow the nominal approach to the specification of binding operators [17, 25, 22], where the syntax of terms includes, in addition to variables, also *atoms* $(a, b, \dots)$, which can be abstracted, and $\alpha$-equivalence is axiomatised by means of a *freshness relation* $a\#t$ and *name-swappings* $(a\,b)$. For example, the first-order logic formula $\forall a.a \geq 0$ can be written as a nominal term $\forall([a]geq(a, 0))$, using an operator $\forall$, an abstraction for the atom $a$, an operator $geq$, and the constant 0. Nominal unification [25] is the problem of solving equations between nominal terms modulo $\alpha$-equivalence; it is a decidable problem and efficient nominal unification algorithms are available [9, 7, 21], that compute solutions consisting of *freshness contexts* (containing freshness constraints of the form $a\#X$) and substitutions.

In many applications, operators obey equational axioms. Nominal reasoning and unification have been extended to deal with equational theories presented by rewrite rules (see, e.g., [15, 14, 4]) or defined by equational axioms (see, e.g., [12, 16]). The case of associative and commutative nominal theories was considered in [3], where a parametric $\{\alpha, AC\}$-equivalence relation was formalised in Coq. However, only equational deduction was considered (not unification). In this paper, we study nominal C-unification.

**Contributions:** We present a nominal C-unification algorithm, based on a set of simplification rules, which transforms a given *nominal C-unification problem* $\langle \Delta, Q \rangle$, where $\Delta$ is


Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a freshness context and $Q$ a set of freshness constraints $a\#_? s$ and equations $s \approx_? t$, into a finite set of triples of the form $\langle \nabla, \sigma, P \rangle$, consisting of a freshness context $\nabla$, a substitution $\sigma$ and a set of fixpoint equations $P$, of the form $\pi.X \approx_? X$. The simplifications are based on the deduction rules for freshness and $\alpha$-C-equivalence (denoted as $\approx_{\{\alpha,C\}}$). We show that in general there is no finitary representation of solutions using only freshness contexts and substitutions, hence, in general, a nominal C-unification problem may have a potentially infinite set of independent most general unifiers (unlike nominal unification, which is unitary, and C-unification, which is finitary). We present a procedure that enumerates the possibly infinite set of combinatorial solutions of fixpoint equations. By combining the simplification and enumeration methods, we obtain a nominal C-unification procedure in two phases: a *simplification phase*, which outputs a finite set of most general solutions that may include fixpoint constraints, and a *generation phase*, which eliminates the fixpoint constraints.

The simplification algorithm was formalised in Coq[1], as follows:

**Soundness of $\approx_{\{\alpha,C\}}$**: The specification in [3] was extended to include rules to compare terms with C-function symbols. We verified that the nominal $\alpha$-relation with C-function symbols, $\approx_{\{\alpha,C\}}$, is indeed an equivalence relation.

**Nominal C-unification problems**: Triples $\mathcal{P} = \langle \nabla, \sigma, P \rangle$ as above, but in which the set $P$ consists of freshness constraints $a\#_? s$ and equations $s \approx_? t$ were specified. Two sets of simplification rules over triples, generating the relations $\Rightarrow_{\#}$ and $\Rightarrow_{\approx}$, were inductively defined. A nominal C-unification problem $\mathcal{Q} = \langle \Delta, Q \rangle$ is associated with an initial triple $\langle \Delta, id, Q \rangle$, where $id$ is the identity substitution.

**Termination**: Termination of the rewriting systems $\Rightarrow_{\#}$ and $\Rightarrow_{\approx}$ was then formalised.

**Soundness**: It is proved that if either $\mathcal{P} \Rightarrow_{\#} \mathcal{P}'$ or $\mathcal{P} \Rightarrow_{\approx} \mathcal{P}'$, then any solution of $\mathcal{P}'$ is also a solution of $\mathcal{P}$;

**Characterisation of successful leaves**: A problem $\langle \Delta, Q \rangle$ has solutions if and only if there exist $\Rightarrow_{\approx}$-normal forms of $\langle \Delta, id, Q \rangle$, that have $\Rightarrow_{\#}$-normal forms, where the third component of the triple contains only fixpoint equations.

**Completeness**: The simplification algorithm, which consists of normalisation by $\Rightarrow_{\approx}$ followed by normalisation by $\Rightarrow_{\#}$, was formalised to be complete by proving first, that if $\mathcal{P} \Rightarrow_{\#} \mathcal{Q}$ then solutions of $\mathcal{P}$ are exactly the solutions of $\mathcal{Q}$; and second that, for any solution $u$ of $\mathcal{P}$ there exists $\mathcal{Q}$ such that $\mathcal{P} \Rightarrow_{\approx} \mathcal{Q}$ and $u$ is also a solution of $\mathcal{Q}$.

**Related work**: Several extensions of the nominal unification algorithm have been defined, in addition to the equational extensions already mentioned. Closely related to our work is the algorithm presented by Kutsia et al [20] to solve problems with recursive *let* operators and higher-order expressions. Using this algorithm they obtain an algorithm to solve nominal C-unification problems, and prove that nominal C-unification is NP-complete. Our approach is different: we formalise a direct algorithm to compute solutions, using rule-based transformations, and apply algebraic techniques to solve fixpoint equations.

Recently, Aoto and Kikuchi [1] proposed a rule-based procedure for nominal equivariant unication [11], an extension of nominal unification that is useful in confluence analysis of nominal rewriting systems [2, 13].

Furthermore, several formalisations and implementations of the nominal unification algorithm are available. For example formalisations of its soundness and completeness can be found in Urban et al [25, 24], Ayala-Rincón et al [5], and Kumar and Norrish [19] using, respectively, the proof assistants Isabelle/HOL, PVS and HOL4. Urban and Cheney used

[1] Available in `https://github.com/wtonribeiro/nominal-ac`

a nominal unification algorithm to develop a Prolog-like language called $\alpha$-Prolog [10]. An implementation in Maude using term graphs [8] is also available.

Syntactic unification with commutativity operators, or just C-unification, is an NP-complete problem and its solutions can be finitely generated [18, 23]. Since C-unification problems are a particular case of nominal C-unification problems, our simplification algorithm, formalised in Coq, is also a formalisation of the C-unification algorithm.

**Organisation**: We present basic concepts and notations in Section 2. We introduce the equational and freshness inference rules for nominal C-unification in Section 3. These inference rules transform unification problems into problems that consist only of fixpoint equations, whenever possible. We then show in Section 4 how fixpoint equations are solved and how these solutions can be combined in order to produce solutions for the initial unification problem. Section 5 concludes the paper and describes future work. Appendices with details of the proofs and the combination of solutions are included.

## 2 Background

Consider countable disjoint sets of variables $\mathcal{X} := \{X, Y, Z, \cdots\}$ and atoms $\mathcal{A} := \{a, b, c, \cdots\}$.

▶ **Definition 1** (Permutations). A *permutation* $\pi$ is a bijection on $\mathcal{A}$ with a finite *domain*, where the domain (i.e., the *support*) of $\pi$ is the set $dom(\pi) := \{a \in \mathcal{A} \mid \pi \cdot a \neq a\}$.

Permutations can be represented by lists of *swappings*, which are pairs of different atoms $(a\, b)$; hence a *permutation* $\pi$ is a finite list of the form $(a_1\, b_1) :: \ldots :: (a_n\, b_n) :: nil$, where *nil* denotes the identity permutation; concatenation is denoted by $\oplus$.

We will assume as in [3] countable sets of function symbols with different equational properties such as associativity, commutativity, idempotence, etc. Function symbols have superscripts that indicate their equational properties; thus, $f_k^C$ will denote the $k^{th}$ function symbol that is commutative and $f_j^\emptyset$ the $j^{th}$ function symbol without any equational property.

▶ **Definition 2** (Nominal grammar). Nominal terms are generated by the following grammar.

$$s, t := \langle\rangle \mid \bar{a} \mid [a]t \mid \langle s, t\rangle \mid f_k^E\, t \mid \pi.X$$

$\langle\rangle$ denotes the *unit* (that is the empty tuple), $\bar{a}$ denotes an *atom term*, $[a]t$ denotes an *abstraction* of the atom $a$ over the term $t$, $\langle s, t\rangle$ denotes a *pair*, $f_k^E\, t$ the *application* of $f_k^E$ to $t$ and, $\pi.X$ a *moderated variable* or *suspension*. Suspensions of the form $\mathtt{id}.X$ will be represented just by $X$.

The set of variables occurring in a term $t$ will be denoted as $Var(t)$. This notation extends to a set $S$ of terms in the natural way: $Var(S) = \bigcup_{t\in S} Var(t)$.

▶ **Definition 3** (Permutation action).

$$nil \cdot a := a$$

$$((c\ d) :: \pi') \cdot a := \begin{cases} \texttt{if } c{=}a \texttt{ then } \pi'\cdot d \\ \texttt{else} \begin{cases} \texttt{if } d{=}a \texttt{ then } \pi'\cdot c \\ \texttt{else } \pi'\cdot a \end{cases} \end{cases}$$

$$\pi \cdot t := \begin{cases} \pi \cdot \langle\rangle & \to & \langle\rangle \\ \pi \cdot \bar{a} & \to & \overline{\pi \cdot a} \\ \pi \cdot f_k^E\, t & \to & f_k^E\, (\pi \cdot t) \\ \pi \cdot \langle u, v\rangle & \to & \langle \pi \cdot u, \pi \cdot v\rangle \\ \pi \cdot ([a]t) & \to & [\pi \cdot a](\pi \cdot t) \\ \pi \cdot (\pi' \,.\, X) & \to & (\pi' \oplus \pi) \,.\, X \end{cases}$$

▶ Remark. Notice that according to the definition of the action of a permutation over atoms, the composition of permutations $\pi$ and $\pi'$, usually denoted as $\pi \circ \pi'$, corresponds to the append $\pi' \oplus \pi$. Also notice that $\pi' \oplus \pi \cdot t = \pi \cdot (\pi' \cdot t)$.

▶ **Definition 4** (Difference set). The *difference set* between two permutations $\pi$ and $\pi'$ is the set of atoms where the action of $\pi$ and $\pi'$ differs: $ds(\pi, \pi') := \{a \in \mathcal{A} \mid \pi \cdot a \neq \pi' \cdot a\}$.

▶ **Definition 5** (Substitution). A *substitution* $\sigma$ is a mapping from variables to terms such that $X \neq X\sigma$ only for a finite set of variables. This set is called the *domain* of $\sigma$ and is denoted by $dom(\sigma)$. For $X \in dom(\sigma)$, $X\sigma$ is called the *image* of $X$ by $\sigma$. Define the image of $\sigma$ as $im(\sigma) = \{X\sigma \mid X \in dom(\sigma)\}$. The set of variables occurring in the image of $\sigma$ is then $Var(im(\sigma))$. A substitution $\sigma$ with $dom(\sigma) := \{X_0, \cdots, X_n\}$ can be represented as a set of *binds* in the form $\{X_0/t_0, \cdots, X_n/t_n\}$, where for $0 \leq i \leq n$, $X_i\sigma = t_i$.

▶ **Definition 6** (Substitution action). The *action of a substitution* $\sigma$ on a term $t$, denoted $t\sigma$, is defined recursively as follows:

$$\begin{array}{llllllllll} \langle\rangle\sigma & := & \langle\rangle & & \overline{a}\sigma & := & \overline{a} & & (f_k^E\, t)\sigma & := & f_k^E\, t\sigma \\ \langle s,t\rangle\sigma & := & \langle s\sigma, t\sigma\rangle & & ([a]t)\sigma & := & [a]t\sigma & & (\pi.X)\sigma & := & \pi \cdot X\sigma \end{array}$$

The following well-known result can be proved by induction on the structure of terms.

▶ **Lemma 7** (Substitutions and Permutations Commute). $(\pi \cdot t)\sigma = \pi \cdot (t\sigma)$

The inference rules defining freshness and $\alpha$-equivalence are given in Figures 1 and 2. The symbols $\nabla$ and $\Delta$ are used to denote *freshness contexts* that are sets of constraints of the form $a\#X$, meaning that the atom $a$ is fresh in $X$. The domain of a freshness context $dom(\Delta)$ is the set of atoms appearing in it; $\Delta|_X$ denotes the restriction of $\Delta$ to the freshness constraints on $X$: $\{a\#X \mid a\#X \in \Delta\}$. The rules in Fig. 1 are used to check if an atom $a$ is fresh in a nominal term $t$ under a freshness context $\nabla$, also denoted as $\nabla \vdash a\#t$. The rules in Fig. 2 are used to check if two nominal terms $s$ and $t$ are $\alpha$-equivalent under some freshness context $\nabla$, written as $\nabla \vdash s \approx_\alpha t$. These rules use the inference system for freshness constraints: specifically freshness constraints are used in rule ($\approx_\alpha$ [**ab**]).

▶ Remark. $dom(\pi)\#X$ and $ds(\pi,\pi')\#X$ denote, respectively, the sets $\{a\#X \mid a \in dom(\pi)\}$ and $\{a\#X \mid a \in ds(\pi,\pi')\}$.

$$\frac{}{\nabla \vdash a\,\#\,\langle\rangle}\,(\#\langle\rangle) \quad \frac{}{\nabla \vdash a\,\#\,\overline{b}}\,(\#\,\mathbf{atom}) \quad \frac{\nabla \vdash a\,\#\,t}{\nabla \vdash a\,\#\,f_k^E\,t}\,(\#\,\mathbf{app}) \quad \frac{}{\nabla \vdash a\,\#\,[a]t}\,(\#\,\mathbf{a[a]})$$

$$\frac{\nabla \vdash a\,\#\,t}{\nabla \vdash a\,\#\,[b]t}\,(\#\,\mathbf{a[b]}) \quad \frac{(\pi^{-1}\cdot a\#X) \in \nabla}{\nabla \vdash a\,\#\,\pi.X}\,(\#\,\mathbf{var}) \quad \frac{\nabla \vdash a\,\#\,s \;\; \nabla \vdash a\,\#\,t}{\nabla \vdash a\,\#\,\langle s,t\rangle}\,(\#\,\mathbf{pair})$$

**Figure 1** Rules for the freshness relation

$$\frac{}{\nabla \vdash \langle\rangle \approx_\alpha \langle\rangle}\,(\approx_\alpha \langle\rangle) \quad \frac{}{\nabla \vdash \overline{a} \approx_\alpha \overline{a}}\,(\approx_\alpha\,\mathbf{atom}) \quad \frac{\nabla \vdash s \approx_\alpha t}{\nabla \vdash f_k^E\, s \approx_\alpha f_k^E\, t}\,(\approx_\alpha\,\mathbf{app})$$

$$\frac{\nabla \vdash s \approx_\alpha t}{\nabla \vdash [a]s \approx_\alpha [a]t}\,(\approx_\alpha\,[\mathbf{aa}]) \quad \frac{\nabla \vdash s \approx_\alpha (a\,b)\cdot t \;\; \nabla \vdash a\,\#\,t}{\nabla \vdash [a]s \approx_\alpha [b]t}\,(\approx_\alpha\,[\mathbf{ab}])$$

$$\frac{ds(\pi,\pi')\#X \subseteq \nabla}{\nabla \vdash \pi.X \approx_\alpha \pi'.X}\,(\approx_\alpha\,\mathbf{var}) \quad \frac{\nabla \vdash s_0 \approx_\alpha t_0 \;\; \nabla \vdash s_1 \approx_\alpha t_1}{\nabla \vdash \langle s_0,t_0\rangle \approx_\alpha \langle s_1,t_1\rangle}\,(\approx_\alpha\,\mathbf{pair})$$

**Figure 2** Rules for the relation $\approx_\alpha$

Key properties of the nominal freshness and $\alpha$-equivalence relations have been extensively explored in previous works [3, 5, 24, 25], among them we have *freshness preservation*: If $\nabla \vdash a\,\#\,s$ and $\nabla \vdash s \approx_\alpha t$, then $\nabla \vdash a\,\#\,t$; *equivariance*: for all permutations $\pi$, if $\nabla \vdash s \approx_\alpha t$ then $\nabla \vdash \pi \cdot s \approx_\alpha \pi \cdot t$; and, **equivalence**: $\_ \vdash \_ \approx_\alpha \_$ is an equivalence relation.

## 2.1 The relation $\approx_{\{\alpha,C\}}$ as an extension of $\approx_\alpha$

In [3], the relation $\approx_\alpha$ was extended to deal with associative and commutative theories. Here we will consider $\alpha$-equivalence modulo commutativity, denoted $\approx_{\{\alpha,C\}}$. This means that some function symbols in our syntax are commutative, and therefore the rule for function application ($\approx_\alpha$ **app**) in Fig. 2 should be replaced by the rules in Fig. 3.

$$\frac{\nabla \vdash s \approx_{\{\alpha,C\}} t}{\nabla \vdash f_k^E\, s \approx_{\{\alpha,C\}} f_k^E\, t}\,,\quad E \neq C \text{ or both } s \text{ and } t \text{ are not pairs}\quad (\approx_{\{\alpha,\mathbf{C}\}} \mathbf{app})$$

$$\frac{\nabla \vdash s_0 \approx_{\{\alpha,C\}} t_i,\quad \nabla \vdash s_1 \approx_{\{\alpha,C\}} t_{(i+1)\,mod\,2}}{\nabla \vdash f_k^C\,\langle s_0, s_1\rangle \approx_{\{\alpha,C\}} f_k^C\,\langle t_0, t_1\rangle}\,,\quad i = 0,1\quad (\approx_{\{\alpha,\mathbf{C}\}} \mathbf{C})$$

**Figure 3** Additional rules for $\{\alpha, C\}$-equivalence

▶ Remark. In the following, for brevity, instead $(i+1)\,mod\,2$ we will write only $(i+1)$.

▶ Remark. Nominal function symbols have no fixed arity according to Definition 2 (any function symbol can apply to any term). The syntax in our Coq formalisation permits the application of non commutative function symbols to any term, but commutative symbols apply only to tuples.

The following properties have been formalised in Coq as simple adaptations of the formalisations given in [3].

▶ **Lemma 8** (Inversion). *The inference rules of* $\approx_{\{\alpha,C\}}$ *can be* reversed*:*

1. $\nabla \vdash f_k^E\, s \approx_{\{\alpha,C\}} f_k^E\, t$, *where* $E \neq C$ *or both* $s$ *and* $t$ *are not pairs, implies* $\nabla \vdash s \approx_{\{\alpha,C\}} t$
2. $\nabla \vdash f_k^C\,\langle s_0, s_1\rangle \approx_{\{\alpha,C\}} f_k^C\,\langle t_0, t_1\rangle$ *implies* $\nabla \vdash s_0 \approx_{\{\alpha,C\}} t_0$ *and* $\nabla \vdash s_1 \approx_{\{\alpha,C\}} t_1$, *or* $\nabla \vdash s_0 \approx_{\{\alpha,C\}} t_1$ *and* $\nabla \vdash s_1 \approx_{\{\alpha,C\}} t_0$
3. $\nabla \vdash \langle s_0, s_1\rangle \approx_{\{\alpha,C\}} \langle t_0, t_1\rangle$ *implies* $\nabla \vdash s_0 \approx_{\{\alpha,C\}} t_0$ *and* $\nabla \vdash s_1 \approx_{\{\alpha,C\}} t_1$
4. $\nabla \vdash [a]s \approx_{\{\alpha,C\}} [a]t$ *implies* $\nabla \vdash s \approx_{\{\alpha,C\}} t$
5. $\nabla \vdash [a]s \approx_{\{\alpha,C\}} [b]t$ *implies* $\nabla \vdash s \approx_{\{\alpha,C\}} (a\,b)\cdot t$ *and* $\nabla \vdash a \# t$
6. $\nabla \vdash \pi.X \approx_{\{\alpha,C\}} \pi'.X$ *implies* $ds(\pi,\pi')\#X \subseteq \nabla$

▶ **Lemma 9** (Freshness preservation). *If* $\nabla \vdash a \# s$ *and* $\nabla \vdash s \approx_{\{\alpha,C\}} t$ *then* $\nabla \vdash a \# t$.

▶ **Lemma 10** (Intermediate transitivity for $\approx_{\{\alpha,C\}}$ with $\approx_\alpha$). *If* $\nabla \vdash s \approx_{\{\alpha,C\}} t$ *and* $\nabla \vdash t \approx_\alpha u$ *then* $\nabla \vdash s \approx_{\{\alpha,C\}} u$.

▶ **Lemma 11** (Equivariance). $\nabla \vdash \pi \cdot s \approx_{\{\alpha,C\}} \pi \cdot t$ *whenever* $\nabla \vdash s \approx_{\{\alpha,C\}} t$.

▶ **Lemma 12** (Equivalence). $\_ \vdash \_ \approx_{\{\alpha,C\}} \_$ *is an equivalence relation.*

# 3 A nominal C-unification algorithm

We present inference rules that transform a nominal unification problem with commutative function symbols into a problem that consist exclusively of equations of the form $\pi.X \approx_?^{} X$, together with a substitution and a set of freshness constraints.

▶ **Definition 13** (Unification problem). A *unification problem* is a pair $\langle \nabla, P\rangle$, where $\nabla$ is a *freshness context* and $P$ is a finite set of *equations* and *freshness constraints* of the form $s \approx_? t$ and $a\#_? s$, respectively, where $\approx_?$ is symmetric, $s$ and $t$ are terms and $a$ is an atom. Nominal terms in the equations preserve the syntactic restriction that commutative symbols are only applied to tuples.

▶ Remark. Equations of the form $\pi.X \approx_? X$ are called *fixpoint* equations. We will denote by $P_\approx, P_\#, P_{\text{fp}_\approx}$ and $P_{\text{nfp}_\approx}$ the sets of equations, freshness constraints, fixpoint equations and non fixpoint equations in the set $P$ of a unification problem $\langle \nabla, P\rangle$.

We design a nominal C-unification algorithm using one set of transformation rules to deal with equations (Figure 4) and another set of rules to deal with freshness constraints and contexts (Figure 5). These rules act over triples of the form $\langle \nabla, \sigma, P\rangle$, where $\sigma$ is a substitution. The triple that will be associated by default with a unification problem $\langle \nabla, P\rangle$ is $\langle \nabla, id, P\rangle$. We will use calligraphic uppercase letters (e.g., $\mathcal{P}, \mathcal{Q}, \mathcal{R}$, etc) to denote triples.

▶ Remark. Let $\nabla$ and $\nabla'$ be freshness contexts and $\sigma$ and $\sigma'$ be substitutions.
- $\nabla' \vdash \nabla\sigma$ denotes that $\nabla' \vdash a \# X\sigma$ holds for each $(a\#X) \in \nabla$;
- $\nabla \vdash \sigma \approx \sigma'$ denotes that $\nabla \vdash X\sigma \approx_{\{\alpha,C\}} X\sigma'$ for all $X$ (in $dom(\sigma) \cup dom(\sigma')$).

▶ **Definition 14** (Solution for a triple or problem)**.** A *solution* for a triple $\mathcal{P} = \langle \Delta, \delta, P\rangle$ is a pair $\langle \nabla, \sigma\rangle$, where the following conditions are satisfied:

**1.** $\nabla \vdash \Delta\sigma$;
**2.** if $a\#_? t \in P$ then $\nabla \vdash a \# t\sigma$;
**3.** if $s \approx_? t \in P$ then $\nabla \vdash s\sigma \approx_{\{\alpha,C\}} t\sigma$;
**4.** there exists $\lambda$ such that $\nabla \vdash \delta\lambda \approx \sigma$.

A *solution for a unification problem* $\langle \Delta, P\rangle$ is a solution for the associated triple $\langle \Delta, id, P\rangle$. The *solution set* for a problem or triple $\mathcal{P}$ is denoted by $\mathcal{U}_C(\mathcal{P})$.

▶ **Definition 15** (More general solution and complete set of solutions)**.** For $\langle \nabla, \sigma\rangle$ and $\langle \nabla', \sigma'\rangle$ in $\mathcal{U}_C(\mathcal{P})$, we say that $\langle \nabla, \sigma\rangle$ is *more general* than $\langle \nabla', \sigma'\rangle$, denoted $\langle \nabla, \sigma\rangle \dot{\preccurlyeq} \langle \nabla', \sigma'\rangle$, if $\nabla' \vdash \nabla\sigma'$ and there exists a substitution $\lambda$ satisfying $\nabla' \vdash \sigma\lambda \approx \sigma'$. A subset $\mathcal{V}$ of $\mathcal{U}_C(\mathcal{P})$ is said to be a *complete set of solutions* of $\mathcal{P}$ if for all $\langle \nabla', \sigma'\rangle \in \mathcal{U}_C(\mathcal{P})$, there exists $\langle \nabla, \sigma\rangle$ in $\mathcal{V}$ that is more general than $\langle \nabla', \sigma'\rangle$.

We will denote the set of variables occurring in the set $P$ of a problem $\langle \nabla, P\rangle$ or triple $\mathcal{P} = \langle \nabla, \sigma, P\rangle$ as $Var(P)$. We also will write $Var(\mathcal{P})$ to denote this set.

The unification algorithm proceeds by simplification. Derivation with rules of Fig. 4 is denoted by $\Rightarrow_\approx$; thus, $\langle \nabla, \sigma, P\rangle \Rightarrow_\approx \langle \nabla, \sigma', P'\rangle$ means that the second triple is obtained from the first one by application of one rule. We will use the standard rewriting nomenclature, e.g., we will say that $\mathcal{P}$ is a *normal form* or *irreducible* by $\Rightarrow_\approx$, denoted by $\Rightarrow_\approx$-*nf*, whenever there is no $\mathcal{Q}$ such that $\mathcal{P} \Rightarrow_\approx \mathcal{Q}$; $\Rightarrow_\approx^*$ and $\Rightarrow_\approx^+$ denote respectively derivations in zero or more and one or more applications of the rules in Fig. 4.

Derivation with rules of Figure 5 is denoted by $\Rightarrow_\#$.

The only rule that can generate branches is ($\approx_?$ **C**). This is really an abbreviation for two rules providing the different forms in which one can relate the arguments $s$ and $t$ in an equation $f_k^C\, s \approx_? f_k^C\, t$ for a commutative function symbol ($s$, $t$ are tuples, by the syntactic restriction in Definition 13): either $\langle s_0, s_1\rangle \approx_? \langle t_0, t_1\rangle$ or $\langle s_0, s_1\rangle \approx_? \langle t_1, t_0\rangle$.

Also, notice that the syntactic restriction on arguments of commutative symbols being only tuples, is not crucial since any equation of the form $f_k^C \pi.X \approx_? t$ can be translated into an equation of form $f_k^C\langle \pi.X_1, \pi.X_2\rangle \approx_? t$, where $X_1$ and $X_2$ are new variables and $\nabla$ is extended to $\nabla'$ in such a way that both $X_1$ and $X_2$ inherit all freshness constraint of $X$ in $\nabla$: $\nabla' = \nabla \cup \{a\#X_i \mid i = 1, 2, \text{ and } a\#X \in \nabla\}$.

In the rule ($\approx_?$ **inst**) the inclusion of new constraints in the problem, given in the set $\bigcup_{\substack{Y \in dom(\sigma'),\\ a\#Y \in \nabla}} \{a\#_? Y\sigma'\}$ is necessary because one needs to guarantee that the new substitution $\sigma'$ is *compatible* with the freshness context $\nabla$.

▶ **Definition 16** (Set of $\Rightarrow_\approx$ and $\Rightarrow_\#$-normal forms)**.** We denote by $\mathcal{P}_{\Rightarrow_\approx}$ (resp. $\mathcal{P}_{\Rightarrow_\#}$) the set of normal forms of $\mathcal{P}$ with respect to $\Rightarrow_\approx$ (resp. $\Rightarrow_\#$).

$$\frac{\langle \nabla, \sigma, P \uplus \{s \approx_? s\}\rangle}{\langle \nabla, \sigma, P\rangle}\,(\approx_? \textbf{refl}) \qquad \frac{\langle \nabla, \sigma, P \uplus \{\langle s_1, t_1\rangle \approx_? \langle s_2, t_2\rangle\}\rangle}{\langle \nabla, \sigma, P \cup \{s_1 \approx_? s_2, t_1 \approx_? t_2\}\rangle}\,(\approx_? \textbf{pair})$$

$$\frac{\langle \nabla, \sigma, P \uplus \{f_k^E\, s \approx_? f_k^E\, t\}\rangle}{\langle \nabla, \sigma, P \cup \{s \approx_? t\}\rangle}\text{, if } E \neq C\ (\approx_? \textbf{app})$$

$$\frac{\langle \nabla, \sigma, P \uplus \{f_k^C\, s \approx_? f_k^C\, t\}\rangle}{\langle \nabla, \sigma, P \cup \{s \approx_? v\}\rangle}\text{, } \left\{ \begin{array}{l} \text{where } s = \langle s_0, s_1\rangle \text{ and } t = \langle t_0, t_1\rangle \\ v = \langle t_i, t_{i+1}\rangle, i = 0, 1, \end{array} \right\} (\approx_? \textbf{C})$$

$$\frac{\langle \nabla, \sigma, P \uplus \{[a]s \approx_? [a]t\}\rangle}{\langle \nabla, \sigma, P \cup \{s \approx_? t\}\rangle}\,(\approx_? [\textbf{aa}]) \qquad \frac{\langle \nabla, \sigma, P \uplus \{[a]s \approx_? [b]t\}\rangle}{\langle \nabla, \sigma, P \cup \{s \approx_? (a\,b)\,t, a\#_? t\}\rangle}\,(\approx_? [\textbf{ab}])$$

$$\frac{\langle \nabla, \sigma, P \uplus \{\pi.X \approx_? t\}\rangle \quad \text{let } \sigma' := \sigma\{X/\pi^{-1} \cdot t\}}{\left\langle \nabla, \sigma', P\{X/\pi^{-1} \cdot t\} \;\cup \bigcup_{\substack{Y \in dom(\sigma'),\\ a\#Y \in \nabla}} \{a\#_? Y\sigma'\} \right\rangle}\text{, if } X \notin Var(t)\ (\approx_? \textbf{inst})$$

$$\frac{\langle \nabla, \sigma, P \uplus \{\pi.X \approx_? \pi'.X\}\rangle}{\langle \nabla, \sigma, P \cup \{\pi \oplus (\pi')^{-1}.X \approx_? X\}\rangle}\text{, if } \pi' \neq \mathtt{id}\ (\approx_? \textbf{inv})$$

**Figure 4** Reduction rules for equational problems

$$\frac{\langle \nabla, \sigma, P \uplus \{a\#_?\langle\rangle\}\rangle}{\langle \nabla, \sigma, P\rangle}\,(\#_?\langle\rangle) \qquad \frac{\langle \nabla, \sigma, P \uplus \{a\#_? \bar{b}\}\rangle}{\langle \nabla, \sigma, P\rangle}\,(\#_? \textbf{a}\bar{\textbf{b}})$$

$$\frac{\langle \nabla, \sigma, P \uplus \{a\#_? f\, t\}\rangle}{\langle \nabla, \sigma, P \cup \{a\#_? t\}\rangle}\,(\#_? \textbf{app}) \qquad \frac{\langle \nabla, \sigma, P \uplus \{a\#_? [a]t\}\rangle}{\langle \nabla, \sigma, P\rangle}\,(\#_? \textbf{a}[\textbf{a}])$$

$$\frac{\langle \nabla, \sigma, P \uplus \{a\#_? [b]t\}\rangle}{\langle \nabla, \sigma, P \cup \{a\#_? t\}\rangle}\,(\#_? \textbf{a}[\textbf{b}]) \qquad \frac{\langle \nabla, \sigma, P \uplus \{a\#_? \pi.X\}\rangle}{\langle \{(\pi^{-1} \cdot a)\#X\} \cup \nabla, \sigma, P\rangle}\,(\#_? \textbf{var})$$

$$\frac{\langle \nabla, \sigma, P \uplus \{a\#_? \langle s, t\rangle\}\rangle}{\langle \nabla, \sigma, P \cup \{a\#_? s, a\#_? t\}\rangle}\,(\langle\#\rangle \textbf{pair})$$

**Figure 5** Reduction rules for freshness problems

▶ **Definition 17** (Fail and success for $\Rightarrow_{\approx}$)**.** Let $\mathcal{P}$ be a triple, such that the rules in Fig. 4 give rise to a normal form $\langle \nabla, \sigma, P\rangle$. The rules in Fig. 4 are said to *fail* if $P$ contains non fixpoint equations. Otherwise $\langle \nabla, \sigma, P\rangle$ is called a *successful* triple regarding $\Rightarrow_{\approx}$ (i.e., in a successful triple, $P$ consists only of fixpoint equations and, possibly, freshness constraints).

The rules in Fig. 5 will only be applied to successful triples regarding $\Rightarrow_{\approx}$.

▶ **Definition 18** (Fail and success for $\Rightarrow_{\#}$)**.** Let $\mathcal{Q} = \langle \nabla, \sigma, Q\rangle$ be a successful triple regarding $\Rightarrow_{\approx}$, and let $\mathcal{Q}' = \langle \nabla', \sigma, Q'\rangle$ be its normal form using the rules in Fig. 5, that is $\mathcal{Q} \Rightarrow_{\#}^{*} \mathcal{Q}'$ and $\mathcal{Q}'$ is in $\mathcal{Q}_{\Rightarrow_{\#}}$. If $Q'$ contains freshness constraints it is said that $\Rightarrow_{\#}$ *fails* for $\mathcal{Q}$; otherwise, $\mathcal{Q}'$ will be called a *successful* triple for $\Rightarrow_{\#}$.

▶ Remark. Since by definition in a successful triple regarding $\Rightarrow_{\approx}$, $\mathcal{Q}$, one has only fixpoint equations and $\Rightarrow_{\#}$ acts only over freshness constraints, $Q'$ in the definition above contains only fixpoint equations and freshness constraints. Also, by a simple case analysis on $t$ one can check that any triple with freshness constraints $a\#_? t$ is reducible by $\Rightarrow_{\#}$, except when $t \equiv \bar{a}$. Hence the freshness constraints in $Q'$ would be only of the form $a\#_? \bar{a}$.

An interesting observation is that the relation $\Rightarrow_{\approx}$, starts from a triple with the substitution identity and always maintains a triple $\langle \nabla, \sigma', P'\rangle$ in which the substitution $\sigma'$ does not affect the current problem $P'$. The same happens for $\Rightarrow_{\#}$ since derivations with this relation do not change the substitution. This motivates the following definition and lemma.

▶ **Definition 19** (Valid triple). $\mathcal{P} = \langle \nabla, \sigma, P\rangle$ *is valid* if $im(\sigma) \cap dom(\sigma) = \emptyset$ and $dom(\sigma) \cap Var(P) = \emptyset$.

▶ Remark. A substitution $\sigma$ in a valid triple $\mathcal{P}$ is *idempotent*, that is, $\sigma\sigma = \sigma$.

▶ **Lemma 20** (Preservation of valid triples). *If* $\mathcal{P} = \langle \nabla, \sigma, P\rangle$ *is valid and* $\mathcal{P} \Rightarrow_{\approx} \cup \Rightarrow_{\#} \mathcal{P}' = \langle \nabla', \sigma', P'\rangle$*, then* $\mathcal{P}'$ *is also valid.*

**Proof.** By case analysis on the derivation rules used in the relation $\Rightarrow_{\approx} \cup \Rightarrow_{\#}$. The only rule that enlarges the domain of $\sigma$ is ($\approx_?$ **inst**) that builds a new substitution $\sigma' = \sigma\{X/\pi^{-1} \cdot t\}$, for $t$ such that $X \notin Var(t)$. Since $dom(\sigma) \cap Var(P) = \emptyset$ and $t$ occurs in $P$, $dom(\sigma') \cap Var(t) = \emptyset$; hence, we obtain the first property: $im(\sigma') \cap dom(\sigma') = \emptyset$. For the second property, notice that $P'$ consists of two parts: 1. $(P - \{\pi.X \approx_? t\})\{X/\pi^{-1} \cdot t\}$ that includes equations and freshness constraints whose variables do not intersect $dom(\sigma')$; and 2. $\bigcup_{\substack{Y \in dom(\sigma'),\\ a\#Y \in \nabla}} \{a\#_? Y\sigma'\}$ that also does not include variables in $dom(\sigma')$, since $im(\sigma') \cap dom(\sigma') = \emptyset$. Consequently, $dom(\sigma') \cap Var(P') = \emptyset$ and then $\mathcal{P}'$ is a valid triple. ◀

From now on, we consider only valid triples.

▶ **Lemma 21** (Termination of $\Rightarrow_{\approx}$). *There is no infinite chain of reductions* $\Rightarrow_{\approx}$ *starting from an arbitrary triple* $\mathcal{P} = \langle \nabla, \sigma, P\rangle$*.*

**Proof.** The proof is by well-founded induction on $\mathcal{P}$ using the measure

$$\|\mathcal{P}\| = \langle |Var(P_{\approx})|, \|P\|, |P_{\mathrm{nfp}_{\approx}}|\rangle$$

with a lexicographic ordering, where $\|P\| = \sum_{s \approx_? t \in P_{\approx}} |s| + |t| + \sum_{a\#_? u \in P_{\#}} |u|$.

Note that the measure on triples, $\|\_\|$, decreases after each step $\langle \nabla, \sigma, P\rangle \Rightarrow_{\approx} \langle \nabla, \sigma', P'\rangle$:

- for ($\approx_?$ **inst**), $|Var(P_{\approx})| > |Var(P'_{\approx})|$;
- for ($\approx_?$ **refl**), ($\approx_?$ **pair**), ($\approx_?$ **app**), ($\approx_?$ [**aa**]), ($\approx_?$ [**ab**]) and ($\approx_?$ **C**), $|Var(P_{\approx})| \geq |Var(P'_{\approx})|$, but $\|P\| > \|P'\|$;
- for ($\approx_?$ **inv**), both $|Var(P_{\approx})| = |Var(P' \approx)|$ and $\|P\| = \|P'\|$, but $|P_{\mathrm{nfp}_{\approx}}| > |P'_{\mathrm{nfp}_{\approx}}|$.

◀

▶ **Lemma 22** (Termination of $\Rightarrow_{\#}$). *There is no infinite chain of reductions* $\Rightarrow_{\#}$ *starting from an arbitrary triple* $\mathcal{P} = \langle \nabla, \sigma, P\rangle$*.*

**Proof.** The proof is by induction on $\mathcal{P}$ using as measure $\|P_{\#}\|$. It can be checked easily that this measure decreases after each step: $\langle \nabla, \sigma, P\rangle \Rightarrow_{\#} \langle \nabla, \sigma', P'\rangle$. ◀

To solve a unification problem, $\langle \nabla, P\rangle$, one starts building the derivation tree for $\Rightarrow_{\approx}$, labelling the root node with the triple $\langle \nabla, id, P\rangle$. This tree has leaves labelled with $\Rightarrow_{\approx}$-nf's that are either failing or successful triples. Then, the tree is extended by building $\Rightarrow_{\#}$-derivations starting from all successful leaves. The extended tree will include failing leaves and successful leaves. The successful leaves will be labelled by triples $\mathcal{P}'$ in which the problem $P'$ consists only of fixpoint equations. Since $\Rightarrow_{\approx}$ and $\Rightarrow_{\#}$ are both terminating (Lemmas 21 and 22), the process described above must be also terminating.

▶ **Definition 23** (Derivation tree for $\langle \nabla, P\rangle$). A *derivation tree* for the unification problem $\langle \nabla, P\rangle$, denoted as $\mathcal{T}_{\langle \nabla, P\rangle}$, is a tree with root label $\mathcal{P} = \langle \nabla, id, P\rangle$ built in two stages:

- Initially, a tree is built, whose branches end in leaf nodes labelled with the triples in $\mathcal{P}_{\Rightarrow_{\approx}}$. The labels in each path from the root to a leaf correspond to a $\Rightarrow_{\approx}$-derivation.

- Further, for each leaf labelled with a successful triple $\mathcal{Q}$ in $\mathcal{P}_{\Rightarrow_\approx}$, the tree is extended with a path to a new leaf that is labelled with a $\bar{\mathcal{Q}} \in \mathcal{Q}_{\Rightarrow_\#}$. The labels in the extended path from the node with label $\mathcal{Q}$ to the new leaf correspond to a $\Rightarrow_\#$-derivation.

▶ Remark. For any $\langle \nabla, P\rangle$, all labels in the nodes of $\mathcal{T}_{\langle \nabla, P\rangle}$ are valid by Lemma 20.

▶ **Lemma 24** (Characterisation of leaves of $\mathcal{T}_{\langle \nabla, P\rangle}$)**.** *Let $\langle \nabla, P\rangle$ be a unification problem. If $\mathcal{P}' = \langle \nabla', \sigma', P'\rangle$ is the label of a leaf in $\mathcal{T}_{\langle \nabla, P\rangle}$, then $P'$ can be partitioned as follows: $P' = P'' \cup P_\perp$, where $P''$ is the set of all fixpoint equations in $P'$ and $P_\perp = P' - P''$. If $P_\perp \neq \emptyset$ then $\mathcal{U}_C(\mathcal{P}') = \emptyset$.*

**Proof.** By case analysis (see Appendix A). ◀

The next definition is motivated by the previous characterisation of the labels of leaves in derivation trees.

▶ **Definition 25** (Successful leaves)**.** Let $\langle \nabla, P\rangle$ be a unification problem. A leaf in $\mathcal{T}_{\langle \nabla, P\rangle}$ that is labelled with a triple of the form $\mathcal{Q} = \langle \nabla', \sigma', Q\rangle$, where $Q$ consists only of fixpoint equations, is called a *successful leaf* of $\mathcal{T}_{\langle \nabla, P\rangle}$. In this case $\mathcal{Q}$ is called a *successful triple* of $\mathcal{T}_{\langle \nabla, P\rangle}$. The sets of successful leaves and triples of $\mathcal{T}_{\langle \nabla, P\rangle}$ are denoted respectively by $SL(\mathcal{T}_{\langle \nabla, P\rangle})$ and $ST(\mathcal{T}_{\langle \nabla, P\rangle})$.

▶ **Lemma 26** (Preservation of solutions by $\Rightarrow_\approx$)**.** *If $\mathcal{P}$ is a valid triple and $\mathcal{P} \Rightarrow_\approx \mathcal{P}'$ then $\mathcal{U}_C(\mathcal{P}') \subseteq \mathcal{U}_C(\mathcal{P})$.*

The proof is by case analysis on one step $\Rightarrow_\approx$-reduction (see Appendix A). An interesting observation is that except for the cases of the rules ($\approx_?$ **C**) and ($\approx_?$ **inst**), it can be proved that $\mathcal{U}_C(\mathcal{P}) = \mathcal{U}_C(\mathcal{P}')$.

▶ **Lemma 27** (Preservation of solutions by $\Rightarrow_\#$)**.** *If $\mathcal{P} \Rightarrow_\# \mathcal{P}'$ then $\mathcal{U}_C(\mathcal{P}) = \mathcal{U}_C(\mathcal{P}')$.*

The proof is by case analysis on one step $\Rightarrow_\#$-reduction (see Appendix A).

The following lemma states that only successful leaves of $\mathcal{T}_{\langle \nabla, P\rangle}$ will produce solutions, which will be analysed in more detail in the next section.

▶ **Lemma 28** (Soundness of $\mathcal{T}_{\langle \nabla, P\rangle}$)**.** *$\mathcal{T}_{\langle \nabla, P\rangle}$ is correct, that is, if $\mathcal{P}' = \langle \nabla', \sigma, P'\rangle$ is the label of a leaf in $\mathcal{T}_{\langle \nabla, P\rangle}$, then*

1. *$\mathcal{U}_C(\mathcal{P}') \subseteq \mathcal{U}_C(\langle \nabla, id, P\rangle)$, and*
2. *if $P'$ contains non fixpoint equations or freshness constraints then $\mathcal{U}_C(\mathcal{P}') = \emptyset$.*

**Proof.** The first is proved by induction on the number of steps of $\Rightarrow_\approx$ and $\Rightarrow_\#$, using Lemmas 26 and 27. The second is a direct application of Lemma 24. ◀

The following lemma complements the soundness lemma (Lemma 28) stating that the set of successful triples in the derivation tree provides a complete set of solutions.

▶ **Lemma 29** (Completeness of $\mathcal{T}_{\langle \nabla, P\rangle}$)**.** *Let $\mathcal{T}_{\langle \nabla, P\rangle}$ be a derivation tree for the unification problem $\langle \nabla, P\rangle$, where $\mathcal{P} = \langle \nabla, id, P\rangle$. Then $\mathcal{U}_C(\mathcal{P}) = \bigcup_{\mathcal{Q} \in ST(\mathcal{T}_{\langle \nabla, P\rangle})} \mathcal{U}_C(\mathcal{Q})$*

The interesting inclusion is proved by induction on the size of subtrees of $\mathcal{T}_{\langle \nabla, P\rangle}$. This is done verifying that in the preservation lemmas for $\Rightarrow_\approx$ and $\Rightarrow_\#$ (Lemmas 26, 27) all rules, except ($\approx_?$ **C**) and ($\approx_?$ **inst**) preserve exactly the same sets of solutions (see Appendix A).

▶ **Corollary 30** (Generality of successful triples). *Let $\mathcal{P} = \langle \nabla, P\rangle$ be a unification problem and $\langle \Delta', \sigma'\rangle \in \mathcal{U}_C(\mathcal{P})$. Then there exists a successful triple $\mathcal{Q} \in ST(\mathcal{T}_{\langle \nabla, P\rangle})$ where $\mathcal{Q} = \langle \Delta, \sigma, Q\rangle$ such that $\langle \Delta', \sigma'\rangle \in \mathcal{U}_C(\mathcal{Q})$, and hence, $\Delta' \vdash \Delta\sigma'$ and there exists $\lambda$ such that $\Delta' \vdash \sigma\lambda \approx \sigma'$.*

**Proof.** By Lemma 29, $\mathcal{U}_C(\mathcal{P}) = \bigcup_{\mathcal{P}' \in ST(\mathcal{T}_{\langle \nabla, P\rangle})} \mathcal{U}_C(\mathcal{P}')$. Then there exists $\mathcal{Q} \in ST(\mathcal{T}_{\langle \nabla, P\rangle})$ such that $\langle \Delta', \sigma'\rangle \in \mathcal{U}_C(\mathcal{Q})$. Suppose $\mathcal{Q} = \langle \Delta, \sigma, Q\rangle$. Then by the first and fourth conditions of the definition of solution (Def. 14) we have that $\Delta' \vdash \Delta\sigma'$ and there exists $\lambda$ such that $\Delta' \vdash \sigma\lambda \approx \sigma'$. ◀

## 4 Generation of solutions for successful leaves of $\mathcal{T}_{\langle \nabla, P\rangle}$

Let $\mathcal{P} = \langle \nabla, \sigma, P\rangle$ be a successful leaf for a given unification problem, say $\langle \Delta, Q\rangle$. In order to build a solution for $\mathcal{P}$, we will select and combine solutions generated for fixpoint equations $\pi.X \approx_? X$, for each $X \in Var(P)$. The set of generated solutions for one fixpoint equation will be denoted as $\langle \nabla, \pi.X \approx_? X\rangle_{Sol_G}$ and consists of solutions of the form $\langle \nabla', \{X/t\}\rangle$.

If there are several fixpoint equations on the same variable $X$ in $P$, say $\pi_i.X \approx_? X$, where $\pi_i \in \Pi_X$, the set of permutations in fixpoint equations for $X$ in $P$, and where $|\Pi_X| = k$, a solution that satisfies all these fixpoint equations should map the variable $X$ to the *same* term and have a freshness context that contains and preserves all freshness constraints for the solutions of all fixpoint equations. The set of generated solutions of all fixpoint equations on a variable $X$ in $\mathcal{P}$ will be denoted as $[X]_{\mathcal{P}_G}$. If these solutions are expressed in terms of independent variables and since they are independent regarding freshness constraints in $\nabla$, a solution of the successful leaf $\langle \nabla, \sigma, P\rangle$, can be built by the union of the constraints in solutions in $[X]_{\mathcal{P}_G}$ for each $X$ in $\mathcal{P}$ and by the union of their substitutions. Generated solutions for a successful leaf will be denoted by $[\mathcal{P}]_{Sol_G}$. Before proceeding to the definitions of $\langle \nabla, \pi.X \approx_? X\rangle_{Sol_G}$ and of $[X]_{\mathcal{P}_G}$, in this section we will study properties of pseudo-cycles of permutations, in order to provide precise conditions to build terms $t$ resulting from combinations of the atoms in $dom(\pi)$, such that $\pi \cdot t \approx_{\{\alpha, C\}} t$.

For convenience, in this section we will use the algebraic cycle representation of permutations. Thus, instead of sequences of swappings, permutations in nominal terms will be read as products of disjoint cycles [6]. For instance, the nominal permutations given as $(a\,b) :: (a\,c) :: (a\,d) :: (e\,f)$ and $(a\,b) :: (a\,c) :: (b\,c) :: (e\,f)$ will be represented as the product of permutation cycles $(a\,b\,c\,d)\,(e\,f)$ and $(a\,c)\,(b)\,(e\,f)$. Permutation cycles of length one are omitted; thus the last product is written as $(a\,c)\,(e\,f)$ and the identity, that is, a product of all unitary cycles is written as the empty product. In general the cyclic representation of a permutation consists of the product of all its cycles.

Let $\pi$ be a permutation with domain of cardinality $n$. Given $0 < i \leq n$ and $a \in dom(\pi)$ the elements of the sequence $a, \pi(a), \pi^2(a), \ldots$ cannot be all distinct. Taking the first $k \leq n$, such that $\pi^k(a) = a$, we have the $k$-cycle $(a\ \pi(a)\ \ldots \pi^{k-1}(a))$, where $\pi^{j+1}(a)$ is the *successor* of $\pi^j(a)$. For the 4-cycle in the permutation $(a\,b\,c\,d)\,(e\,f)$, the 4-cycles generated by $a, b, c$ and $d$ are the same: $(a\,b\,c\,d) = (b\,c\,d\,a) = (c\,d\,a\,b) = (d\,a\,b\,c)$.

### 4.1 Pseudo-cycle set characterisation

The following definition establishes the notion of a *pseudo-cycle w.r.t. a $k$-cycle $\kappa$*. Intuitively, given a $k$-cycle $\kappa$ and a commutative function symbol $*$, a pseudo-cycle w.r.t $\kappa$, $(A_0 \ldots A_l)$, is a cycle whose elements are either atom terms built from the atoms in $\kappa$ or terms of the form $A'_i * A'^2_j$, for $A'_i, A'_j$ elements of a pseudo-cycle w.r.t $\kappa$.

[2] We will adopt infix notation for commutative operators, thus $*\langle B_i, B_j\rangle$ will be written $B_i * B_j$.

▶ **Definition 31** (Pseudo-cycle)**.** Let $\kappa = (a_0\ a_1\ \ldots\ a_{k-1})$ be a $k$-cycle of a permutation $\pi$. A *pseudo-cycle* w.r.t. $\kappa$ is inductively defined as follows:

1. $\overline{\kappa} = (\overline{a_0} \cdots \overline{a_{k-1}})$ is a *pseudo-cycle* w.r.t. $\kappa$, called *trivial pseudo-cycle* of $\kappa$.
2. $\kappa' = (A_0\ \ldots A_{k'-1})$ is a *pseudo-cycle* w.r.t. $\kappa$, if the following conditions are simultaneously satisfied:
   a. each element of $\kappa'$ is of the form $B_i * B_j$, where $*$ is a commutative function symbol in the signature, and $B_i, B_j$ are different elements of $\kappa''$, a *pseudo-cycle* w.r.t. $\kappa$. $\kappa'$ will be called a *first-instance pseudo-cycle* of $\kappa''$ w.r.t. $\kappa$.
   b. $A_i \not\approx_{\alpha,C} A_j$ for $i \neq j$, $0 \leq i, j \leq k' - 1$;
   c. for each $0 \leq i < k' - 1$, $\kappa \cdot A_i \approx_{\{\alpha,C\}} A_{i+1}$, where $i + 1$ abbreviates $i + 1 \bmod k'$.

The *length* of the pseudo-cycle $\kappa$, denoted by $|\kappa|$, consists of the number of elements in $\kappa$. A pseudo-cycle of length one will be called *unitary*.

▶ **Example 32.** Let $\kappa = (a\ b\ c\ d)$ be a 4-cycle and suppose $*, \oplus, +$ are commutative operators in the signature. The following are pseudo cycles w.r.t. $\kappa$: $\overline{\kappa} = (\overline{a}\ \overline{b}\ \overline{c}\ \overline{d})$; $\kappa_1 = ((\overline{a} * \overline{b})\ (\overline{b} * \overline{c})\ (\overline{c} * \overline{d})\ (\overline{d} * \overline{a}))$; $\kappa_2 = ((\overline{a} \oplus \overline{c})\ (\overline{b} \oplus \overline{d}))$; $\kappa_{11} = (((\overline{a} * \overline{b}) + (\overline{b} * \overline{c}))\ ((\overline{b} * \overline{c}) + (\overline{c} * \overline{d}))((\overline{c} * \overline{d}) + (\overline{d} * \overline{a}))\ ((\overline{d} * \overline{a}) + (\overline{a} * \overline{b})))$; $\kappa_{12} = (((\overline{a} * \overline{b}) * (\overline{c} * \overline{d}))\ ((\overline{b} * \overline{c}) * (\overline{d} * \overline{a})))$; $\kappa_{21} = (((\overline{a} \oplus \overline{c}) * (\overline{b} \oplus \overline{d})))$; $\kappa_{121} = (((\overline{a} * \overline{b}) * (\overline{c} * \overline{d})) * ((\overline{b} * \overline{c}) * (\overline{d} * \overline{a})))$. $\kappa_1$ and $\kappa_2$ are first-instance pseudo-cycles of $\overline{\kappa}$, and $\kappa_{11}$ and $\kappa_{12}$ of $\kappa_1$ and $\kappa_{21}$ of $\kappa_2$. Notice that, $|\overline{\kappa}| = |\kappa_1| = |\kappa_{11}| = 4$, $|\kappa_{12}| = 2$, and $|\kappa_{21}| = |\kappa_{121}| = 1$. According to the definition, $((\overline{a} * \overline{d})\ (\overline{b} * \overline{a})\ (\overline{c} * \overline{b})\ (\overline{d} * \overline{c}))$ is a first-instance pseudo-cycle of $\overline{\kappa}$, but it corresponds to $\kappa_1$.

Also, observe that for the elements of the unitary pseudo-cycles $\kappa_{21}$ and $\kappa_{121}$, say $s$ and $t$, $\{X/s\}$ and $\{X/t\}$ are solutions of the fixpoint equation $\kappa \cdot X \approx_? X$.

The notion of equivalence between pseudo-cycles can be extended to *equivalence modulo commutativity* of a binary operator $*$:

▶ **Definition 33** (Pseudo-cycle commutative equivalence)**.** Two pseudo-cycles $\kappa_1$ and $\kappa_2$ w.r.t. the pseudo-cycle $\kappa$ are said to be *equivalent modulo commutativity of* $*$, denoted by $\kappa_1 \approx_{(C,*)} \kappa_2$, if each element $A$ of $\kappa_1$ is equivalent modulo commutativity of $*$ to one element $B$ of $\kappa_2$, i.e., $A \approx_{(C,*)} B$ and they have the same successor, that is, $\kappa \cdot A \approx_{(C,*)} \kappa \cdot B$.

▶ **Example 34.** $((A * B)\ (C * D)\ (E * F)) \approx_C ((F * E)\ (A * B)\ (D * C))$

$[\kappa]_{(C,*)}$ denotes the equivalence class modulo commutativity of $*$ of the pseudo-cycle $\kappa$, that is, $[\kappa]_{(C,*)} = \{\kappa' \mid \kappa \approx_C \kappa'\}$. When the operator $*$ is clear from the context, we will denote it by $\approx_C$ and the congruence class of the pseudo-cycle $\kappa$ by $[\kappa]_C$. So, pseudo-cycle equivalence modulo $C$ can be determined by a single element:

▶ **Lemma 35.** *If $\kappa_1$ and $\kappa_2$ are two pseudo-cycles w.r.t. the cycle $\kappa$, and there exists $t \in \kappa_1$ such that $t \approx_C t'$, for some $t' \in \kappa_2$, then $\kappa_1 \approx_C \kappa_2$.*

**Proof.** The proof follows directly from the fact that $\kappa.t \approx_C \kappa.t'$. ◀

▶ Remark. The elements of pseudo-cycles can be represented by the coefficients of $\kappa$. Thus, if one chooses an element $A_0$ of $\kappa$, the pseudo-cycle $\kappa' = ((\kappa^0 \cdot A_0) \cdots (\kappa^{k-1} \cdot A_0))$ can be represented by $(\overline{0} \cdots \overline{k-1})$, the choice of $A_0$ will only reorder the pseudo-cycle. Then, one can define an operation $+$ between elements of $(\overline{0} \cdots \overline{k-1})$ where $\overline{i} + \overline{j} = \overline{i+j}$ and $\overline{i+k} = \overline{i}$.

With the aim to compute the first instance pseudo-cycles for a commutative symbol $*$, we define below the matrix associated with a pseudo-cycle $\kappa$ represented by its coefficients:

▶ **Definition 36.** The *first-instance pseudo-cycle matrix* $\mathcal{M}_{(k-1)\times k}$ *of a pseudo-cycle* $\kappa = (\overline{0}\cdots\overline{k-1})$ *for a commutative function symbol* $*$, is defined as the $(k-1)\times k$-matrix with components $a_{ij} := \overline{j-1} * \overline{i+j-1}$. When the function symbol $*$ is clear from the context $\mathcal{M}_{(k-1)\times k}$ will be called simply *first instance pseudo-cycle matrix of* $\kappa$.

▶ Remark. We want to establish a relationship between the rows of the matrix $\mathcal{M}_{(k-1)\times k}$ related to $\kappa$ and the first instance pseudo-cycles of $\kappa$: for each $i$, $1 \le i \le k-1$, we map the $i$-th row $[a_{i1}\ a_{i2}\ldots a_{ik}]$ of $\mathcal{M}_{(k-1)\times k}$ to the $k$-cycle: $\kappa_i = (\overline{0} * \overline{i} \quad \overline{1} * \overline{i+1} \quad \ldots \quad \overline{k-1} * \overline{i-1})$.

▶ **Example 37.** The first-instance pseudo-cycle matrix of $\kappa = (\overline{0}\,\overline{1}\,\overline{2}\,\overline{3})$ is $\mathcal{M}_{3\times 4} =$

$$\begin{bmatrix} \overline{0}*\overline{1} & \overline{1}*\overline{2} & \overline{2}*\overline{3} & \overline{3}*\overline{0} \\ \overline{0}*\overline{2} & \overline{1}*\overline{3} & \overline{2}*\overline{0} & \overline{3}*\overline{1} \\ \overline{0}*\overline{3} & \overline{1}*\overline{0} & \overline{2}*\overline{1} & \overline{3}*\overline{2} \end{bmatrix}$$

Not all the rows of $\mathcal{M}_{3\times 4}$ are pseudo-cycles of $\kappa$: the second row doesn't, since it contradicts condition 2.b) of the Definition 31; however, it contains two first instance pseudo-cycles of $\kappa$, both with length 2. Also note that the first and third rows are equivalent modulo $C$.

The next results will establish properties of the matrix $\mathcal{M}_{(k-1)\times k}$ and conditions that should be satisfied for the rows of $\mathcal{M}_{(k-1)\times k}$ to be pseudo-cycles of $\kappa$.

▶ **Lemma 38.** *Let* $\mathcal{M} = (a_{ij})_{k-1\times k}$ *be a first-instance pseudo-cycle matrix for a pseudo-cycle* $\kappa$*. The following properties are valid in* $\mathcal{M}$*:*

1. $a_{i(j+1)} = \kappa \cdot a_{ij}$*, for* $j < k$ .
2. $\kappa \cdot a_{ik} = a_{i1}$*;*
3. *The element* $a_{ij}$ *is equivalent modulo commutativity of* $*$ *to the element* $a_{(k-i)(i+j)}$*, i.e.,* $a_{ij} \approx_C a_{(k-i)(i+j)}$*, for* $1 \le i \le \lfloor \frac{k-1}{2} \rfloor$.
4. *Suppose* $k = 2n$ *for some positive integer* $n$.
   a. $a_{ni} \approx_C a_{n(n+i)}$*, for* $1 \le i \le k$.
   b. *If* $\kappa_{n_1} = (a_{n1}\ a_{n2}\ldots\ a_{nn})$ *and* $\kappa_{n_2} = (a_{n(n+1)}\ a_{n(n+2)}\ldots\ a_{nk})$ *then* $\kappa_{n_1} \approx_C \kappa_{n_2}$.

   *That is, when* $k$ *is even, the* $\frac{k}{2}$*-th row of the matrix , has two equivalent modulo* $C$ *pseudo-cycles with relation to* $\kappa$*, both with length* $\frac{k}{2}$.

**Proof.** The proof is technical and can be found in Appendix B. ◀

The next lemma says that the first-instance pseudo-cycle matrix of $\kappa$ contains all possible first-instance pseudo-cycles of $\kappa$.

▶ **Theorem 39.** *Let* $\kappa$ *be a pseudo-cycle with* $k$ *elements and* $\mathcal{M}$ *be its first instance pseudo-cycle matrix. The following properties hold*

1. *if* $k$ *is even, then* $\kappa$ *has exactly* $\lfloor \frac{k-1}{2} \rfloor$ *first-instance pseudo-cycles with* $k$ *elements, and one with* $\frac{k}{2}$ *elements.*
2. *if* $k$ *is odd, then* $\kappa$ *has exactly* $\frac{k-1}{2}$ *first-instance pseudo-cycles with* $k$ *elements.*

**Proof.** The proof follows immediately from Lemma 38. The proof is in Appendix B. ◀

As corollary, we obtain the condition to be able to build unitary pseudo-cycles.

▶ **Corollary 40.** *A pseudo-cycle* $\kappa$ *contains unitary pseudo-cycles iff* $|\kappa|$ *is a power of two.*

Below, given $\mathcal{P} = \langle \emptyset, \pi \cdot X \approx_? X \rangle$ a fixpoint equational problem, we call a *combinatory solution* of $P$, a substitution $\{X/t\}$, such that $\pi \cdot t \approx_C t$, and $t$ contains only atoms from $\pi$ and commutative function symbols, built as unary pseudo-cycles w.r.t. $\kappa$ a cycle in $\pi$.

▶ **Theorem 41.** *Let $P = \langle \emptyset, \pi \cdot X \approx_? X \rangle$ be a fixpoint equational. $P$ has a combinatory solution iff there exists a unitary pseudo-cycle $\kappa$ w.r.t. $\pi$.*

**Proof.** ($\Leftarrow$) Suppose that $\pi$ has a unitary pseudo-cycle, say $\kappa = (\ t\ )$, then $\kappa \cdot t \approx_C t$, by definition of pseudo-cycles, and $\pi \cdot t \approx_C t$. Therefore, $\{X/t\}$ is a solution for $P$.
($\Rightarrow$) Suppose that $\pi$ does not have a unitary pseudo-cycle, then by Theorem 40, every pseudo-cycle $\kappa$ w.r.t. $\pi$ has length $k = 2^n(2r+1)$, for some positive integer $r$.

Suppose, by contradiction, that there is a combinatory solution for $P$, say $\{X/t\}$. If any atom from $dom(\kappa)$ is in $t$, then all atoms of $\kappa$ have to occur in $t$, otherwise we would not have $\pi \cdot t \approx_C t$. Since we only work modulo *commutativity*, terms may change their positions pairwise, inside $t$, respecting the parentheses. Therefore, we should be able to arrange the atoms in $\kappa$ pairs, and permute them around the commutative symbols. To do so, we have to take the $k$ elements and organise them in pairs, interactively. But if $k$ has an odd factor, different from 1, it will not be possible. Contradiction. ◀

▶ **Example 42.** Consider the fixpoint problem $\mathcal{P} = \langle \emptyset, \pi \cdot X \approx_? X \rangle$ where the 4-cycle $\kappa = (a\, b\, c\, d)$ belongs to $\pi$. By Example 32, $\kappa_{21}$ and $\kappa_{121}$ are unitary pseudo-cycles of $\kappa$.

1. $\kappa_{21} = ((\overline{a} \oplus \overline{c}) * (\overline{b} \oplus \overline{d}))$. Notice that $\{X/(\overline{a} \oplus \overline{c}) * (\overline{b} \oplus \overline{d})\}$ is a solution for $\mathcal{P}$. Also observe that $\pi \cdot (\overline{a} \oplus \overline{c}) = (\overline{b} \oplus \overline{d})$ and $\pi \cdot (\overline{b} \oplus \overline{d}) = (\overline{a} \oplus \overline{c})$.
2. $\kappa_{121} = (((\overline{a} * \overline{b}) * (\overline{c} * \overline{d})) * ((\overline{b} * \overline{c}) * (\overline{d} * \overline{a})))$. Similarly to the previous case $\{X/((\overline{a} * \overline{b}) * (\overline{c} * \overline{d})) * ((\overline{b} * \overline{c}) * (\overline{d} * \overline{a}))\}$ is a solution for $\mathcal{P}$. In addition, $\pi \cdot ((\overline{a} * \overline{b}) * (\overline{c} * \overline{d})) = ((\overline{b} * \overline{c}) * (\overline{d} * \overline{a}))$ and $\pi \cdot ((\overline{b} * \overline{c}) * (\overline{d} * \overline{a})) = ((\overline{a} * \overline{b}) * (\overline{c} * \overline{d}))$.

▶ Remark. Since one can generate infinitely many unitary pseudo-cycles from a given $2^n$-cycle $\kappa$ in $\pi$, $n \in \mathbb{N}$, there exist infinite solutions for the fixpoint problem $\langle \emptyset, \pi.X \approx_? X \rangle$.

## 4.2 Computation of solutions for fixpoint problems

Given a permutation $\pi$, the set of solutions of a fixpoint equational problem $\langle \nabla, \pi.X \approx_? X \rangle_{Sol_C}$ is built from their non trivial unitary pseudo-cycles according to the recursive definition of extended pseudo-cycles below. This notion will consider all possible and feasible terms built from combinations using the atom terms in the permutation cycles of interest (those of length a power of two) in a given permutation $\pi$ and the symbols in the signature.

▶ **Example 43.** For a permutation $\pi$ with permutation cycles $(g\ h\ i\ j\ k\ l\ m\ n)$, $(a\ b\ c\ d)$ and $(s\ t)$, $\oplus$ and $\star$ commutative operators and $\square$ and $\diamond$ function symbols in the signature. From the trivial pseudo-cycle $(\overline{a}\ \overline{b}\ \overline{c}\ \overline{d})$ we obtain the first-instance pseudo-cycle $(\overline{a} \oplus \overline{c} \quad \overline{b} \oplus \overline{d})$; from the last and the trivial pseudo-cycle $(\overline{s}\ \overline{t})$ we obtain the combination $(\langle \overline{a} \oplus \overline{c}, \overline{s} \rangle \quad \langle \overline{b} \oplus \overline{d}, \overline{t} \rangle)$, notice that the property that the elements of this "cycle" are image each of the other under $\pi$ holds, which means that combining in this manner preserves invariant 2.c in the definition of pseudo-cycles. To the elements of the last cycle we can apply any function symbol preserving this property, for instance, obtaining the "cycle" $(\diamond\langle \overline{a} \oplus \overline{c}, \overline{s} \rangle \quad \diamond\langle \overline{b} \oplus \overline{d}, \overline{t} \rangle)$. Finally, after applying the construction to build first-instance pseudo-cycles we obtain the "cycle" $(\diamond\langle \overline{a} \oplus \overline{c}, \overline{s} \rangle \star \diamond\langle \overline{b} \oplus \overline{d}, \overline{t} \rangle)$. Notice that the element in this cycle is a solution of the problem $\langle \emptyset, \pi.X \approx_? X \rangle$.

▶ **Definition 44** (Extended Pseudo-cycle)**.** Let $\pi.X \approx_? X$ be a fixpoint equation in a successful leaf of $\mathcal{T}_{\langle \Delta, Q \rangle}$. The *extended pseudo-cycles* $\kappa'$ of $\pi$ are inductively defined from its permutation cycles as follows:

1. $\kappa' = (Y)$, for any variable not occurring in the problem, is an extended pseudo-cycle;

2. $\kappa' = (\overline{a_0} \cdots \overline{a_{k'-1}})$ is an *extended pseudo-cycle* w.r.t. $(a_0 \ \cdots a_{k'-1})$ a permutation cycle in $\pi$ such that $k' = 2^l$, for $l > 0$, called *trivial pseudo-cycle* of $\pi$.
3. $\kappa' = (A_0 \ ... \ A_{k'-1})$ is an *extended pseudo-cycle* w.r.t. $\pi$, if the following conditions are simultaneously satisfied:
   a. i. each element of $\kappa'$ is of the form $B_i \star B_j$, where $\star$ is a commutative function symbol in the signature, and $B_i, B_j$ are different elements of $\kappa$, an *extended pseudo-cycle* w.r.t. $\pi$. $\kappa'$ will be called an *extended first-instance pseudo-cycle* of $\kappa$ w.r.t. $\pi$, or
      ii. each element of $\kappa'$ is of the form $B_i \star C_j$ for any commutative symbol $\star$, where $B_i$ and $C_j$ are elements of $\kappa$ and $\kappa''$ *extended pseudo-cycles* w.r.t. $\pi$, that might be the same, but not being $\kappa'$ an extended first-instance pseudo-cycle, or
      iii. each element of $\kappa'$ is of the form $\langle B_i, C_j \rangle$, where $B_i$ and $C_j$ are elements of $\kappa$ and $\kappa''$ *extended pseudo-cycles* w.r.t. $\pi$, that might be the same, or
      iv. either each element of $\kappa'$ is of the form $g\, B_i$ or each element is of the form $[e']\, B_i$, where $g$ is a non commutative function symbol in the signature and $e' \notin dom(\pi)$, and each $B_i$ is an element of an extended pseudo-cycle w.r.t $\pi$.
      v. each element of $\kappa'$ is of the form $[a_j]B_i$, where $a_j$ and $B_i$ are resp. atoms in $\kappa$ and elements in $\kappa'$, a permutation cycle of $\pi$ and an *extended pseudo-cycle* w.r.t. $\pi$.
   b. For $\nabla = \cup_{Y \in Var(\kappa')} dom(\pi) \# Y$,
      i. it does not hold that $\nabla \vdash A_i \approx_{\{\alpha, C\}} A_j$ for $i \neq j$, $0 \leq i, j \leq k' - 1$, where $k'$ is the length of $\kappa'$, and
      ii. for each $0 \leq i < k' - 1$ one has $\nabla \vdash \pi(A_i) \approx_{\{\alpha, C\}} A_{i+1}$, where $i + 1$ abbreviates $i + 1$ modulo $k'$.

Extended pseudo-cycles are used to generate all solutions of the fixpoint equations with the same variable in a successful leaf, say $\langle \nabla, \sigma, P \rangle$. More precisely, for each fixpoint equation for $X$ in $P$, unitary extended pseudo-cycles generate solutions (as in the previous section), which are then combined to produce a solution for $X$. A solution for $\langle \nabla, \sigma, P \rangle$ is then built as the composition of $\sigma$ with the solutions for all variables occurring in fixpoint equations in $P$. Details of this greedy generator of solutions are given in Appendix C.

## 5 Conclusions and future work

A Coq formalisation of a sound and complete nominal C-unification algorithm was obtained by combining $\Rightarrow_{\approx}$- and $\Rightarrow_{\#}$-reduction. The algorithm builds finite derivation trees such that the leaves provide a complete set of most general unifiers consisting of freshness contexts, substitutions and fixpoint equations. We have also introduced the notion of pseudo-cycle to eliminate fixpoint equations by generating their possibly infinite set of solutions. The generator is based on the analysis of permutation cycles, followed by a brute-force enumeration procedure. The generated solutions are correct; completeness and optimisations to avoid generating redundant solutions will be the subject of future work.

## References


1 T. Aoto and K. Kikuchi. *A Rule-Based Procedure for Equivariant Nominal Unification*. In *Pre-proc. of Higher-Order Rewriting (HOR)*, pages 1–5, 2016.

2 T. Aoto and K. Kikuchi. *Nominal Confluence Tool*. In *Proc. of 8th Int. Joint Conf.: Automated Reasoning (IJCAR)*, volume 9706 of *LNCS*, pages 173–182. Springer, 2016.

3 M. Ayala-Rincón, W. Carvalho-Segundo, M. Fernández, and D. Nantes-Sobrinho. *A Formalisation of Nominal Equivalence with Associative-Commutative Function Symbols*. In

*Pre-proc. of Logical and Semantic Frameworks with Applications (LSFA)*, to appear in *ENTCS*, pages 78–93, 2016.

4 M. Ayala-Rincón, M. Fernández, and D. Nantes-Sobrinho. *Nominal Narrowing*. In *Proc. of 1st Int. Conf. on Formal Structures for Computation and Deduction (FSCD)*, volume 52 of *LIPIcs*, pages 1–16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

5 M. Ayala-Rincón, M. Fernández, and A. C. Rocha-oliveira. *Completeness in PVS of a Nominal Unification Algorithm*. *ENTCS*, 323:57–74, 2016.

6 B. E. Sagan. *The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions*, volume 203 of *Graduate Texts in Math.* Springer, 2nd edition, 2001.

7 C. F. Calvès. *Complexity and implementation of nominal algorithms*. PhD Thesis, King's College London, 2010.

8 C. F. Calvès and M. Fernández. *Implementing Nominal Unification*. *ENTCS*, 176(1):25–37, 2007.

9 C. F. Calvès and M. Fernández. *The First-order Nominal Link*. In *Proc. of 20th Int. Symp. Logic-based Program Synthesis and Transformation (LOPSTR)*, volume 6564 of *LNCS*, pages 234–248. Springer, 2011.

10 J. Cheney. *αProlog User's Guide & Language Reference Version 0.3 DRAFT*, 2003.

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

12 R. A. Clouston and A. M. Pitts. *Nominal Equational Logic*. *ENTCS*, 172:223–257, 2007.

13 M. Fernández and M. J. Gabbay. *Nominal Rewriting*. *Information and Computation*, 205(6):917–965, 2007.

14 M. Fernández and M. J. Gabbay. *Closed nominal rewriting and efficiently computable nominal algebra equality*. In *Proc. of 5th Int. Work. on Logical Frameworks and Meta-languages: Theory and Practice (LFMTP)*, volume 34 of *EPTCS*, pages 37–51, 2010.

15 M. Fernández, M. J. Gabbay, and I. Mackie. *Nominal Rewriting Systems*. In *Proc. of 6th Int. Conf. on Principles and Practice of Declarative Programming (PPDP)*, pages 108–119. ACM Press, 2004.

16 M. J. Gabbay and A. Mathijssen. *Nominal (Universal) Algebra: Equational Logic with Names and Binding*. *Journal of Logic and Computation*, 19(6):1455–1508, 2009.

17 M. J. Gabbay and A. M. Pitts. *A new approach to abstract syntax with variable binding*. *Formal Aspects of Computing*, 13(3-5):341–363, 2002.

18 D. Kapur and P. Narendran. *Matching, Unification and Complexity*. *SIGSAM Bulletin*, 21(4):6–9, 1987.

19 R. Kumar and M. Norrish. *(Nominal) Unification by Recursive Descent with Triangular Substitutions*. In *Proc. of Interactive Theorem Proving (ITP)*, volume 6172 of *LNCS*, pages 51–66. Springer, 2010.

20 T. Kutsia, J. Levy, M. Schmidt-Schauß, and M. Villaret. *Nominal Unification of Higher Order Expressions with Recursive Let*. In *Proc. of 26th Int. Sym. on Logic-Based Program Synthesis and Transformation (LOPSTR)*, pages 1–15, 2016. To appear in *LNCS*.

21 J. Levy and M. Villaret. *An Efficient Nominal Unification Algorithm*. In *Rewriting Techniques and Applications (RTA)*, volume 6 of *LIPIcs*, pages 209–226. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2010.

22 A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. CUP, 2013.

23 J. H. Siekmann. *Unification of Commutative Terms*. In *Proc. of An Int. Symp. on Symbolic and Algebraic Manipulation*, volume 72 of *LNCS*, pages 22–29. Springer, 1979.

24 C. Urban. *Nominal Unification Revisited*. In *Proc. of Int. Work. on Unification (UNIF)*, volume 42 of *EPTCS*, pages 1–11, 2010.

25 C. Urban, A. M. Pitts, and M. J. Gabbay. *Nominal Unification*. *Theoretical Computer Science*, 323(1-3):473—497, 2004.

## A Proofs of the Unification Algorithm - Section 3

▶ **Lemma 24** (Characterisation of leaves of $\mathcal{T}_{\langle\nabla,P\rangle}$). *Let $\langle\nabla, P\rangle$ be a unification problem. If $\mathcal{P}' = \langle\nabla', \sigma', P'\rangle$ is the label of a leaf in $\mathcal{T}_{\langle\nabla,P\rangle}$, then $P'$ can be partitioned as follows: $P' = P'' \cup P_\perp$, where $P''$ is the set of all fixpoint equations in $P'$ and $P_\perp = P' - P''$. If $P_\perp \neq \emptyset$ then $\mathcal{U}_C(\mathcal{P}') = \emptyset$.*

**Proof.** If there is a non fixpoint equation in $P_\perp$, say $s \approx_? t$, then by definition of $\mathcal{T}_{\langle\nabla,P\rangle}$, $\mathcal{P}'$ should be a $\Rightarrow_\approx$-nf in $\mathcal{P}_{\Rightarrow_\approx}$. Since no rule of the relation $\Rightarrow_\approx$ applies to decompose the equation $s \approx_? t$ in $P'$, one has only the following possibilities:

1. Neither $s$ nor $t$ are suspended variables and $s$ and $t$ are nominal terms of different grammatical type (for instance an abstraction and a pair, or an atom term and a functional term).
2. $s$ and $t$ are functional terms rooted by different function symbols.
3. $s$ and $t$ are different atom terms.
4. $s$ is a suspension, say $\pi.X$, and $t \neq \pi'.X$, but $X \in Var(t)$.

Since for all these cases there is no $\langle\Delta, \delta\rangle$ such that $\Delta \vdash s\delta \approx_{\{\alpha,C\}} t\delta$, one can conclude that $\mathcal{U}_C(\langle\nabla', \sigma', \{s \approx_? t\}\rangle) = \emptyset$, which implies that $\mathcal{U}_C(\mathcal{P}') = \emptyset$.

In the case in which $P_\perp$ consists only of freshness constraints, there exists a success triple $\mathcal{Q} = \langle\nabla, \sigma, Q\rangle \in \mathcal{P}_{\Rightarrow_\approx}$ and $\mathcal{P}'$ is a $\Rightarrow_\#$-nf of $\mathcal{Q}$. The set $Q$ can be split into sets of freshness constraints, $Q_\perp$, and fixpoint equations $Q'' = P''$. The relation $\Rightarrow_\#$ will change only $Q_\perp$, and since freshness constraints in $P_\perp$ should be of the form $a\#_?\bar{a}$ (see remark after Def. 18), and there is no $\langle\Delta, \delta\rangle$ such that $\Delta \vdash a \# \bar{a}\delta$, that is $\Delta \vdash a \# \bar{a}$, one concludes that also in this case $\mathcal{U}_C(\mathcal{P}') = \emptyset$. ◀

▶ **Lemma 26** (Preservation of solutions by $\Rightarrow_\approx$). *If $\mathcal{P}$ is a valid triple and $\mathcal{P} \Rightarrow_\approx \mathcal{P}'$ then $\mathcal{U}_C(\mathcal{P}') \subseteq \mathcal{U}_C(\mathcal{P})$.*

**Proof.** The proof is by case analysis on one step $\Rightarrow_\approx$-reduction.
• Rules ($\approx_?$ **refl**), ($\approx_?$ **pair**), ($\approx_?$ **app**) and ($\approx_?$ [**aa**]): Let us analyse a one step derivation with the rule ($\approx_?$ [**aa**]):

$$\mathcal{P} = \langle\nabla, \sigma, P \uplus \{[a]s \approx_? [a]t\}\rangle \Rightarrow_\approx \langle\nabla, \sigma, P \cup \{s \approx_? t\}\rangle = \mathcal{P}'$$

Let $\langle\nabla', \sigma'\rangle$ be a solution in $\mathcal{U}_C(\mathcal{P}')$. Then, according to the definition of solution (Definition 14) four conditions are satisfied: first, for all $a\#X \in \nabla$, $\nabla' \vdash a \# X\sigma'$; second, for all $a\#_?w \in P$, $\nabla' \vdash a \# w\sigma'$; third, for all $u \approx_? v \in P \cup \{s \approx_? t\}$, $\nabla' \vdash u\sigma' \approx_{\{\alpha,C\}} v\sigma'$, and; fourth, there exists $\lambda$ such that $\nabla' \vdash \sigma\lambda \approx \sigma'$.

Except for the third condition, all other conditions hold trivially. The third condition also holds, since (by the inference rules of $\approx_{\{\alpha,C\}}$ and Lemma 8) $\nabla' \vdash s\sigma' \approx_{\{\alpha,C\}} t\sigma'$ if only if $\nabla' \vdash [a]s\sigma' \approx_{\{\alpha,C\}} [a]t\sigma'$; hence, $\mathcal{U}_C(\mathcal{P}') \subseteq \mathcal{U}_C(\mathcal{P})$. Notice that a solution $\langle\nabla', \sigma'\rangle$ in $\mathcal{U}_C(\mathcal{P})$, satisfies the four conditions for $\mathcal{P}'$, hence one has that $\mathcal{U}_C(\mathcal{P}') = \mathcal{U}_C(\mathcal{P})$ indeed.

A similar analysis shows that $\mathcal{U}_C(\mathcal{P}) = \mathcal{U}_C(\mathcal{P}')$ also for rules ($\approx_?$ **refl**), ($\approx_?$ **pair**), ($\approx_?$ **app**).
• Rule ($\approx_?$ **C**): Consider a derivation of the form below, where $i = 0$ or $i = 1$:

$$\mathcal{P} = \langle\nabla, \sigma, P \uplus \{f_k^C\langle s_0, s_1\rangle \approx_? f_k^C\langle t_0, t_1\rangle\}\rangle \Rightarrow_\approx \langle\nabla, \sigma, P \cup \{\langle s_0, s_1\rangle \approx_? \langle t_i, t_{(i+1)}\rangle\}\rangle = \mathcal{P}'$$

As for the previous rules, for $\langle\nabla', \sigma'\rangle \in \mathcal{U}_C(\mathcal{P}')$, the first, second and fourth conditions in Def. 14 are preserved trivially. Regarding the third condition, it also holds since $\nabla' \vdash$

$\langle s_0, s_1\rangle\sigma' \approx_{\{\alpha,C\}} \langle t_i, t_{i+1}\rangle\sigma'$ implies for the commutative function symbol $f_k^C$ that $\nabla' \vdash f_k^C\langle s_0, s_1\rangle\sigma' \approx_{\{\alpha,C\}} f_k^C\langle t_0, t_1\rangle\sigma'$. Thus, $\mathcal{U}_C(\mathcal{P}') \subseteq \mathcal{U}_C(\mathcal{P})$.

• Rule ($\approx_?$ [**ab**]):

$$\mathcal{P} = \langle \nabla, \sigma, P \uplus \{[a]s \approx_? [b]t\}\rangle \Rightarrow_{\approx} \langle \nabla, \sigma, P \cup \{s \approx_? (a\,b)\,t, a\#_? t\}\rangle = \mathcal{P}'$$

Let $\langle \nabla', \sigma'\rangle$ be a solution in $\mathcal{U}_C(\mathcal{P}')$. Again, the interesting condition to be checked is the third condition in Def. 14. Since $\langle \nabla', \sigma'\rangle$ is a solution of $\mathcal{P}'$, one has that $\nabla' \vdash a \# t\sigma'$ and $\nabla' \vdash s\sigma' \approx_{\{\alpha,C\}} ((a\,b)\cdot t)\sigma'$. By Lemma 7, one has $((a\,b)\cdot t)\sigma' = (a\,b)\cdot(t\sigma')$. Hence, by application of the $\alpha$-equivalence rule ($\approx_{\{\alpha,\mathbf{C}\}}$ [**ab**]) in Fig. 2, one concludes that $\nabla' \vdash [a](s\sigma') \approx_{\{\alpha,C\}} [b](t\sigma')$, which by the definition of substitution action (Def. 6) can be written as $\nabla' \vdash ([a]s)\sigma' \approx_{\{\alpha,C\}} ([b]t)\sigma'$. Thus, $\mathcal{U}_C(\mathcal{P}') \subseteq \mathcal{U}_C(\mathcal{P})$.

Also in this case notice that $\mathcal{U}_C(\mathcal{P}) \subseteq \mathcal{U}_C(\mathcal{P}')$; indeed, if $\nabla' \vdash ([a]s)\sigma' \approx_{\{\alpha,C\}} ([b]t)\sigma'$, by Def. 6, reverse application of the $\alpha$-equivalence rule ($\approx_\alpha$ [**ab**]) (Lemma 8) and Lemma 7, one has that $\nabla' \vdash s\sigma' \approx_{\{\alpha,C\}} ((a\,b)\,t)\sigma'$ and $\nabla' \vdash a \# t\sigma'$.

• Rule ($\approx_?$ **inst**). Consider the reduction

$$\mathcal{P} = \langle \nabla, \sigma, P \uplus \{\pi.X \approx_? t\}\rangle \Rightarrow_{\approx} \left\langle \nabla, \sigma'', P\{X/\pi^{-1}\cdot t\} \cup \bigcup_{\substack{Y \in dom(\sigma''),\\ a\#Y \in \nabla}} \{a\#_? Y\sigma''\} \right\rangle = \mathcal{P}'$$

where $\sigma'' := \sigma\{X/\pi^{-1}\cdot t\}$ and $X \notin Var(t)$.

Let $\langle \nabla', \sigma'\rangle \in \mathcal{U}_C(\mathcal{P}')$. First, we analyse the the third condition in Def. 14. Let $u \approx_? v$ be an equation in $P$. We have that $\nabla' \vdash u\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} v\{X/\pi^{-1}\cdot t\}\sigma'$ and $\nabla' \vdash \sigma' \approx \sigma\{X/\pi^{-1}\cdot t\}\lambda$, for some $\lambda$. Thus, $\nabla' \vdash u\{X/\pi^{-1}\cdot t\}(\sigma\{X/\pi^{-1}\cdot t\}\lambda) \approx_{\{\alpha,C\}} v\{X/\pi^{-1}\cdot t\}(\sigma\{X/\pi^{-1}\cdot t\}\lambda)$, which implies $\nabla' \vdash u\{X/\pi^{-1}\cdot t\}\lambda \approx_{\{\alpha,C\}} v\{X/\pi^{-1}\cdot t\}\lambda$. For the last part we use the general assumption that $\mathcal{P}$ and $\mathcal{P}'$ are valid triples; hence, by Lemma 20, $dom(\sigma)$ does not intersect the set $Var(P)\cup Var(t)\cup\{X\}$. Thus, $\nabla' \vdash u\sigma\{X/\pi^{-1}\cdot t\}\lambda \approx_{\{\alpha,C\}} v\sigma\{X/\pi^{-1}\cdot t\}\lambda$, and finally, by the hypothesis $\nabla' \vdash \sigma' \approx \sigma\{X/\pi^{-1}\cdot t\}\lambda$, one concludes that $\nabla' \vdash u\sigma' \approx_{\{\alpha,C\}} v\sigma'$.

To conclude the analysis of the third condition, $\pi.X \approx_? t$ should be considered. One has that $\pi.X(\sigma\{X/\pi^{-1}\cdot t\}\lambda) = \pi\cdot(\pi^{-1}\cdot t)\lambda$. The last term corresponds to $t\lambda$ that is equal to $t(\sigma\{X/\pi^{-1}\cdot t\}\lambda)$. So by reflexivity of $\approx_{\{\alpha,C\}}$ (Lem. 12) one concludes that $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$.

The first condition in Def. 14 is immediate. The second condition is more interesting and depends on the third one. We need to prove that for any $a\#_? u \in P$, $\nabla' \vdash a \# u\sigma'$. The proof proceeds by induction in $u$. The cases in which $u = \langle\rangle$, $u = \bar{b}$, $u = [a]v$ and $u = \phi.Y$, for $Y \neq X$, are immediate. The case in which $u = \bar{a}$ is not possible since it contradicts the hypothesis. The cases in which $u = fv$ or $u = \langle u_1, u_2\rangle$ and $u = [b]v$ follow by direct application of the induction hypothesis. The interesting case is when $u = \phi.X$ for which the hypothesis is that $\nabla' \vdash a \# \phi.X\{X/\pi^{-1}\cdot t\}\sigma'$. By application of the substitution we have $\nabla' \vdash a\#\phi\cdot(\pi^{-1}\cdot t)\sigma'$; by application of nominal properties for the freshness relation $\#$ this gives $\nabla' \vdash \pi\cdot\phi^{-1}\cdot a \# t\sigma'$. By freshness preservation (Lemma 9) and the fact that $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$ (proved in the analysis of the third condition of Def. 14) one obtains $\nabla' \vdash \pi\,\phi^{-1}\cdot a \# \pi.X\sigma'$; thus, by nominal properties one obtains $\nabla' \vdash \phi^{-1}\cdot a \# X\sigma'$ and finaly that $\nabla' \vdash a \# \phi.X\sigma'$.

The analysis of the fourth condition in Def. 14 uses the hypothesis that there exists a $\lambda$ such that $\nabla' \vdash \sigma''\lambda \approx \sigma'$ and, given that $\sigma'' := \sigma\{X/\pi^{-1}\cdot t\}$, for $\mathcal{P}$ one has that the substitution $\{X/\pi^{-1}\cdot t\}\lambda$ satisfies the requirement that $\nabla' \vdash \sigma\{X/\pi^{-1}\cdot t\}\lambda \approx \sigma'$.

• Finally, for the rule ($\approx_?$ **inv**) consider a derivation:

$$\mathcal{P} = \langle \nabla, \sigma, P \uplus \{\pi.X \approx_? \pi'.X\}\rangle \Rightarrow_{\approx} \langle \nabla, \sigma, P \cup \{\pi \oplus (\pi')^{-1}.X \approx_? X\}\rangle = \mathcal{P}'$$

Suppose that $\langle \nabla', \sigma' \rangle \in \mathcal{U}_C(\mathcal{P}')$. All conditions in Def. 14 hold trivially (in both directions) except the third. Suppose $\nabla' \vdash (\pi \oplus (\pi')^{-1}.X)\sigma' \approx_{\{\alpha,C\}} X\sigma'$, by substitution properties (Def. 6), this holds if and only if $\nabla' \vdash \pi \oplus (\pi')^{-1} \cdot (X\sigma') \approx_{\{\alpha,C\}} X\sigma'$, and by equivariance (Lemma 11) and the definition of action of substitutions (Def. 6), the last holds if and only if, $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} \pi'.X\sigma'$. One concludes that $\mathcal{U}_C(\mathcal{P}) = \mathcal{U}_C(\mathcal{P}')$. ◀

▶ **Lemma 27** (Preservation of solutions by $\Rightarrow_{\#}$)**.** *If $\mathcal{P} \Rightarrow_{\#} \mathcal{P}'$ then $\mathcal{U}_C(\mathcal{P}) = \mathcal{U}_C(\mathcal{P}')$.*

**Proof.** The proof is by case analysis on one step $\Rightarrow_{\#}$-reduction. The interesting case is rule ($\#_?$**var**). For rules ($\#_?\langle\rangle$), ($\#_?\mathbf{a}\overline{\mathbf{b}}$), ($\#_?$**app**), ($\#_?\mathbf{a}[\mathbf{a}]$) ($\#_?\mathbf{a}[\mathbf{b}]$) and ($\langle\#\rangle$**pair**) the analysis is simple and similar.

• Let us analyse, for example, the case of rule ($\#_?\mathbf{a}[\mathbf{b}]$):

$$\mathcal{P} = \langle \nabla, \sigma, P \uplus \{a\#_?[b]t\} \rangle \Rightarrow_{\#} \langle \nabla, \sigma, P \cup \{a\#_?t\} \rangle = \mathcal{P}'$$

Supposing $\langle \nabla', \sigma' \rangle \in \mathcal{U}_C(\mathcal{P})$, except for the second condition in Def. 14, all other three conditions hold trivially for $\mathcal{P}'$. The same happens when $\langle \nabla', \sigma' \rangle \in \mathcal{U}_C(\mathcal{P}')$: conditions first, third and fourth in Def. 14 hold for $\mathcal{P}$. For the second condition in Def. 14, by application of rule ($\#\,\mathbf{a}[\mathbf{b}]$) of the freshness relation (Fig. 1) and inversion property of these inference rule and, substitution action (Def. 6), one has that $\nabla' \vdash a \# t\sigma'$ if and only if $\nabla' \vdash a \# [b]t\sigma'$ if and only if $\nabla' \vdash a \# ([b]t\sigma')$. Hence, $\mathcal{U}_C(\mathcal{P}) = \mathcal{U}_C(\mathcal{P}')$.

• Now, consider the interesting case of ($\#_?$**var**):

$$\mathcal{P} = \langle \nabla, \sigma, P \uplus \{a\#_?\pi.X\} \rangle \Rightarrow_{\#} \langle \{(\pi^{-1} \cdot a)\#X\} \cup \nabla, \sigma, P \rangle = \mathcal{P}'$$

On the one hand, if $\langle \nabla', \sigma' \rangle \in \mathcal{U}_C(\mathcal{P})$, the second, third and fourth conditions in Def. 14 hold trivially for $\mathcal{P}'$. To prove the first condition for $\mathcal{P}'$, by the second condition in Def. 14 for $\mathcal{P}$ one has that $\nabla' \vdash a \# \pi.X\sigma'$, which, by nominal properties and substitution action (Def. 6), implies that $\nabla' \vdash \pi^{-1} \cdot a \# X\sigma'$. Since by hypothesis $\nabla' \vdash \nabla\sigma'$ (first condition of Def. 14 for $\mathcal{P}$), one has that $\nabla' \vdash (\{(\pi^{-1} \cdot a)\#X\} \cup \nabla)\sigma'$. Therefore, $\mathcal{U}_C(\mathcal{P}) \subseteq \mathcal{U}_C(\mathcal{P}')$.

On the other hand, if $\langle \nabla', \sigma' \rangle \in \mathcal{U}_C(\mathcal{P}')$, the first, third and fourth conditions in Def. 14 hold trivially for $\mathcal{P}$. For proving the second condition for $\mathcal{P}$, by the first condition in Def. 14 one has that $\nabla' \vdash (\pi^{-1} \cdot a) \# X\sigma'$, which again by nominal properties and Def. 6 implies that $\nabla' \vdash a \# (\pi.X)\sigma'$. Thus, by the last and the second condition in Def. 14 for $\mathcal{P}'$, one obtains the second condition for $\mathcal{P}$. Therefore, $\mathcal{U}_C(\mathcal{P}') \subseteq \mathcal{U}_C(\mathcal{P})$. ◀

▶ **Lemma 29** (Completeness of $\mathcal{T}_{\langle \nabla, P \rangle}$)**.** *Let $\mathcal{T}_{\langle \nabla, P \rangle}$ be a derivation tree for the unification problem $\langle \nabla, P \rangle$, where $\mathcal{P} = \langle \nabla, id, P \rangle$. Then*

$$\mathcal{U}_C(\mathcal{P}) = \bigcup_{\mathcal{Q} \in ST(\mathcal{T}_{\langle \nabla, P \rangle})} \mathcal{U}_C(\mathcal{Q})$$

**Proof.** From the soundness lemma (Lemma 28) we have that $\mathcal{U}_C(\mathcal{P}) \supseteq \bigcup_{\mathcal{Q} \in ST(\mathcal{T}_{\langle \nabla, P \rangle})} \mathcal{U}_C(\mathcal{Q})$. The other inclusion is proved by induction on the size of subtrees of $\mathcal{T}_{\langle \nabla, P \rangle}$. This is done verifying that in the preservation lemmas for $\Rightarrow_{\approx}$ and $\Rightarrow_{\#}$ (Lemmas 26 and 27) all rules, except ($\approx_?$ **C**) and ($\approx_?$ **inst**) preserve exactly the same sets of solutions.

• For one step application of rule ($\approx_?$ **C**) on a triple $\mathcal{Q}$, labelling a node in $\mathcal{T}_{\langle \nabla, P \rangle}$, one has two sibling nodes labelled with triples $\mathcal{Q}_1$ and $\mathcal{Q}_2$ such that

$$\mathcal{Q} = \langle \Delta, \sigma, Q \uplus \{f^C\langle s_0, s_1 \rangle \approx_? f^C\langle t_0, t_1 \rangle\} \rangle,$$
$$\mathcal{Q}_1 = \langle \Delta, \sigma, Q \cup \{\langle s_0, s_1 \rangle \approx_? \langle t_0, t_1 \rangle\} \rangle \quad \text{and} \quad \mathcal{Q}_2 = \langle \Delta, \sigma, Q \cup \{\langle s_0, s_1 \rangle \approx_? \langle t_1, t_0 \rangle\} \rangle.$$

If $\langle \Delta', \sigma' \rangle \in \mathcal{U}_C(\mathcal{Q})$ then it satisfies the four conditions in Def. 14 for $\mathcal{Q}$. It can be easily checked that $\langle \Delta', \sigma' \rangle$ satisfies the first, second and fourth conditions for both $\mathcal{Q}_1$ and $\mathcal{Q}_2$. Regarding the third condition, since it holds for $\mathcal{Q}$, it holds for any equation in $Q$ and also $\Delta' \vdash f^C\langle s_0, s_1\rangle\sigma' \approx_{\{\alpha,C\}} f^C\langle t_0, t_1\rangle\sigma'$. From the last, by substitution action one has that $\Delta' \vdash f^C\langle s_0\sigma', s_1\sigma'\rangle \approx_{\{\alpha,C\}} f^C\langle t_0\sigma', t_1\sigma'\rangle$. Then, by Lemma 8 either $\Delta' \vdash s_0\sigma' \approx_{\{\alpha,C\}} t_0\sigma'$ and $\Delta' \vdash s_1\sigma' \approx_{\{\alpha,C\}} t_1\sigma'$, or $\Delta' \vdash s_0\sigma' \approx_{\{\alpha,C\}} t_1\sigma'$ and $\Delta' \vdash s_1\sigma' \approx_{\{\alpha,C\}} t_0\sigma'$. Thus, the third condition holds for $\mathcal{Q}_1$ or for $\mathcal{Q}_2$, and it can be concluded that $\langle \Delta', \sigma' \rangle \in \mathcal{U}_C(\mathcal{Q})$ if and only if $\langle \Delta', \sigma' \rangle \in \mathcal{U}_C(\mathcal{Q}_1)$ or $\langle \Delta', \sigma' \rangle \in \mathcal{U}_C(\mathcal{Q}_2)$. Therefore $\mathcal{U}_C(\mathcal{Q}) = \mathcal{U}_C(\mathcal{Q}_1) \cup \mathcal{U}_C(\mathcal{Q}_2)$. By induction hypothesis, the solutions of the successful leaves in the subtrees rooted by $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are exactly the set $\mathcal{U}_C(\mathcal{Q})$.

• For one step application of rule ($\approx_?$ **inst**) on a triple $\mathcal{Q}$, labelling a node in $\mathcal{T}_{\langle \nabla, P\rangle}$, one has a sibling node labelled with a triple $\mathcal{Q}'$ such that

$$\mathcal{Q} = \langle \Delta, \sigma, Q \uplus \{\pi.X \approx_? t\}\rangle \Rightarrow_{\approx} \left\langle \Delta, \sigma'', Q\{X/\pi^{-1}\cdot t\} \cup \bigcup_{\substack{Y \in dom(\sigma''),\\ a\#Y\in\Delta}} \{a\#_? Y\sigma''\} \right\rangle = \mathcal{Q}'$$

where $\sigma'' := \sigma\{X/\pi^{-1}\cdot t\}$ and $X \notin Var(t)$.

Suppose that $\langle \Delta', \sigma' \rangle \in \mathcal{U}_C(\mathcal{Q})$. We will check that $\langle \Delta', \sigma' \rangle$ satisfies the four conditions in Def. 14 for $\mathcal{Q}'$.

– The **first condition**, that is $\Delta' \vdash \Delta\sigma'$, is the same for $\mathcal{Q}$ and $\mathcal{Q}'$.

– Proving that the **second condition** holds for $\mathcal{Q}'$, requires proving that:

**1.** $a\#_? u \in Q$ implies $\Delta' \vdash a\#(u\{X/\pi^{-1}\cdot t\})\sigma'$ and

**2.** for all $Y \in dom(\sigma'')$ such that $a\#Y \in \Delta$, $\Delta' \vdash a\#Y\sigma''\sigma'$.

For the first subcase we use induction in $u$ similarly to the case of rule ($\approx_?$ **inst**) in Lemma 26, being the interesting case, when $u = \phi.X$, for which the hypotheses of interest are that $\Delta' \vdash a\#\phi.X\sigma'$ and $\Delta' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$, by the second and third conditions in Def. 14 for $\mathcal{Q}$, respectively. From the first hypothesis, by nominal properties of the freshness relation one has that $\Delta' \vdash \phi^{-1}\cdot a\#X\sigma'$; by nominal properties of the freshness relation, also it follows that $\Delta' \vdash \pi\cdot\phi^{-1}\cdot a\#\pi\cdot X\sigma'$; by the second hypothesis and freshness preservation (Lemma 9) one obtains $\Delta' \vdash \pi\cdot\phi^{-1}\cdot a\#t\sigma'$; again by nominal properties the last gives $\Delta' \vdash a\#\phi\cdot\pi^{-1}\cdot t\sigma'$, which by substitution application gives finally $\Delta' \vdash a\#(\phi.X\{X/\pi^{-1}\cdot t\})\sigma'$.

For the second subcase above, observe initially that for a $Y \in dom(\sigma'')$ such that $a\#Y \in \Delta$, $\Delta' \vdash a\#Y\sigma''\sigma'$ equals $\Delta' \vdash a\#Y\sigma\{X/\pi^{-1}\cdot t\}\sigma'$. We will use induction on $Y\sigma$. The cases in which $Y\sigma = \langle\rangle$, $Y\sigma = \bar{b}$ and $Y\sigma = [a]v$ are immediate. The case in which $Y\sigma = \bar{a}$ is not possible since it contradicts the hypothesis, the first condition in Def. 14 for $\mathcal{Q}$: namely, it contradicts $\Delta' \vdash a\#Y\sigma'$ that is $\Delta' \vdash a\#\bar{a}$ since $\Delta' \vdash \sigma\lambda \approx \sigma'$ (fourth condition in Def. 14 for $\mathcal{Q}$). The cases in which $Y\sigma = fv$, $Y\sigma = \langle u_1, u_2\rangle$ and $Y\sigma = [b]v$ proceed by direct application of the induction hypothesis. The interesting cases happen when $Y\sigma = \phi.Z$. If $Z \neq X$, by the first condition for $\mathcal{Q}$ one has that $\Delta' \vdash a\#Y\sigma'$, that by the fourth condition gives $\Delta' \vdash a\#Y\sigma\lambda$; thus, $\Delta' \vdash a\#\phi.Z\lambda$ from which one has (since we are dealing with valid triples, Lemma 20) $\Delta' \vdash a\#\phi.Z\sigma\lambda$ and then $\Delta' \vdash a\#\phi.Z\{X/\pi^{-1}\cdot t\}\sigma\lambda$, and finally, $\Delta' \vdash a\#Y\sigma\{X/\pi^{-1}\cdot t\}\sigma\lambda$ that is $\Delta' \vdash a\#Y\sigma''\sigma'$. Otherwise, if $Z = X$, that is $Y\sigma = \phi.X$, by the first and fourth conditions for $\mathcal{Q}$ one has that $\Delta' \vdash a\#Y\sigma'$ and then that $\Delta' \vdash a\#Y\sigma\lambda$; thus, $\Delta' \vdash a\#\phi.X\lambda$. On the other side, by the third and fourth conditions for $\mathcal{Q}$, one has that $\Delta' \vdash \pi.X\sigma\lambda \approx_{\{\alpha,C\}} t\sigma\lambda$, which by equivariance (Lemma 11) and since $\mathcal{Q}$ is a valid triple gives $\Delta' \vdash X\lambda \approx_{\{\alpha,C\}} \pi^{-1}\cdot t\lambda$. Again, by equivariance and substitution and permutation properties one obtains $\Delta' \vdash \phi.X\lambda \approx_{\{\alpha,C\}} \phi\cdot\pi^{-1}\cdot t\lambda$. From the last and $\Delta' \vdash a\#\phi.X\lambda$, by freshness preservation (Lemma 9) one obtains $\Delta' \vdash a\#\phi\cdot\pi^{-1}\cdot t\lambda$ and, by

freshness properties this gives $\Delta' \vdash \pi \cdot \phi^{-1} \cdot a\#t\lambda$. Then, $\Delta' \vdash \pi \cdot \phi^{-1} \cdot a\#\pi \cdot X\{X/\pi^{-1} \cdot t\}\lambda$ which, since $\mathcal{Q}$ is a valid triple, gives $\Delta' \vdash \pi \cdot \phi^{-1} \cdot a\#\pi \cdot X\{X/\pi^{-1} \cdot t\}\sigma\lambda$ and by permutation and freshness properties $\Delta' \vdash a\#\phi.X\{X/\pi^{-1} \cdot t\}\sigma\lambda$. The last gives the desired property: $\Delta' \vdash a\#Y\sigma\{X/\pi^{-1} \cdot t\}\sigma\lambda$, that is $\Delta' \vdash a\#Y\sigma''\sigma'$.
– Now, we consider the **third condition** in Def. 14 for $\mathcal{Q}'$. It should be proved that for any equation $u \approx_? v$ in $Q$, except $\pi.X \approx_? t$, $\Delta' \vdash u\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} v\{X/\pi^{-1} \cdot t\}\sigma'$. This is done by induction in $u$ and $v$.

- Case $u = \langle\rangle$. Since $\Delta' \vdash \langle\rangle\sigma' \approx_{\{\alpha,C\}} v\sigma'$ (by the third condition for $\mathcal{Q}$), either $v = \langle\rangle$ or $v = \phi.Y$. The former subcase is trivial. For the latter subcase, it is necessary to consider whether $X \neq Y$ or $X = Y$. If $X \neq Y$, $\Delta' \vdash \langle\rangle\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1} \cdot t\}\sigma'$. If $X = Y$, we have the following chain using nominal properties and the hypothesis that $\Delta' \vdash \langle\rangle\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$: $\Delta' \vdash \pi \cdot \phi^{-1} \cdot \langle\rangle\sigma' \approx_{\{\alpha,C\}} \pi.X\sigma'$ implies $\Delta' \vdash \pi \cdot \phi^{-1} \cdot \langle\rangle\sigma' \approx_{\{\alpha,C\}} t\sigma'$, since $\Delta' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$ (by the third property for $\mathcal{Q}$), implies $\Delta' \vdash \phi^{-1} \cdot \langle\rangle\sigma' \approx_{\{\alpha,C\}} \pi^{-1} \cdot t\sigma'$ iff $\Delta' \vdash \phi^{-1} \cdot \langle\rangle\sigma' \approx_{\{\alpha,C\}} X\{X/\pi^{-1} \cdot t\}\sigma'$, implies $\Delta' \vdash \langle\rangle\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1} \cdot t\}\sigma'$ iff $\Delta' \vdash \langle\rangle\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1} \cdot t\}\sigma'$.
- Case $u = \bar{a}$ Since $\Delta' \vdash \bar{a}\sigma' \approx_{\{\alpha,C\}} v\sigma'$ (third condition for $\mathcal{Q}$), either $v = \bar{a}$ or $v = \phi.Y$. The former subcase is trivial. For the latter, either $X \neq Y$ or $X = Y$ as in the case of the unity. If $X \neq Y$, $\Delta' \vdash \bar{a}\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1} \cdot t\}\sigma'$ since $\Delta' \vdash \bar{a}\sigma' \approx_{\{\alpha,C\}} \phi.Y\sigma'$ holds. If $X = Y$, we have the following chain using nominal properties and the hypothesis that $\Delta' \vdash \bar{a}\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$: $\Delta' \vdash \pi \cdot \phi^{-1} \cdot \bar{a}\sigma' \approx_{\{\alpha,C\}} \pi.X\sigma'$ implies $\Delta' \vdash \pi \cdot \phi^{-1} \cdot \bar{a}\sigma' \approx_{\{\alpha,C\}} t\sigma'$, since $\Delta' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$ (by the third property for $\mathcal{Q}$), implies $\Delta' \vdash \phi^{-1} \cdot \bar{a}\sigma' \approx_{\{\alpha,C\}} \pi^{-1} \cdot t\sigma'$ iff $\Delta' \vdash \phi^{-1} \cdot \bar{a}\sigma' \approx_{\{\alpha,C\}} X\{X/\pi^{-1} \cdot t\}\sigma'$, implies $\Delta' \vdash \bar{a}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1} \cdot t\}\sigma'$ iff $\Delta' \vdash \bar{a}\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1} \cdot t\}\sigma'$.
- Case $u = [a]s$. Since $\Delta' \vdash [a]s\sigma' \approx_{\{\alpha,C\}} v\sigma'$, either $v$ is an abstraction or $v = \phi.Y$. The former subcase gives rise to two subcases:

  - $v = [a]w$ which holds by induction hypothesis, that is $\Delta' \vdash s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} w\{X/\pi^{-1} \cdot t\}\sigma'$ implies $\Delta' \vdash [a]s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} [a]w\{X/\pi^{-1} \cdot t\}\sigma'$ by rule ($\approx_\alpha$ [**aa**]),
  - and $v = [b]w$ which holds by induction hypothesis also; indeed, since $\Delta' \vdash [a]s\sigma' \approx_{\{\alpha,C\}} [b]w\sigma'$, by rule ($\approx_\alpha$ [**ab**]) it holds that $\Delta' \vdash s\sigma' \approx_{\{\alpha,C\}} (a\,b)w\sigma'$ and $\Delta' \vdash a\#w\sigma'$. Thus, one has that $\Delta' \vdash s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} (a\,b)w\{X/\pi^{-1} \cdot t\}\sigma'$ and, if also $\Delta' \vdash a\#w\{X/\pi^{-1} \cdot t\}\sigma'$ holds, one obtains, again by rule ($\approx_\alpha$ [**ab**]), that $\Delta' \vdash [a]s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} [b]w\{X/\pi^{-1} \cdot t\}\sigma'$. To prove that $\Delta' \vdash a\#w\{X/\pi^{-1} \cdot t\}\sigma'$ observe that if $X \notin Var(w)$ this holds since $\Delta' \vdash a\#w\sigma'$; otherwise, if $X \in Var(w)$, it would be proved by induction on $w$ being the elaborated case the case of a suspended variable: $\Delta' \vdash a\#\phi.X\{X/\pi^{-1} \cdot t\}\sigma'$ or equivalently that $\Delta' \vdash a\#\phi \cdot \pi^{-1} \cdot t\sigma'$. For proving this notice that since $\Delta' \vdash a\#w\sigma'$ and $X \in Var(w)$, $\Delta' \vdash a\#\phi.X\sigma'$; from the last and since $\Delta' \vdash \pi.X\sigma' \approx_? t\sigma'$ implies that $\Delta' \vdash \phi.X\sigma' \approx_? \phi \cdot \pi^{-1} \cdot t\sigma'$, one concludes that $\Delta \vdash a\#\phi \cdot \pi^{-1} \cdot t\sigma'$.

  For the subcase in which $v = \phi.Y$, notice that $Y\sigma' = [b]w$. If $X = Y$, the hypothesis that $\Delta' \vdash [a]s\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$ implies that $X \notin Var(s)$, and since also $\Delta' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$, we obtain $\Delta' \vdash [a]s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1} \cdot t\}\sigma'$. Otherwise, let $Y'$ be a new variable and extend $\mathcal{Q}$ with the equation $Y \approx_? [b]Y'$, $\Delta'$ with the constraints $c\#Y'$ for all $c\#Y \in \Delta'$ except $b\#Y'$, and $\sigma'$ with the bind $\{Y'/w\}$, so that $Y'\sigma' = w$. Thus, $\Delta' \vdash [a]s\sigma' \approx_{\{\alpha,C\}} [\phi \cdot b]\phi.Y'\sigma'$. If $\phi \cdot b = a$, by rule ($\approx_\alpha$ [**aa**]) we have $\Delta' \vdash s\sigma' \approx_{\{\alpha,C\}} \phi.Y'\sigma'$ and by i.h. $\Delta' \vdash s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi \cdot Y'\{X/\pi^{-1} \cdot t\}\sigma'$, which again by rule ($\approx_\alpha$ [**aa**])

gives $\Delta' \vdash [a]s\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1}\cdot t\}\sigma'$. If $\phi\cdot b = c$, by rule ($\approx_\alpha$ [**ab**]) we have $\Delta' \vdash s\sigma' \approx_{\{\alpha,C\}} (a\,c)\phi.Y'\sigma'$ and $\Delta' \vdash a\#\phi.Y'\sigma'$. Thus, $\Delta' \vdash a\#\phi.Y'\{X/\pi^{-1}\cdot t\}\sigma'$ and since by i.h. $\Delta' \vdash s\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} (a\,c)\phi.Y'\{X/\pi^{-1}\cdot t\}\sigma'$, by application of rule ($\approx_\alpha$ [**ab**]) we can conclude that $\Delta' \vdash [a]s\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} [\phi\cdot b]\phi.Y'\{X/\pi^{-1}\cdot t\}\sigma'$ or equivalently that $\Delta' \vdash [a]s\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1}\cdot t\}\sigma'$.

- Case $u = \langle s, w\rangle$, either $v = \langle v_1, v_2\rangle$ or $v = \phi.Y$. The former case holds by induction: by the third condition for $\mathcal{Q}$ one has $\Delta' \vdash \langle s, w\rangle\sigma' \approx_{\{\alpha,C\}} \langle v_1, v_2\rangle\sigma'$, and by rule ($\approx_\alpha$ **pair**), this holds iff $\Delta' \vdash s\sigma' \approx_{\{\alpha,C\}} v_1\sigma'$ and $\Delta' \vdash w\sigma' \approx_{\{\alpha,C\}} v_2\sigma'$; this gives, by i.h., $\Delta' \vdash s\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} v_1\{X/\pi^{-1}\cdot t\}\sigma'$ and $\Delta' \vdash w\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} v_2\{X/\pi^{-1}\cdot t\}\sigma'$ and finally, again by rule ($\approx_\alpha$ **pair**), $\Delta' \vdash \langle s, w\rangle\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \langle v_1, v_2\rangle\{X/\pi^{-1}\cdot t\}\sigma'$.
  For the case in which $v = \phi.X$, since $\Delta' \vdash \langle s, w\rangle\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$, $X \notin Var\langle s, w\rangle$; thus, since also $\Delta' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$, one has that $\Delta' \vdash \langle s, w\rangle\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1}\cdot t\}\sigma'$. For the case in which $v = \phi.Y$, for $X \neq Y$, notice that $Y\sigma' \approx_? \langle v_1, v_2\rangle$. Let $Y_1$ and $Y_2$ be new variables and extend $\mathcal{Q}$ with the equation $Y = \langle Y_1, Y_2\rangle$, $\Delta'$ with constraints $a\#Y_i$ for $i = 1, 2$ for all $a\#Y \in \Delta'$, and $\sigma'$ with the binds $\{Y_i/v_i\}$ for $i = 1, 2$, so that $Y_i\sigma' = v_i$, for $i = 1, 2$. Then by the hypothesis and ($\approx_\alpha$ **pair**) we have, $\Delta' \vdash s\sigma' \approx_{\{\alpha,C\}} \phi.Y_1\sigma'$ and $\Delta \vdash w\sigma' \approx_{\{\alpha,C\}} \phi.Y_2\sigma'$ and by i.h. $\Delta' \vdash s\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y_1\{X/\pi^{-1}\cdot t\}\sigma'$ and $\Delta \vdash w\sigma'\{X/\pi^{-1}\cdot t\} \approx_{\{\alpha,C\}} \phi.Y_2\{X/\pi^{-1}\cdot t\}\sigma'$; thus, again by rule ($\approx_\alpha$ **pair**), one has $\Delta' \vdash \langle s, w\rangle\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi\cdot\langle v_1, v_2\rangle\{X/\pi^{-1}\cdot t\}\sigma'$, that is $\Delta' \vdash \langle s, w\rangle\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1}\cdot t\}\sigma'$.
- Case $u = fs$, with $f$ non commutative or $s$ a non pair term. In this case $v$ should be either of the form $fw$ or $\phi.Y$. The former case holds by induction: $\Delta' fs\sigma' \approx_{\{\alpha,C\}} fw\sigma'$ iff $\Delta' \vdash s\sigma \approx_{\{\alpha,C\}} w\sigma'$, by rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **app**); then, by i.h. $\Delta' \vdash s\{X/\pi^{-1}\cdot t\}\sigma \approx_{\{\alpha,C\}} w\{X/\pi^{-1}\cdot t\}\sigma'$, and, again by rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **app**) one has that $\Delta' \vdash fs\{X/\pi^{-1}\cdot t\}\sigma \approx_{\{\alpha,C\}} fw\{X/\pi^{-1}\cdot t\}\sigma'$.
  For the case in which $v = \phi.X$, notice that $X \notin Var(s)$ since $\Delta \vdash fs\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$; thus, $\Delta \vdash fs\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1}\cdot t\}\sigma'$. For the case in which $v = \phi.Y$, with $X \neq Y$, notice that $Y\sigma' = fw$. Let $Y'$ be a new variable and extend $\mathcal{Q}$ with the equation $Y \approx_? fY'$, $\Delta'$ including the constraint $a\#Y'$ for all $a\#Y \in \Delta'$ and $\sigma'$ with the bind $\{Y'/w\}$; so $Y'\sigma' = w$. First, $\Delta' \vdash s\sigma' \approx_{\{\alpha,C\}} \phi.Y'\sigma'$ by the hypothesis and application of rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **app**); then, by i.h. one has that $\Delta' \vdash s\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y'\{X/\pi^{-1}\cdot t\}\sigma'$ and by application of the rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **app**) one has that $\Delta' \vdash fs\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} f\phi.Y'\{X/\pi^{-1}\cdot t\}\sigma'$ or equivalently $\Delta' \vdash fs\{X/\pi^{-1}\cdot t\}\sigma \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1}\cdot t\}\sigma'$.
- Case $u = f\langle u_0, u_1\rangle$ with $f$ commutative. In this case $v$ should be of the form $f\langle v_0, v_1\rangle$ or $\phi.Y$. The former case holds by induction, since by i.h., either $\Delta' \vdash u_0\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} v_0\{X/\pi^{-1}\cdot t\}\sigma'$ and $\Delta' \vdash u_1\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} v_1\{X/\pi^{-1}\cdot t\}\sigma'$, or $\Delta' \vdash u_0\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} v_1\{X/\pi^{-1}\cdot t\}\sigma'$ and $\Delta' \vdash u_1\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} v_0\{X/\pi^{-1}\cdot t\}\sigma'$ and, by application of rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **pair**), one obtains $\Delta' \vdash f\langle u_0, u_1\rangle\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} f\langle v_0, v_1\rangle\{X/\pi^{-1}\cdot t\}\sigma'$.
  For the case in which $v = \phi.X$, one has that $X \notin Var(u)$ since $\Delta' \vdash f\langle u_0, u_1\rangle\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$. Thus, since $\Delta' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$, we obtain that $\Delta' \vdash f\langle u_0, u_1\rangle\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \pi.X\{X/\pi^{-1}\cdot t\}\sigma'$. For the case in which $v = \phi.Y$ and $X \neq Y$, let $Y_0$ and $Y_1$ be new variables and extend $\Delta'$ with $a\#Y_i$ for $i = 0$ and $i = 1$ and all $a\#Y \in \Delta'$, and $\sigma'$ with the binds $\{Y_0/u_i, Y_2/u_{i+1}\}$ for $i = 0$ or $i = 1$. Then, one has that $\Delta' \vdash f\langle u_0, u_1\rangle\sigma' \approx_{\{\alpha,C\}} f\langle Y_0, Y_1\rangle\sigma'$ which by rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **pair**) and i.h. implies that either $\Delta' \vdash u_0\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} Y_0\{X/\pi^{-1}\cdot t\}\sigma'$ and $\Delta' \vdash u_1\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} Y_1\{X/\pi^{-1}\cdot t\}\sigma'$ or, $\Delta' \vdash u_0\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} Y_1\{X/\pi^{-1}\cdot t\}\sigma'$ and $\Delta' \vdash u_1\{X/\pi^{-1}\cdot$

$t\}\sigma' \approx_{\{\alpha,C\}} Y_0\{X/\pi^{-1}\cdot t\}\sigma'$. From the last and by application of rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **pair**) we conclude $\Delta' \vdash f\langle u_0,u_1\rangle\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} f\langle Y_0,Y_1\rangle\{X/\pi^{-1}\cdot t\}\sigma'$.

- Case $u = \phi.Y$. All cases, except the case of $v$ a suspended variable are treated as previously interchanging $u$ and $v$. When $v = \phi'.Y'$, we have to consider whether both $Y$ and $Y'$ are equal to $X$, both different from $X$ or only one of them is equal to $X$. Case $Y = Y' = X$, we have that $\Delta' \vdash \phi.X\sigma \approx_{\{\alpha,C\}} \phi'.X\sigma'$, since $\Delta' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$, we conclude that $\Delta' \vdash \phi.X\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi'.X\{X/\pi^{-1}\cdot t\}\sigma'$. Case $Y \neq X = Y'$, we also use the hypotheses that $\Delta' \vdash \phi.X\sigma \approx_{\{\alpha,C\}} \phi'.X\sigma'$ and $\Delta' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$ to conclude that $\Delta' \vdash \phi.Y\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi'.X\{X/\pi^{-1}\cdot t\}\sigma'$; the case $Y = X \neq Y'$ follows analogously. The case $Y \neq X \neq Y'$ requires only the hypothesis that $\Delta' \vdash \phi.X\sigma \approx_{\{\alpha,C\}} \phi'.X\sigma'$.

– To prove the **fourth condition** in Def. 14 for $\mathcal{Q}'$, select $\lambda' = \lambda \setminus \{X/X\lambda\}$. We have to prove that for any variable $Y$, $\Delta' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} Y\sigma'$ assuming that $\Delta' \vdash Y\sigma\lambda \approx_{\{\alpha,C\}} Y\sigma'$.

- First, we consider the case in which $Y \in Var(Q)\cup\{X\}\cup Var(t)$. Since $\mathcal{Q}$ is a valid triple $Y \notin dom(\sigma)$. If $Y \neq X$ then $\Delta' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} Y\lambda'$ and, since $\Delta' \vdash Y\lambda' \approx_{\{\alpha,C\}} Y\lambda$, applying the hypothesis we obtain $\Delta' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} Y\sigma'$. If $Y = X$ then $\Delta' \vdash X\sigma''\lambda' \approx_{\{\alpha,C\}} \pi^{-1}\cdot t\lambda'$ and since $X \notin Var(t)$, we also have that $\Delta' \vdash t\lambda' \approx_{\{\alpha,C\}} t\lambda$, which gives $\Delta' \vdash X\sigma''\lambda' \approx_{\{\alpha,C\}} \pi^{-1}\cdot t\sigma\lambda$, and by the hypothesis ($\Delta' \vdash \sigma\lambda \approx \sigma'$) and the third condition for $\mathcal{Q}$, more specifically by the hypothesis $\Delta' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$, one concludes that $\Delta' \vdash X\sigma''\lambda' \approx_{\{\alpha,C\}} X\sigma'$.
- Second, when $Y \in dom(\sigma)$ we prove that $\Delta' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} Y\sigma'$ by induction in the nominal term $Y\sigma$. All cases except when $Y\sigma = \phi.Z$ follow by simple application of the induction hypothesis. For instance, consider $Y\sigma = \langle u_1,u_2\rangle$. Let $Y_i$ for $i = 1,2$ be new variables, extend $\sigma'$ with bindings $\{Y_i/u_i\}$, for $i = 1,2$. Since $\Delta' \vdash \langle u_1,u_2\rangle\lambda \approx_{\{\alpha,C\}} Y\sigma'$, $Y\sigma'$ should be of the form $\langle v_1,v_2\rangle$. Then, extend $\sigma'$ with binds $\{Y_i/v_i\}$ for $i = 1,2$. Thus, we have that $\Delta' \vdash Y_i\sigma\lambda \approx_{\{\alpha,C\}} Y_i\sigma'$, for $i = 1,2$ and by i.h. that $\Delta' \vdash Y_i\sigma''\lambda' \approx_{\{\alpha,C\}} Y_i\sigma'$; so by rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **pair**) we can conclude that $\Delta' \vdash \langle Y_1,Y_2\rangle\sigma''\lambda' \approx_{\{\alpha,C\}} \langle Y_1,Y_2\rangle\sigma'$ that implies $\Delta' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} Y\sigma'$. For the interesting case in which $Y\sigma = \phi.Z$, we have $\Delta' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} \phi.Z\{X/\pi^{-1}\cdot t\}\lambda'$. If $Z \neq X$, then $\Delta' \vdash \phi.Z\{X/\pi^{-1}\cdot t\}\lambda' \approx_{\{\alpha,C\}} \phi.Z\lambda'$ and $\Delta' \vdash \phi.Z\lambda' \approx_{\{\alpha,C\}} \phi.Z\lambda$; thus, $\Delta' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} Y\sigma\lambda$ which by hypothesis implies $\Delta' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} Y\sigma'$. If $Z = X$, then $\Delta' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} \phi\cdot\pi^{-1}\cdot t\lambda'$; since $X \notin Var(t)$, $\Delta' \vdash \phi\cdot\pi^{-1}\cdot t\lambda' \approx_{\{\alpha,C\}} \phi\cdot\pi^{-1}\cdot t\lambda$; also, since $\mathcal{Q}$ is a valid triple $dom(\sigma)\cap Var(t) = \emptyset$, thus, $\Delta' \vdash \phi\cdot\pi^{-1}\cdot t\lambda \approx_{\{\alpha,C\}} \phi\cdot\pi^{-1}\cdot t\sigma\lambda$, and then $\Delta' \vdash \phi\cdot\pi^{-1}\cdot t\sigma\lambda \approx_{\{\alpha,C\}} \phi\cdot\pi^{-1}\cdot t\sigma'$. By the third condition for $\mathcal{Q}$, we have that $\Delta' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$, then $\Delta' \vdash \phi\cdot\pi^{-1}\cdot t\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$. Until this point we have proved that $\Delta' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} \phi.X\sigma'$. To conclude, since $\Delta' \vdash \phi.X\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma\lambda$, we have that $\Delta' \vdash \phi.X\sigma' \approx_{\{\alpha,C\}} Y\sigma\sigma\lambda$ and, since $\mathcal{Q}$ is a valid triple $im(\sigma)\cap dom(\sigma) = \emptyset$, thus, $\Delta' \vdash \phi.X\sigma' \approx_{\{\alpha,C\}} Y\sigma\lambda$, from which we conclude that $\Delta' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} Y\sigma'$.

◀

## B Proofs of Subsection 4.1

▶ Example B.1. For the 4-cycle $\kappa = (a\ b\ c\ d)$, if one considers its pseudo-cycle $\overline{\kappa} = (\overline{a}\ \overline{b}\ \overline{c}\ \overline{d})$, the representation $\kappa = (\overline{0}\ \ \overline{1}\ \ \overline{2}\ \ \overline{3})$ of $\kappa$ via coeffients, does not depend on the choice of $A_0$.

- if $A_0 = \overline{a}$, then $(\overline{0}\ \overline{1}\ \overline{2}\ \overline{3})$ corresponds to $((\kappa^0\cdot\overline{a})\ \ (\kappa^1\cdot\overline{a})\ \ (\kappa^2\cdot\overline{a})\ \ (\kappa^3\cdot\overline{a})) = (\overline{a}\ \overline{b}\ \overline{c}\ \overline{d})$.
- if $A_0 = \overline{b}$, then $(\overline{0}\ \overline{1}\ \overline{2}\ \overline{3})$ corresponds to $((\kappa^0\cdot\overline{b})\ \ (\kappa^1\cdot\overline{b})\ \ (\kappa^2\cdot\overline{b})\ \ (\kappa^3\cdot\overline{b})) = (\overline{b}\ \overline{c}\ \overline{d}\ \overline{a})$.

and so on.

If one chooses the pseudo-cycle $k_1 = ((a * c)\ (b * d))$ of $\kappa$, we can still represent it via coefficients $(\overline{0}\ \overline{1})$.

- if $A_0 = (a * c)$ then $(\overline{0}\ \overline{1})$ corresponds to the $\kappa_1$, itself.
- if $A_0 = (b * d)$ then $(\overline{0}\ \overline{1})$ corresponds to $((\kappa_1^0 \cdot (b * d)) \quad (\kappa_1^1 \cdot (b * d))) = ((b * d)\ (a * c)))$.

▶ **Lemma 38.** *Let $\mathcal{M} = (a_{ij})_{k-1\times k}$ be a first-instance pseudo-cycle matrix for a pseudo-cycle $\kappa$. The following properties are valid in $\mathcal{M}$:*

1. $a_{i(j+1)} = \kappa \cdot a_{ij}$, *for* $j < k$ .
2. $\kappa \cdot a_{ik} = a_{i1}$;
3. *The element* $a_{ij}$ *is equivalent modulo commutativity of* $*$ *to the element* $a_{(k-i)(i+j)}$, *i.e.,* $a_{ij} \approx_C a_{(k-i)(i+j)}$, *for* $1 \leq i \leq \lfloor \frac{k-1}{2} \rfloor$.
4. *Suppose* $k = 2n$ *for some positive integer* $n$.
   a. $a_{ni} \approx_C a_{n(n+i)}$, *for* $1 \leq i \leq k$.
   b. *If* $\kappa_{n_1} = (a_{n1}\ a_{n2} \ldots\ a_{nn})$ *and* $\kappa_{n_2} = (a_{n(n+1)}\ a_{n(n+2)} \ldots\ a_{nk})$ *then* $\kappa_{n_1} \approx_C \kappa_{n_2}$.

   *That is, when* $k$ *is even, the* $\frac{k}{2}$*-th row of the matrix , has two equivalent modulo* $C$ *pseudo-cycles with relation to* $\kappa$, *both with length* $\frac{k}{2}$.

**Proof.** The proof of all items follows by algebraic manipulation, using the commutativity of $*$ and Definition 36:

1.
$$\begin{array}{rcl} a_{i(j+1)} & = & (\overline{j+1-1}) * (\overline{i+j+1-1}) = \overline{(j-1)+1} * \overline{(i+j-1)+1} \\ & = & (\kappa \cdot \overline{j-1}) * (\kappa \cdot \overline{i+j-1}) = \kappa \cdot a_{ij} \end{array}$$

2.
$$\begin{array}{rcl} \kappa \cdot a_{ik} & = & (\kappa \cdot \overline{k-1}) * (\kappa \cdot \overline{i+k-1}) = \overline{k-1+1} * \overline{i+k-1+1} \\ & = & \overline{(1-1)+k} * \overline{(i+1-1)+k} = \overline{1-1} * \overline{i+1-1} = a_{i1} \end{array}$$

3.
$$\begin{array}{rcl} a_{(k-i)(i+j)} & = & \overline{i+j-1} * \overline{k-i+i+j-1} \approx_C \overline{k-i+i+j-1} * \overline{i+j-1} \\ & = & \overline{(j-1)+k} * \overline{i+j-1} = \overline{j-1} * \overline{i+j-1} = a_{ij} \end{array}$$

4. **a.** By Definition 36, it follows that
   $a_{n(n+i)} = \overline{n+i-1} * \overline{n+n+i-1} = \overline{n+i-1} * \overline{(i-1)+k} \approx_C \overline{i-1} * \overline{n+i-1} = a_{ni}$
   **b.** The proof follows directly from Definition 33 and the mapping from the rows of $M_{(k-1)\times k}$ to cycles.

◀

▶ Example B.2. Let $\rho = (a\ b\ c\ d\ e)$ be a 5-cycle, its pseudo-cyce representation via coefficients is $\kappa = (\overline{0}\,\overline{1}\,\overline{2}\,\overline{3}\,\overline{4})$, and the corresponding first-instance pseudo-cycle matrix is

$$\mathcal{M}_{3\times 4} = \begin{bmatrix} \overline{0}*\overline{1} & \overline{1}*\overline{2} & \overline{2}*\overline{3} & \overline{3}*\overline{4} & \overline{4}*\overline{0} \\ \overline{0}*\overline{2} & \overline{1}*\overline{3} & \overline{2}*\overline{4} & \overline{3}*\overline{0} & \overline{4}*\overline{1} \\ \overline{0}*\overline{3} & \overline{1}*\overline{4} & \overline{2}*\overline{0} & \overline{3}*\overline{1} & \overline{4}*\overline{2} \\ \overline{0}*\overline{4} & \overline{1}*\overline{0} & \overline{2}*\overline{1} & \overline{3}*\overline{2} & \overline{4}*\overline{3} \end{bmatrix}$$

Notice that, for instance, $a_{12} \approx_C a_{43}$ which implies that $\kappa_1 \approx_C \kappa_4$. Similarly, $a_{23} \approx_C a_{35}$ implies $\kappa_2 \approx_C \kappa_3$. Every row of $\mathcal{M}_{3\times 4}$ is a first instance pseudo-cycle of $\kappa$, with length 5.

- Assuming that $A_0 = a$, the matrix of coefficients $\mathcal{M}_{3\times 4}$ corresponds to the matrix of pseudo-cycles of $\rho$:

$$\mathcal{M}'_{3\times 4}\begin{bmatrix} \overline{a}*\overline{b} & \overline{b}*\overline{c} & \overline{c}*\overline{d} & \overline{d}*\overline{e} & \overline{e}*\overline{a} \\ \overline{a}*\overline{c} & \overline{b}*\overline{d} & \overline{c}*\overline{e} & \overline{d}*\overline{a} & \overline{e}*\overline{b} \\ \overline{a}*\overline{d} & \overline{b}*\overline{e} & \overline{c}*\overline{a} & \overline{d}*\overline{b} & \overline{e}*\overline{c} \\ \overline{a}*\overline{e} & \overline{b}*\overline{a} & \overline{c}*\overline{b} & \overline{d}*\overline{c} & \overline{e}*\overline{d} \end{bmatrix}$$

  The pseudo-cycles in rows 2 and 3 are equivalent modulo commutativity.
- When $A_0 = b, c, d$ or $e$, we obtain another matrix, whose rows are equivalent to the rows in $\mathcal{M}'_{3\times 4}$.

The following lemma shows that for a first-instance matrix $\mathcal{M} = (a_{ij})_{k-1\times k}$ w.r.t a pseudo-cycle $\kappa$, if there exist two elements equivalent modulo $C$, in a row $n$, then $k$ is even.

▶ Lemma B.1. Let $\mathcal{M} = (a_{ij})_{k-1\times k}$ be a first-instance pseudo-cycle matrix for a pseudo-cycle $\kappa$ . If there exists a positive integer $n \leq k$ such that $a_{ni} \approx_C a_{ni'}$, for some $i \neq i'$, then $k = 2n$.

**Proof.** Suppose that $a_{ni} \approx_C a_{ni'}$ with $i \neq i'$. Then, $\overline{i-1} * \overline{i+n-1} \approx_C \overline{i'-1} * \overline{i'+n-1}$. Since $i \neq i'$, it follows that $\overline{i-1} = \overline{i'+n-1}$ and $\overline{i'-1} = \overline{i+n-1}$. Therefore, $\overline{i} = \overline{i'+n}$ and $\overline{i'} = \overline{i+n}$, which imply, $\overline{i+i'} = \overline{(i+i')+2n}$. Hence, $\overline{0} = \overline{2n}$ and $k = 2n$. ◀

The next results states that every row of the first instance pseudo-cycle matrix of a pseudo-cycle $\kappa$ with odd length, is also a first instance pseudo-cycle of $\kappa$.

▶ Theorem B.1. Let $\kappa$ be a pseudo-cycle with $k$ elements and $\mathcal{M}$ its first instance matrix, with $k$ an odd number. The $k-1$ rows of $\mathcal{M}$ are first instance pseudo-cycles with relation to $\kappa$, with $k$ elements.

**Proof.** For each row $r_i = [a_{i1} \quad a_{i2} \quad \dots \quad a_{ik}]$ of $\mathcal{M}$, for $1 \leq i \leq k$, we use the mapping from Remark 4.1, to obtain the candidate a pseudo-cycle of $\kappa$ : $\kappa_i = (\overline{0} * \overline{i} \quad \overline{1} * \overline{i+1} \quad \dots \quad \overline{k-1} * \overline{i-1})$. whose elements clearly satisfy the conditions **b.** and **c.** of Definition 31. Also, since $k$ is odd, it follows from Lemma 38, that the elements of $\kappa_i$ satisfy condition **a.** of the Definition 31. ◀

▶ Lemma B.2. Let $\mathcal{M}_{(k-1)\times k}$ be first instance matrix of the pseudo-cycle $\kappa$, with $k = 2n+1$ for some positive integer $n$. If $\kappa_i$ is the pseudo-cycle in the $i$-th row of $\mathcal{M}$, for $1 \leq i \leq k-1$, then $\kappa_i \approx_C \kappa_{k-i}$.

**Proof.** By Lemma 38, $a_{(k-i)(i+j)} \approx_C a_{ij}$, for each $1 \leq i \leq k-1$, therefore, by Lemma 35, $\kappa_i \approx_C \kappa_{k-i}$. ◀

The next lemma says that the first-instance pseudo-cycle matrix of $\kappa$ contains all possible first-instance pseudo-cycles of $\kappa$.

▶ Theorem B.2. $\kappa'$ is a first-instance pseudo-cycle of $\kappa$ iff it is equivalent to a pseudo-cycle that is in a row of the first instance pseudo-cycle matrix $\mathcal{M}_{(k-1)\times k}$ of $\kappa$.

**Proof.** Let $A_0 \in \kappa'$, by definition, $A_0 = B_1 * B_2$ for some $B_1, B_2 \in \kappa$.

- If $B_1 \neq B_2$ then $A_0 = \overline{m} * \overline{n}$ with $m \neq n$ and $0 \leq m, n \leq k-1$. But for all $m, n$ exist $i, j$ so that $\overline{m} * \overline{n} \approx_C a_{ij} \in M_{(k-1)\times k}$. On one hand, if $k$ is odd and $\kappa'' = (A_0 \quad \kappa A_0 \quad \dots \quad \kappa^{(k-1)} A_0)$ then, by Theorem B.1, $\kappa''$ is a pseudo-cycle in a row of $\mathcal{M}_{(k-1)\times k}$, with $k$ elements. Besides, by Lemma 35, $\kappa' \approx_C \kappa''$. On the other hand, if $k$ is even and $\kappa'' = (A_0 \quad \kappa A_0 \quad \dots \quad \kappa^{(k-1)} A_0)$, by Lemma 38, either $\kappa''$ is equivalent to a row $i$ for $1 \leq i \leq \lfloor \frac{k-1}{2} \rfloor$, and therefore, $\kappa''$ is equivalent to a first instance pseudo-cycle of $\kappa$ with $k$ elements, or $\kappa''$ can be split in two pseudo-cycles $\kappa''_1$ and $\kappa''_2$ of length $\frac{k}{2}$, both containing an element equivalent to $A_0$, by Lemma 35, $\kappa' \approx_C \kappa''_i$ $(i = 1, 2)$, and therefore, $\kappa'$ is in a row of $M_{(k-1)\times k}$.

◀

▶ **Theorem 39.** *Let $\kappa$ be a pseudo-cycle with $k$ elements and $\mathcal{M}$ be its first instance pseudo-cycle matrix. The following properties hold*

1. *if $k$ is even, then $\kappa$ has exactly $\lfloor \frac{k-1}{2} \rfloor$ first-instance pseudo-cycles with $k$ elements, and one with $\frac{k}{2}$ elements.*
2. *if $k$ is odd, then $\kappa$ has exactly $\frac{k-1}{2}$ first-instance pseudo-cycles with $k$ elements.*

**Proof.** 1. Suppose $k = 2n$ for some positive integer $n$. By Lemma 38, it follows that rows 1 to $\lfloor \frac{k-1}{2} \rfloor$ of $\mathcal{M}$ are first instance pseudo-cycles of $\kappa$ with $k$ elements and $\kappa_i \approx_C \kappa_{k-i}$, for $1 \leq i \leq \lfloor \frac{2n-1}{2} \rfloor$. According to Lemmas 38 and B.1, the $n$-th row of $\mathcal{M}$ contains two equivalent first instance pseudo-cycles of $\kappa$ both with $\frac{k}{2}$ elements.

2. Suppose $k = 2n+1$, for some non-negative integer $n$. By Theorem B.1 the $k-1$ rows of $\mathcal{M}$ are first instance pseudo-cycles of $\kappa$ with length $k$. From Lemma 38, it follows that $\kappa_i \approx_C \kappa_{k-i}$, for $1 \leq i \leq \lfloor \frac{k-1}{2} \rfloor$ and the result follows.

◀

▶ **Theorem 40.** *Let $\kappa$ be a pseudo-cycle of length $k$. $\kappa$ contains a unitary pseudo-cycle iff $k = 2^n$, for some positive integer $n$.*

**Proof.** Let $\kappa$ be of the form $\kappa = (a_0 \quad a_1 \quad \dots \quad a_{k-1})$.
($\Leftarrow$) The proof is by induction on $n$.

- **Base Case.** $n = 1$
  In this case, $k = 2$ and the first instance pseudo-cycle matrix w.r.t. $\kappa$ and $*$ is

  $$\mathcal{M}_{1\times 2} = \begin{bmatrix} \overline{0} * \overline{1} & \overline{1} * \overline{0} \end{bmatrix}$$

  Notice that $\overline{0} * \overline{1} \approx_C \overline{1} * \overline{0}$, and this single row of $\mathcal{M}_{1\times 2}$ contains two equivalent first instance unitary pseudo-cycles $\kappa_1 = (\overline{0} * \overline{1})$ and $\kappa_2 = (\overline{1} * \overline{0})$ .
- **Induction Step.** Suppose that the result holds for $k = 2^n$. We will show that it holds for $k = 2^{n+1}$.
  Let $\kappa_l$ be a pseudo-cycle of length $l = 2^{n+1} = 2.(2^n)$. By Theorem 39, $\kappa_l$ has a first instance pseudo-cycle of length $\frac{l}{2} = 2^n$, and by IH, the result follows.

($\Rightarrow$) Let $\kappa_1$ and $\kappa_2$ be pseudo-cycles w.r.t. a pseudo-cycle $\kappa$ such that $\kappa_2$ a first-instance pseudo-cycle of $\kappa_1$. By Lemma 38 , $|\kappa_2| < |\kappa_1|$ only if $|\kappa_1| = 2.|\kappa_2|$.

Notice that $\overline{\kappa} = (\overline{a_0} \cdots \overline{a_{k-1}})$ is an immediate first-instance pseudo-cycle of $\kappa$ with $k$ elements.

- If $k = 1 = 2^0$, the result follows.

- Suppose that $k > 1$. Then, there exists a pseudo-cycle $\kappa_p$ regarding to $\kappa$, with $|\kappa_p| = 1$ only if one has a chain of pseudo-cycles $\kappa, \kappa_1, \cdots, \kappa_{p-1}, \kappa_p$, where $\kappa_1$ is a first-instance pseudo-cycle of $\kappa$, and $\kappa_{i+1}$ is a first-instance pseudo-cycle of $\kappa_i$, for all $i = 1, \cdots, (p-1)$. Besides,

$$|\kappa| = 2.|\kappa_1|,\ |\kappa_1| = 2.|\kappa_2|,\ \cdots,\ |\kappa_{p-1}| = 2.|\kappa_p| \ \text{ and } \ |\kappa_p| = 1.$$

So, $k$ must be equal to $2^p$, and the result follows.

◀

## C Correcteness of the generation of combinatorial solutions

▶ Example C.1. Consider the permutation $\pi$ with permutation cycles $(g\ h\ i\ j\ k\ l\ m\ n)$, $(a\ b\ c\ d)$ and $(s\ t)$, $\oplus$ and $\star$ commutative operators and $\Box$ and $\diamond$ function symbols in the signature. We have the following extended pseudo-cycles (see also Example 43).

- $((\overline{g} \star \overline{s})\ \ (\overline{h} \star \overline{t})\ \ (\overline{i} \star \overline{s})\ \ (\overline{j} \star \overline{t})\ \ (\overline{k} \star \overline{s})\ \ (\overline{l} \star \overline{t})\ \ (\overline{m} \star \overline{s})\ \ (\overline{n} \star \overline{t}))$;
- $((((\overline{g} \star \overline{k}) \oplus \diamond\langle \overline{b}, \overline{s}\rangle) \oplus ((\overline{i} \star \overline{m}) \oplus \diamond\langle \overline{d}, \overline{s}\rangle)) \star (((\overline{h} \star \overline{l}) \oplus \diamond\langle \overline{c}, \overline{t}\rangle) \oplus ((\overline{j} \star \overline{n}) \oplus \diamond\langle \overline{a}, \overline{t}\rangle))))$;
- $(\diamond\langle\langle(\overline{a} \oplus \overline{b}), \overline{t}\rangle \star \langle(\overline{c} \oplus \overline{d}), \overline{t}\rangle, \overline{s}\rangle \star \diamond\langle\langle(\overline{b} \oplus \overline{c}), \overline{s}\rangle \star \langle(\overline{d} \oplus \overline{a}), \overline{s}\rangle, \overline{t}\rangle)$.

For $\kappa = (a\,b\,c\,d)$ a permutation cycle of $\pi$ such that $e \notin dom(\pi)$, $\star$ a commutative operator, and $f$ and $g$ non commutative symbols, the following are extended pseudo-cycles:

1. $(f\overline{a}\ f\overline{b}\ f\overline{c}\ f\overline{d})$ (case 3.a.iv. with $f$);
2. $([e]\overline{a}\ [e]\overline{b}\ [e]\overline{c}\ [e]\overline{d})$ (case 3.a.iv.);
3. $(g\langle f\overline{a}, [e]\overline{a}\rangle\ \ g\langle f\overline{b}, [e]\overline{b}\rangle\ \ g\langle f\overline{c}, [e]\overline{c}\rangle\ \ g\langle f\overline{d}, [e]\overline{d}\rangle)$ (case 3.a.iii with cycles in itens 1 and 2 and then 3.a.iv. with $g$);
4. Combining cycles in item 3 with itself (3.a.ii) we get:
   $((g\langle f\overline{a}, [e]\overline{a}\rangle)\star(g\langle f\overline{a}, [e]\overline{a}\rangle)\ \ (g\langle f\overline{b}, [e]\overline{b}\rangle\star g\langle f\overline{b}, [e]\overline{b}\rangle)\ \ (g\langle f\overline{c}, [e]\overline{c}\rangle\star g\langle f\overline{c}, [e]\overline{c}\rangle)\ \ (g\langle f\overline{d}, [e]\overline{d}\rangle)\star g\langle f\overline{d}, [e]\overline{d}\rangle)$

Also, notice that if we have a new variable $Y$, and $\oplus$ is a commutative operator, by applying case 1, case 3.a.i, etc., we can build the extended pseudo-cycles such as:

- $(\langle t, f\langle g\langle f\overline{a}, [e]\overline{b}\rangle, Y\rangle \oplus f\langle g\langle f\overline{c}, [e]\overline{d}\rangle, Y\rangle\rangle \star \langle t, f\langle g\langle f\overline{b}, [e]\overline{c}\rangle, Y\rangle \oplus f\langle g\langle f\overline{d}, [e]\overline{a}\rangle, Y\rangle\rangle\rangle)$.

▶ Lemma C.1 (Correctness of extended pseudo-cycle solutions). For each unitary non-trivial extended pseudo-cycle $(s)$ of a permutation $\pi$, $\langle\nabla, \{X/s\}\rangle$, where $\nabla = \Delta \bigcup \cup_{Y \in s} dom(\pi)\#Y$, is a solution of $\langle\Delta, \pi.X \approx_? X\rangle$, whenever $\nabla \vdash dom(\Delta|_X)\#s$.

**Proof.** The proof follows the lines of reasoning used for non trivial unitary pseudo-cycles. By construction, the invariant that the elements of an extended pseudo-cycle of length $l$, $\kappa' = (e_0 \ldots e_{l-1})$, satisfy the property $\nabla' \vdash \pi(e_i) \approx_{\{\alpha, C\}} e_{i+1}$, where $i+1$ abbreviates $i+1$ modulo $l$, and $\nabla' = \cup_{Y \in Var(\kappa')} dom(\pi)\#Y$, holds. The only case in which the length of an extended pseudo-cycle decreases is 3.a.i. Thus, when this case applies to a binary pseudo-cycle, say $(s_0\ s_1)$, an extended unitary pseudo-cycle $(s)$ is built, being this of the form $(s_0 \oplus s_1)$ for a commutative function symbol $\oplus$. Since by the invariant we have that $\nabla' \vdash \pi(s_i) \approx_{\{\alpha, C\}} s_{i+1}$, for $i = 0, 1$, we have that $\nabla' \vdash \pi(s_0 \oplus s_1) \approx_{\{\alpha, C\}} s_0 \oplus s_1$; thus, we have that $\nabla' \vdash \pi(s) \approx_{\{\alpha, C\}} s$. In further steps in the construction of extended pseudo-cycles, new unitary pseudo cycles $(t')$ might be built from unitary extended pseudo-cycles $(t)$ applying cases 3.a.ii, iii, iv and v, that, can easily be checked, preserve the property $\nabla' \vdash \pi(t') \approx_{\{\alpha, C\}} t'$, for $\nabla' = \cup_{Y \in Var(t')} dom(\pi)\#Y$, whenever $\nabla' \vdash \pi(t) \approx_{\{\alpha, C\}} t$, for $\nabla' = \cup_{Y \in Var(t)} dom(\pi)\#Y$. Therefore all unitary non-trivial extended pseudo-cycles give a

correct solution of the form $\langle \nabla', \{X/s\}\rangle$ of the problem $\langle \emptyset, \pi.X \approx_? X\rangle$. Hence, if in addition, we have that $\nabla' \cup \Delta \vdash dom(\Delta|_X)\#s$, then $\langle \nabla' \cup \Delta, \{X/s\}\rangle$ is a solution of $\langle \Delta, \pi.X \approx_? X\rangle$, whenever $\Delta \vdash dom(\Delta|_X)\#s$. ◀

Assuming the symbols in the signature are enumerable, it is possible to enumerate the unitary extended pseudo-cycles and thus the *generated solutions*. This can be done as usual, enumerating first all possible unitary pseudo-cycles with an element of length bounded by a small natural, say twice the length of $\pi$, and using only the first $|\pi|$ symbols in the signature and atoms in $dom(\pi)$; then, this length is being increased generating all extended unitary pseudo-cycles with elements of length $|\pi|+1$ and using only the first $|\pi+1|$ symbols in the signature and atoms in $dom(\pi)$ and so on.

▶ Definition C.1 (Generated solutions of fixpoint equations). Given a unification problem $\langle \Delta, Q\rangle$ and a successful leaf of $\mathcal{T}_{\langle \Delta,Q\rangle}$, $\mathcal{P} = \langle \nabla, \sigma, P\rangle$, such that $\pi.X \approx_? X \in P$. The set of *generated solutions* for $\langle \nabla, \pi.X \approx_? X\rangle$, denoted as $\langle \nabla, \pi.X \approx_? X\rangle_{Sol_G}$, includes all solutions of the form $\langle \nabla', \{X/s\}\rangle$ where $(s)$ is an extended unitary pseudo-cycle of $\pi$ such that $\nabla' \vdash dom(\nabla|_X)\#s$, where $\nabla' = \nabla \cup_{Y \in Var(s)} (dom(\nabla|_X)\#Y \cup dom(\pi)\#Y)$.

▶ Definition C.2 (General C-matchers). Let $s_i$, for $i = 1..k$, be nominal terms. A *most general C-matcher* of these terms, if it exists, is the image of a most general C-unifier restricted to the domain $Z$, say $\{Z/t\}$, of the C-unification problem $\{s_i =_? Z\}_{i=1..k}$, where $Z$ is a new variable.

▶ Remark. Notice that to solve the set of commutative equations generated in Definition C.2 one can use the algorithm proposed by Siekmann [23], which provides a finite, minimal and complete set of C-unifiers.

Alternatively, Definition 44 could be restricted to ground terms (by removing the first case in the construction of extended pseudo-cycles), and then instead of computing $C$-matchers via $C$-unification, one could use an $\alpha$-C-equivalence checker (for example, the one specified in [3]). This would also simplify case iv in Definition 44, since it would be sufficient to consider just one atom $e'$ not in $dom(\pi)$.

▶ Definition C.3 (Generated solutions for a variable). Let $\mathcal{P} = \langle \nabla, \sigma, P\rangle$ be a successful leaf of $\mathcal{T}_{\langle \Delta,Q\rangle}$ for a unification problem $\langle \Delta, Q\rangle$, such that the set of fixpoint equations for $X$ in $P$ are given by $\pi_i.X \approx_? X$, for $\pi_i \in \Pi_X$, and such that $|\Pi_X| = k$. If there exist solutions $\langle \nabla_i, \{X/t_i\}\rangle \in \langle \nabla, \pi_i.X \approx_? X\rangle_{Sol_G}$ and for:

- $\delta$, a most general C-matcher of terms $\{t_i\}_{i=1..k}$ with $X$ as new variable and
- $\nabla'' := \nabla \cup_{i=1}^{k} \nabla_i$, and $\nabla' := \nabla'' \cup_{Y \in dom(\delta)} \cup_{Y' \in Var(Y\delta)} dom(\nabla''|_Y)\#Y'$;

it holds that for all $Y \in dom(\delta)$, $\nabla' \vdash dom(\nabla''|_Y)\#Y\delta$. In this case we say that $\langle \nabla', \{X/X\delta\}\rangle$ is a *generated solution* for $X$. The set of all generated solutions is denoted by $[X]_{\mathcal{P}_G}$.

▶ Example C.2. Let $P_i := \pi_i.X \approx_? X$, for $i = 1..3$, be fixpoint equations for $\pi_1 = (a\ b\ c\ d)$, $\pi_2 = (a\ c)$ and $\pi_3 = (b\ d)$ and suppose that $\mathcal{P} := \langle \nabla, \sigma, P\rangle$ is a successful leaf such that $P_i$ for $i = 1..3$ are the fixpoint equations for $X$ in $P$.

1. $\langle a, b, c, d\#Y, \delta_1 := \{X/((a * c) * (b * d)) \oplus Y\}\rangle \in \langle \nabla, P_1\rangle_{Sol_G}$;
2. $\langle a, c\#Y', Y'', \delta_2 := \{X/((a * c) * Y') \oplus Y''\}\rangle \in \langle \nabla, P_2\rangle_{Sol_G}$; and
3. $\langle b, d\#Y_1', Y_1'', \delta_3 := \{X/((b * d) * Y_1') \oplus Y_1''\}\rangle \in \langle \nabla, P_3\rangle_{Sol_G}$.

Notice that $\delta = \{X/((a * c) * (b * d)) \oplus Y'',\ Y'/(b * d),\ Y_1'/(a * c),\ Y/Y'', Y_1''/Y''\}$ is a most general $C$-unifier of terms $\{t_i := X\delta_i\}$ with variable $X$.

According to the definiton, the set of initial freshness constraints is given as $\nabla'' = \nabla \cup \{a, b, c, d\#Y,\ a, c\#Y', Y'',\ b, d\#Y_1', Y_1''\}$. Notice that $Y'' \in Var(im(\delta))$, have to satisfy the constraints on $Y_1'', Y$ and $X$, that is, $a, b, c, d\#Y''$ is a new constraint on $Y''$, inherited from the constraints of the variables in the domain of $\delta$.

For $\nabla' = \nabla'' \cup \{a, b, c, d\#Y''\}$, it holds that $\nabla' \vdash dom(\nabla''|_Z)\#Z\delta$, for all $Z \in dom(\delta)$. Therefore, $\langle \nabla', X/X\delta\rangle$ belongs to $[X]_{\mathcal{P}_G}$.

Now we prove that the set of generated solutions for fixpoint equations $[X]_{\mathcal{P}_G}$ is correct.

▶ Lemma C.2 (Correctness of generated solutions for a variable). Let $\mathcal{P} = \langle \nabla, \sigma, P\rangle$ be a successful leaf of $\mathcal{T}_{\langle \Delta, Q\rangle}$ for a unification problem $\langle \Delta, Q\rangle$. Any solution in $[X]_{\mathcal{P}_G}$ is a solution of each fixpoint equation for $X$ in $P$.

**Proof.** By Lemma C.1 and Definition C.1 each solution $\langle \nabla_i, \{X/t_i\}\rangle$ in $\langle \nabla, \pi_i.X \approx_?\ X\rangle_{Sol_G}$ is a correct solution for $\langle \nabla, \pi_i.X \approx_?\ X\rangle$, for $\pi_i \in \Pi_X$. Suppose $\langle \nabla', \{X/X\delta\}\rangle$ belongs to $[X]_{\mathcal{P}_G}$. Since $\delta$ is a C-unifier of terms $t_i$ with variable $X$, we have that $X\delta \approx_C t_i\delta$, and also that $\nabla_i \vdash \pi.t_i \approx_{\{\alpha, C\}} t_i$. Thus, $\nabla' \vdash \pi.t_i\delta \approx_{\{\alpha, C\}} t_i\delta$ since by definition we also have that $\nabla' \vdash dom(\nabla|_X)\#X\delta$, because by construction for all $Y \in Var(X\delta)$, $\nabla'$ includes the freshness constraints $dom(\nabla''|_X)\#Y$ and $\nabla''$ is an extension of $\nabla$. ◀

▶ Definition C.4 (Generated Solutions for Successful Leaves). Let $\langle \Delta, Q\rangle$ be a unification problem and $\mathcal{P} = \langle \nabla, \sigma, P\rangle$ a successful leaf of $\mathcal{T}_{\langle \Delta, Q\rangle}$. The set of generated solutions for $\mathcal{P}$ denoted as $[\mathcal{P}]_{Sol_G}$ is defined as the set that contains all solutions of the form

$$\left\langle \bigcup_{X \in Var(P)} \nabla_X,\quad \sigma\left(\bigcup_{X \in Var(P)} \{X/s_X\}\right)\right\rangle, \text{ where each } \langle \nabla_X, \{X/s_X\}\rangle \in [X]_{\mathcal{P}_G}.$$

▶ Corollary C.1 (Correctness of generated solutions for successful leaves).

Let $\langle \Delta, Q\rangle$ be a unification problem and $\mathcal{P} = \langle \nabla, \sigma, P\rangle$ a successful leaf of $\mathcal{T}_{\langle \Delta, Q\rangle}$. Any solution in the set of solutions $[\mathcal{P}]_{Sol_G}$ is a correct solution of $\langle \Delta, Q\rangle$.

**Proof.** A solution in this set is of the form $\left\langle \bigcup_{X \in Var(P)} \nabla_X,\quad \sigma\left(\bigcup_{X \in Var(P)} \{X/s_X\}\right)\right\rangle$, where, by previous lemma, each $\langle \nabla_X, \{X/s_X\}\rangle$ is a correct solution for all fixpoint equations in $P$ for the variable $X$. Thus, since all variables in terms $s_X$ are new ones, the solution is indeed a solution for the problem $\langle \nabla, \sigma, P\rangle$ and by Lemma 28 of the problem. ◀

A greedy procedure for the generation of solutions in $[X]_{\mathcal{P}}$ proceeds as follows. Follow the construction of generated solutions in Definition C.1 for each fixpoint problem $\langle \nabla, \pi_i.X \approx_?\ X\rangle$ in $P$, where $\pi_i \in \Pi_X$, as given in Lemma C.2; for each generated solution $\langle \nabla', \{X/s\}\rangle$ build the freshness context $\nabla'' = \nabla' \cup \bigcup_{Y \in Var(s)} dom(\nabla|_X)\#Y \cup dom(\Pi_X)\#Y$ and check whether $\langle \nabla'', \{X/s\}\rangle$ is a solution for all $\langle \nabla, \pi_i.X \approx_?\ X\rangle$, for $\pi_i \in \Pi_X$. Here, $dom(\Pi_X)\#Y$ abbreviates $\cup_{\pi_i \in \Pi_X} dom(\pi_i)\#Y$.