Nominal C-Unification

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

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

Abstract. Nominal unification is an extension of first-order unification that takes into account the α-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 (finite families) of *fixed point* problems, whose solutions can be generated by algebraic techniques on combinatorics of permutations.

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 α-equivalence. In this paper, we follow the nominal approach to the specification of binding operators [20,30,26], where the syntax of terms includes, in addition to variables, also *atoms* $(a, b, \dots)$, which can be abstracted, and α-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 function symbols $\forall$ and geq and an abstracted atom a. Nominal unification [30] is the problem of solving equations between nominal terms modulo α-equivalence; it is a decidable problem and efficient nominal unification algorithms are available [11,9,24], 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., [18,17,5]) or defined by equational axioms (see, e.g., [14,19]). The case of associative and commutative nominal theories was considered in [3], where a parametric $\{\alpha, AC\}$-equivalence relation was formalised in

* Work supported by the Brazilian agencies FAPDF (DE 193.001.369/2016), CAPES (Proc. 88881.132034/2016-01, 2nd author) and CNPq (PQ 307009/2013, 1st author).

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*. The algorithm transforms a given *nominal C-unification problem* $\langle \Delta, Q \rangle$, where Δ is a freshness context and Q a set of freshness constraints and equations, respectively of the form $a\#_? s$ and $s \approx_? t$, into a finite set of triples of the form $\langle \nabla, \sigma, P \rangle$, consisting of a freshness context ∇, a substitution σ and a set of fixed point equations P of the form $\pi.X \approx_? X$. The simplifications are based on a set deduction rules for freshness and α-C-equivalence (denoted as $\approx_{\{\alpha,C\}}$).

The role of fixed point equations (for short, FP equations) in nominal C-unification is tricky: while in standard nominal unification [30], solving a FP equation of the form $(a\ b).X \approx_? X$ reduces to checking whether the constraints $a\#X, b\#X$ (a and b fresh in X) are satisfied, and in this case the solution is the *identity* substitution, in nominal C-unification, for $*$ and $+$ commutative operators, one can have additional combinatory solutions of the form $\{X/a + b\}, \{X/(a + b) * \ldots * (a + b)\}$, $\{X/f(a) + f(b)\}$, etc. We show that in general there is no finitary representation of solutions using only freshness contexts and substitutions, hence a nominal C-unification problem may have a potentially infinite set of independent most general unifiers (unlike standard C-unification, which is well-known to be finitary).

We adapt the proof of NP-completeness of syntactic C-unification to show that nominal C-unification is NP-complete as well.

Soundness and completeness of the simplification rules were formalised in Coq (available at http://ayala.mat.unb.br/publications.html). An OCaml implementation is also available. Details of the proofs are in the appendix.

Related work: To generate the set of combinatorial solutions for FP equations we can use an enumeration procedure given in [4], which is based on the combinatorics of permutations. By combining the simplification and enumeration methods, we obtain a nominal C-unification procedure in two phases: a *simplification phase*, described in this paper, which outputs a finite set of most general solutions that may include fixed point constraints, and a *generation phase*, which eliminates the fixed point constraints according to [4].

Several extensions of the nominal unification algorithm have been defined, in addition to the equational extensions already mentioned.

An algorithm for nominal unification of higher-order expressions with recursive *let* was proposed in [23]; as in the case of nominal C-unification, FP equations are obtained in the process. Using the techniques in [4], it is possible to proceed further and generate the combinatorial solutions of FP equations.

Recently, Aoto and Kikuchi [1] proposed a rule-based procedure for nominal equivariant unication [13], an extension of nominal unification that is useful in confluence analysis of nominal rewriting systems [2,16].

Furthermore, several formalisations and implementations of the nominal unification algorithm are available. For example, formalisations of its soundness and completeness were developed by Urban et al [30,29], Ayala-Rincón et al [6], and

Kumar and Norrish [22] using, respectively, the proof assistants Isabelle/HOL, PVS and HOL4. An implementation in Maude using term graphs [10] is also available. Urban and Cheney used a nominal unification algorithm to develop a Prolog-like language called α-Prolog [12]. Our formalisation of nominal C-unification is based on the formalisation of equivalence modulo $\{\alpha, AC\}$ presented in [3]. The representations of permutations and terms are similar, but here we deal also with substitutions and unification rules, and prove soundness and completeness of the unification algorithm.

Non nominal reasoning modulo equational theories has been subject of formalisations. For instance, Nipkow [25] presented a set of Isabelle/HOL tactics for reasoning modulo A, C and AC; Braibant and Pous [8] designed a plugin for Coq, with an underlying AC-matching algorithm, that extends the system tactic `rewrite` to deal with AC function symbols; also, Contejean [15] formalised in Coq the correction of an AC-matching algorithm implemented in C*i*ME.

Syntactic unification with commutative operators is an NP-complete problem and its solutions can be finitely generated [21,28]. Since C-unification problems are a particular case of nominal C-unification problems, our simplification algorithm, checked in Coq, is also a formalisation of the C-unification algorithm.

Organisation: Sec. 2 presents basic concepts and notations. Sec. 3 introduces the formalised equational and freshness inference rules for nominal C-unification, and briefly discusses NP-completeness; Sec. 4 shows that a single fixed point equation can have infinite independent solutions; Sec. 5 shortly discusses the formalisation in Coq and Sec. 6 concludes and proposes future work.

2 Background

Consider countable disjoint sets of variables $\mathcal{X} := \{X, Y, Z, \cdots\}$ and atoms $\mathcal{A} := \{a, b, c, \cdots\}$. A *permutation* π is a bijection on $\mathcal{A}$ with a finite *domain*, where the domain (i.e., the *support*) of π is the set $dom(\pi) := \{a \in \mathcal{A} \mid \pi \cdot a \neq a\}$. The inverse of π is denoted by π^{-1}. Permutations can be represented by lists of *swappings*, which are pairs of different atoms $(a\, b)$; hence a *permutation* π is a finite list of the form $(a_1\, b_1) :: \ldots :: (a_n\, b_n) :: nil$, where the empty list *nil* corresponds to the identity permutation; concatenation is denoted by $\oplus$ and, when no confusion may arise, :: and *nil* are omitted. We follow Gabbay's permutative convention: Atoms differ on their names, so for atoms a and b the expression $a \neq b$ is redundant. Also, $(a\, b)$ and $(b\, a)$ represent the same swapping.

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.

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 $nil.X$ 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: $Var(S) = \bigcup_{t \in S} Var(t)$. As usual, $|\,\_\,|$ will be used to denote the cardinality of sets as well as to denote the size or number of symbols occurring in a given term.

Definition 1 (Permutation action). *The action of a permutation on atoms is defined as: $nil \cdot a := a$; $(b\,c) :: \pi \cdot a := \pi \cdot a$; and, $(b\,c) :: \pi \cdot b := \pi \cdot c$. The action of a permutation on terms is defined recursively as:*

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

Notice that according to the definition of the action of a permutation over atoms, the composition of permutations π and π', usually denoted as $\pi \circ \pi'$, corresponds to the append $\pi' \oplus \pi$. Also notice that $\pi' \oplus \pi \cdot t = \pi \cdot (\pi' \cdot t)$. The *difference set* between two permutations π and π' is the set of atoms where the action of π and π' differs: $ds(\pi, \pi') := \{a \in \mathcal{A} \mid \pi \cdot a \neq \pi' \cdot a\}$.

A *substitution* σ is a mapping from variables to terms such that its *domain*, $dom(\sigma) := \{X \mid X \neq X\sigma\}$, is finite. For $X \in dom(\sigma)$, $X\sigma$ is called the *image* of X. Define the *image* of σ as $im(\sigma) := \{X\sigma \mid X \in dom(\sigma)\}$. Let $dom(\sigma) = \{X_1, \cdots, X_n\}$, then σ can be represented as a set of *bindings* in the form $\{X_1/t_1, \cdots, X_n/t_n\}$, where $X_i\sigma = t_i$, for $1 \leq i \leq n$.

Definition 2 (Substitution action). *The* action of a substitution σ *on a term t, denoted $t\sigma$, is defined recursively as follows:*

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

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

Lemma 1 (Substitutions and Permutations Commute). $(\pi \cdot t)\sigma = \pi \cdot (t\sigma)$

The inference rules defining freshness and α-equivalence are given in Fig. 1 and 2. The symbols ∇ and Δ 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 $dom(\nabla)$ is the set of atoms appearing in it; $\nabla|_X$ denotes the restriction of ∇ to the freshness constraints on X: $\{a\#X \mid a\#X \in \nabla\}$. The rules in Fig. 1 are used to check if an atom a is fresh in a nominal term t under a freshness context ∇, also denoted as $\nabla \vdash a\#t$. The rules in Fig. 2 are used to check if two nominal terms s and t are α-equivalent under some freshness context ∇, 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]**).

Example 1. Let $\sigma = \{X/[a]a\}$. Verify that $\langle (a\ b).X, f(e)\rangle\sigma \approx_\alpha \langle X, f(e)\rangle\sigma$.

By $dom(\pi)\#X$ and $ds(\pi, \pi')\#X$ we abbreviate the sets $\{a\#X \mid a \in dom(\pi)\}$ and $\{a\#X \mid a \in ds(\pi, \pi')\}$, respectively.

Key properties of the nominal freshness and α-equivalence relations have been extensively explored in previous works [3,6,29,30].

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

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

Fig. 1. Rules for the freshness relation

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

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

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

Fig. 2. Rules for the relation $\approx_\alpha$

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 α-equivalence modulo commutativity, denoted as $\approx_{\{\alpha,C\}}$. 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.

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

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

Fig. 3. Additional rules for $\{\alpha, C\}$-equivalence

The following properties for $\approx_{\{\alpha,C\}}$ were formalised as simple adaptations of the formalisations given in [3] for $\approx_\alpha$.

Lemma 2 (Inversion). *The inference rules of $\approx_{\{\alpha,C\}}$ are* invertible.

This means, for instance, that for rules ($\approx_\alpha$ [**ab**]) one has $\nabla \vdash [a]s \approx_{\{\alpha,C\}} [b]t$ implies $\nabla \vdash s \approx_{\{\alpha,C\}} (a\,b) \cdot t$ and $\nabla \vdash a \,\#\, t$; and for ($\approx_{\{\alpha,\mathbf{C}\}}$ **app**), $\nabla \vdash f_k^C\; \langle s_0, s_1\rangle \approx_{\{\alpha,C\}} f_k^C\; \langle t_0, t_1\rangle$ 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 $\nabla \vdash s_1 \approx_{\{\alpha,C\}} t_0$.

Lemma 3 (Freshness preservation). *If $\nabla \vdash a \,\#\, s$ and $\nabla \vdash s \approx_{\{\alpha,C\}} t$ then $\nabla \vdash a \,\#\, t$.*

Lemma 4 (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 5 (Equivariance). *$\nabla \vdash \pi \cdot s \approx_{\{\alpha,C\}} \pi \cdot t$ whenever $\nabla \vdash s \approx_{\{\alpha,C\}} t$.*

Lemma 6 (Equivalence). $\_ \vdash \_ \approx_{\{\alpha,C\}} \_$ *is an equivalence relation.*

Remark 1. According to the grammar for nominal terms, function symbols have no fixed arity: any function symbol can apply to any term. Despite this, in the syntax of our Coq formalisation commutative symbols apply only to tuples.

3 A nominal C-unification algorithm

Inference rules are given that transform a nominal C-unification problem into a finite family of problems that consist exclusively of FP equations of the form $\pi.X \approx_? X$, together with a substitution and a set of freshness constraints.

Definition 3 (Unification problem). *A* unification problem *is a pair* $\langle \nabla, P\rangle$, *where* ∇ *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.*

Equations of the form $\pi.X \approx_? X$ are called *fixed point* equations. Given $\langle \nabla, P\rangle$, by $P_{\approx}, P_{\#}, P_{\mathrm{fp}_{\approx}}$ and $P_{\mathrm{nfp}_{\approx}}$ we will resp. denote the sets of equations, freshness constraints, FP equations and non FP equations in the set P.

Example 2. Given the nominal unification problem $\mathcal{P} = \langle \emptyset, \{[a][b]X \approx_? [b][a]X\}\rangle$, the standard unification algorithm [30] reduces it to $\langle \emptyset, \{X \approx_? (a\, b).X\}\rangle$, which gives the solution $\langle \{a\#X, b\#X\}, id\rangle$. However, we will see that infinite independent solutions are feasible when there is at least a commutative operator.

We design a nominal C-unification algorithm using one set of transformation rules to deal with equations (Fig. 4) and another set of rules to deal with freshness constraints and contexts (Fig. 5). These rules act over triples of the form $\langle \nabla, \sigma, P\rangle$, where σ 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 2. Let ∇ and ∇' be freshness contexts and σ and σ' be substitutions.

- $\nabla' \vdash \nabla\sigma$ denotes that $\nabla' \vdash a \# X\sigma$ holds for each $(a\#X) \in \nabla$, and
- $\nabla \vdash \sigma \approx \sigma'$ that $\nabla \vdash X\sigma \approx_{\{\alpha,C\}} X\sigma'$ for all X (in $dom(\sigma) \cup dom(\sigma')$).

Definition 4 (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. $\nabla \vdash a \# t\sigma$, *if* $a\#_? t \in P$;
3. $\nabla \vdash s\sigma \approx_{\{\alpha,C\}} t\sigma$, *if* $s \approx_? t \in P$;
4. *there is a substitution* λ *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 5 (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* $\langle \nabla, \sigma\rangle \preccurlyeq \langle \nabla', \sigma'\rangle$, *if there exists a substitution* λ *satisfying* $\nabla' \vdash \sigma\lambda \approx \sigma'$ *and* $\nabla' \vdash \nabla\lambda$. *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* $\mathcal{V}$ *such that* $\langle \nabla, \sigma\rangle \preccurlyeq \langle \nabla', \sigma'\rangle$.

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

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

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

$$\frac{\langle\nabla, \sigma, P \uplus \{f_k^C\, s \approx_? f_k^C\, t\}\rangle}{\langle\nabla, \sigma, P \cup \{s \approx_? v\}\rangle}\ , \left\{\begin{array}{l}\text{where } s = \langle s_0, s_1\rangle \text{ and } t = \langle t_0, t_1\rangle \\ v = \langle t_i, t_{(i+1)\, mod\, 2}\rangle, i = 0, 1\end{array}\right\}\ (\approx_?\ \mathbf{C})$$

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

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

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

Fig. 4. Reduction rules for equational problems

The unification algorithm proceeds by simplification. Derivation with rules of Figs. 4 and 5 is respectively denoted by $\Rightarrow_\approx$ and $\Rightarrow_\#$. 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.

The only rule that can generate branches is ($\approx_?$ **C**), which is 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 3): either $\langle s_0, s_1\rangle \approx_? \langle t_0, t_1\rangle$ or $\langle s_0, s_1\rangle \approx_? \langle t_1, t_0\rangle$.

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 ∇ is extended to ∇' in such a way that both X_1 and X_2 inherit all freshness constraints of X in ∇: $\nabla' = \nabla \cup \{a\#X_i \mid i = 1, 2, \text{ and } a\#X \in \nabla\}$.

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

Example 3. Let $*$[3] be a commutative function symbol. Below, we show how the problem $\mathcal{P} = \langle\emptyset, \{[e](a\,b).X * Y \approx_? [f](a\,c)(c\,d).X * Y\}\rangle$ reduces (via rules in

[3] Infix notation is adopted for commutative symbols: $s * t$ abbreviates $*\langle s, t\rangle$.

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

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

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

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

Fig. 5. Reduction rules for freshness problems

Figs. 4 and 5). Application of rule ($\approx_? \mathbf{C}$) gives two branches which reduce into two fixed point problems: $\mathcal{Q}_1$ and $\mathcal{Q}_2$. Highlighted terms show where the rules are applied. For brevity, let $\pi_1 = (a\,c)(c\,d)(e\,f)$, $\pi_2 = (a\,b)(e\,f)(c\,d)(a\,c)$, $\pi_3 = (a\,c)(c\,d)(e\,f)(a\,b)$ and $\sigma = \{X/(e\,f)(a\,b).Y\}$.

$$
\begin{array}{ll}
\multicolumn{2}{l}{\langle \emptyset, id, \{ [e](a\,b).X * Y \approx_? [f](a\,c)(c\,d).X * Y \}\rangle \qquad\qquad \Rightarrow_{(\approx_? [\mathbf{ab}])}} \\
\multicolumn{2}{l}{\langle \emptyset, id, \{ (a\,b).X * Y \approx_? \pi_1.X * (e\,f).Y , e \#_? (a\,c)(c\,d).X * Y\}\rangle \Rightarrow_{(\approx_? \mathbf{C})}} \\
\text{branch 1:} & \langle \emptyset, id, \{ (a\,b).X \approx_? \pi_1.X , Y \approx_? (e\,f).Y , e \#_? (a\,c)(c\,d).X * Y\}\rangle \\
\Rightarrow_{(\approx_? \mathbf{inv})}(2\times) & \langle \emptyset, id, \{(a\,b)[\pi_1]^{-1}.X \approx_? X, [(e\,f)]^{-1}.Y \approx_? Y, e \#_? (a\,c)(c\,d).X * Y \}\rangle \\
\Rightarrow_{\substack{(\#_? \mathbf{app}),\\ (\#_? \mathbf{pair})}} & \langle \emptyset, id, \{\pi_2.X \approx_? X, (e\,f).Y \approx_? Y, e \#_? (a\,c)(c\,d).X , e \#_? Y \}\rangle \\
\Rightarrow_{(\#_? \mathbf{var})}(2\times) & \langle \{e \# X, e \# Y\}, id, \{\pi_2.X \approx_? X, (e\,f).Y \approx_? Y\}\rangle = \mathcal{Q}_1 \\
\text{branch 2:} & \langle \emptyset, id, \{ (a\,b).X \approx_? (e\,f).Y , Y \approx_? \pi_1.X, e \#_? (a\,c)(c\,d).X * Y\}\rangle \\
\Rightarrow_{(\approx_? \mathbf{inst})} & \langle \emptyset, \sigma, \{ Y \approx_? (a\,c)(c\,d)(e\,f)(e\,f)[(a\,b)]^{-1}.Y , e \#_? \pi_1[(a\,b)]^{-1}.Y * Y\}\rangle \\
\Rightarrow_{(\approx_? \mathbf{inv})} & \langle \emptyset, \sigma, \{[(a\,c)(c\,d)(a\,b)]^{-1}.Y \approx_? Y, e \#_? \pi_3.Y * Y \}\rangle \\
\Rightarrow_{\substack{(\#_? \mathbf{app}),\\ (\#_? \mathbf{pair})}} & \langle \emptyset, \sigma, \{(a\,b)(c\,d)(a\,c).Y \approx_? Y, e \#_? \pi_3.Y , e \#_? Y \}\rangle \\
\Rightarrow_{(\#_? \mathbf{var})}(2\times) & \langle \{e \# Y, f \# Y\}, \sigma, \{(a\,b)(c\,d)(a\,c).Y \approx_? Y\}\rangle = \mathcal{Q}_2
\end{array}
$$

A visual representation of the derivation tree for this example, generated using our OCaml implementation, is depicted in Fig. 8 Appendix C. Please see Appendix C for more examples.

Definition 6 (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. $\Rightarrow_{\#}$).*

Definition 7 (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 FP 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 FP equations and, possibly, freshness constraints).*

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

Definition 8 (Fail and success for $\Rightarrow_{\#}$). *Let $\mathcal{Q} = \langle \nabla, \sigma, Q\rangle$ be a successful triple regarding $\Rightarrow_{\approx}$, and $\mathcal{Q}' = \langle \nabla', \sigma, Q'\rangle$ its normal form via 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 3. Since in a successful triple regarding $\Rightarrow_{\approx}$, $\mathcal{Q}$, one has only FP equations and $\Rightarrow_{\#}$ acts only over freshness constraints, Q' in the definition above contains only FP 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 $a\#_{?}\bar{a}$.

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

Definition 9 (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 4. A substitution σ in a valid triple $\mathcal{P}$ is *idempotent*, that is, $\sigma\sigma = \sigma$.

Lemma 7 is proved by case analysis on the rules used by $\Rightarrow_{\approx}$ and $\Rightarrow_{\#}$.

Lemma 7 (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.*

From now on, we consider only valid triples.

Lemma 8 (Termination of $\Rightarrow_{\approx}$ and $\Rightarrow_{\#}$). *There is no infinite chain of reductions $\Rightarrow_{\approx}$ (or $\Rightarrow_{\#}$) starting from an arbitrary triple $\mathcal{P} = \langle \nabla, \sigma, P \rangle$.*

Proof. – The proof for $\Rightarrow_{\approx}$ 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 this measure 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 $\|P\| > \|P'\|$; and, for ($\approx_{?}$ **inv**), both $|Var(P_{\approx})| = |Var(P' \approx)|$ and $\|P\| = \|P'\|$, but $|P_{\mathrm{nfp}_{\approx}}| > |P'_{\mathrm{nfp}_{\approx}}|$.

– The proof for $\Rightarrow_{\#}$ is by induction on $\mathcal{P}$ using as measure $\|P_{\#}\|$. It can be checked 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 builds the derivation tree for $\Rightarrow_{\approx}$, labelling the root node with $\langle \nabla, id, P \rangle$. 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 FP equations. Since $\Rightarrow_{\approx}$ and $\Rightarrow_{\#}$ are both terminating (Lemma 8), the process described above must be also terminating.

Definition 10 (Derivation tree for $\langle \Delta, P \rangle$). *A* derivation tree *for the unification problem $\langle \Delta, P \rangle$, denoted as $\mathcal{T}_{\langle \Delta, P \rangle}$, is a tree with root label $\mathcal{P} = \langle \Delta, id, P \rangle$ 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 5. For $\langle \Delta, P \rangle$, all labels in the nodes of $\mathcal{T}_{\langle \Delta, P \rangle}$ are *valid* by Lemma 7.

The next lemma is proved by case analysis on elements of $\mathcal{P}_{\Rightarrow_{\approx}}$ and $\mathcal{P}_{\Rightarrow_{\#}}$.

Lemma 9 (Characterisation of leaves of $\mathcal{T}_{\langle \Delta, P \rangle}$). *Let $\langle \Delta, P \rangle$ be a unification problem. If $\mathcal{P}' = \langle \nabla, \sigma', P' \rangle$ is the label of a leaf in $\mathcal{T}_{\langle \Delta, P \rangle}$, then P' can be partitioned as follows: $P' = P'' \cup P_{\perp}$, where P'' is the set of all FP equations in P' and $P_{\perp} = P' - P''$. If $P_{\perp} \neq \emptyset$ then $\mathcal{U}_C(\mathcal{P}') = \emptyset$.*

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

Definition 11 (Successful leaves). *Let $\langle \Delta, P \rangle$ be a unification problem. A leaf in $\mathcal{T}_{\langle \Delta, P \rangle}$ that is labelled with a triple of the form $\mathcal{Q} = \langle \nabla, \sigma, Q \rangle$, where Q consists only of FP equations, is called a* successful leaf *of $\mathcal{T}_{\langle \Delta, P \rangle}$. In this case $\mathcal{Q}$ is called a* successful triple *of $\mathcal{T}_{\langle \Delta, P \rangle}$. The sets of successful leaves and triples of $\mathcal{T}_{\langle \Delta, P \rangle}$ are denoted respectively by $SL(\mathcal{T}_{\langle \Delta, P \rangle})$ and $ST(\mathcal{T}_{\langle \Delta, P \rangle})$.*

The soundness theorem states that successful leaves of $\mathcal{T}_{\langle \Delta, P \rangle}$ produce *correct* solutions. The proof is by induction on the number of steps of $\Rightarrow_{\approx}$ and $\Rightarrow_{\#}$ and uses Lemma 9 and auxiliary results on the *preservation of solutions* by $\Rightarrow_{\approx}$ and $\Rightarrow_{\#}$. Proving preservation of solutions for rules ($\approx_?$ **[ab]**) and ($\approx_?$ **inst**) is not straightforward and uses Lemmas 1 2, 3 and 5 to check that the four conditions of Def. 4 are valid before, if one supposes their validity after the rule application.

Theorem 1 (Soundness of $\mathcal{T}_{\langle \Delta, P \rangle}$). *$\mathcal{T}_{\langle \Delta, P \rangle}$ is correct, i.e., if $\mathcal{P}' = \langle \nabla, \sigma, P' \rangle$ is the label of a leaf in $\mathcal{T}_{\langle \Delta, P \rangle}$, then 1. $\mathcal{U}_C(\mathcal{P}') \subseteq \mathcal{U}_C(\langle \Delta, id, P \rangle)$, and 2. if P' contains non FP equations or freshness constraints then $\mathcal{U}_C(\mathcal{P}') = \emptyset$.*

The completeness theorem guarantees that the set of successful triples provides a complete set of solutions. Its proof uses case analysis on the rules of the relations $\Rightarrow_{\approx}$ and $\Rightarrow_{\#}$ by an argumentation similar to the one used for Theorem 1. For $\Rightarrow_{\#}$ one has indeed equivalence: $\mathcal{P} \Rightarrow_{\#} \mathcal{P}'$, implies $\mathcal{U}_C(\mathcal{P}) = \mathcal{U}_C(\mathcal{P}')$. The same is true for all rules of the relation $\Rightarrow_{\approx}$ except the branching rule ($\approx_?$ **C**), for which it is necessary to prove that all solutions of a triple reduced by ($\approx_?$ **C**) must belong to the set of solutions of one of its children triples.

Theorem 2 (Completeness of $\mathcal{T}_{\langle\Delta,P\rangle}$). *Let $\langle\Delta, P\rangle$ and $\mathcal{T}_{\langle\Delta,P\rangle}$ be a unification problem and its derivation tree. Then $\mathcal{U}_C(\langle\Delta, id, P\rangle) = \bigcup_{\mathcal{Q}\in ST(\mathcal{T}_{\langle\Delta,P\rangle})} \mathcal{U}_C(\mathcal{Q})$.*

Corollary 1 (Generality of successful triples). *Let $\mathcal{P} = \langle\Delta, P\rangle$ be a unification problem and $\langle\nabla'', \sigma'\rangle \in \mathcal{U}_C(\mathcal{P})$. Then there exists a successful triple $\mathcal{Q} \in ST(\mathcal{T}_{\langle\Delta,P\rangle})$ where $\mathcal{Q} = \langle\nabla, \sigma, Q\rangle$ such that $\langle\nabla'', \sigma'\rangle \in \mathcal{U}_C(\mathcal{Q})$, and hence, $\nabla'' \vdash \nabla\sigma'$ and there exists λ such that $\nabla'' \vdash \sigma\lambda \approx \sigma'$.*

Proof. By Theorem 2, $\mathcal{U}_C(\mathcal{P}) = \bigcup_{\mathcal{P}'\in ST(\mathcal{T}_{\langle\Delta,P\rangle})} \mathcal{U}_C(\mathcal{P}')$. Then there exists $\mathcal{Q} \in ST(\mathcal{T}_{\langle\Delta,P\rangle})$ such that $\langle\nabla'', \sigma'\rangle \in \mathcal{U}_C(\mathcal{Q})$. Suppose $\mathcal{Q} = \langle\nabla, \sigma, Q\rangle$. Then by the first and fourth conditions of the definition of solution (Def. 4) we have that $\nabla'' \vdash \nabla\sigma'$ and there exists λ such that $\nabla'' \vdash \sigma\lambda \approx \sigma'$.

Remark 6. The nominal C-unification problem is to decide, for a given $\mathcal{P}$, if $\mathcal{U}_C(\mathcal{P})$ is non empty; that is, whether $\mathcal{P}$ has nominal C-unifiers. To prove that this problem is in NP, a non-deterministic procedure using the reduction rules in the same order as in Definition 10 is designed. In this procedure, whenever rule ($\approx_?$ **C**) applies, only one of the two possible branches is guessed. In this manner, if the derivation tree has a successful leaf, this procedure will guess a path to the successful leaf, answering positively to the decision problem. According to the measures used in the proof of termination (Lemma 8), reduction with both the relations $\Rightarrow_\approx$ and $\Rightarrow_\#$ is polynomially bound, which implies that this non-deterministic procedure is polynomially bound.

To prove NP-completeness, one can polynomially reduce the well-known NP-complete positive 1-in-3-SAT problem into nominal C-unification, as done in [7] for the C-unification problem. An instance of the positive 1-in-3-SAT problem consists of a set of clauses $\mathcal{C} = \{\mathcal{C}_i | 1 \leq i \leq n\}$, where each $\mathcal{C}_i$ is a disjunction of three propositional variables, say $\mathcal{C}_i = p_i \vee q_i \vee r_i$. A solution of $\mathcal{C}$ is a valuation with exactly one variable true in each clause. The proposed reduction of $\mathcal{C}$ into a nominal C-unification problem would require just a commutative function symbol, say $\oplus$, two atoms, say a and b, a variable for each clause $\mathcal{C}_i$, say Y_i, and a variable for each propositional variable p in $\mathcal{C}$, say X_p. Instantiating X_p as $\overline{a}$ or $\overline{b}$, would be interpreted as evaluating p as true or false, respectively. Each clause $\mathcal{C}_i = p_i \vee q_i \vee r_i$ in $\mathcal{C}$ is translated into an equation E_i of the form $((X_{p_i} \oplus X_{q_i}) \oplus X_{r_i}) \oplus Y_i \approx_? ((\overline{b} \oplus \overline{b}) \oplus \overline{a}) \oplus ((\overline{b} \oplus \overline{a}) \oplus \overline{b})$. The nominal C-unification problem for $\mathcal{C}$ is given by $\mathcal{P}_\mathcal{C} = \langle\emptyset, \{E_i | 1 \leq i \leq n\}\rangle$. Simplifying $\mathcal{P}_\mathcal{C}$ would not introduce freshness constraints since the problem does not include abstractions. Thus, to conclude it is only necessary to check that $\langle\emptyset, \sigma\rangle$ is a solution for $\mathcal{P}_\mathcal{C}$ if and only if σ instantiates exactly one of the variables X_{p_i}, X_{q_i} and X_{r_i} in each equation with $\overline{a}$ and the other two with $\overline{b}$, which means that $\mathcal{C}$ has a solution.

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

To build solutions for a successful leaf $\mathcal{P} = \langle\nabla, \sigma, P\rangle$ in the derivation tree of a given unification problem, we will select and combine solutions generated for FP equations $\pi.X \approx_? X$, for each $X \in Var(P)$. We introduce the notion of *pseudo-cycle of a permutation*, in order to provide precise conditions to build terms t by combining the atoms in $dom(\pi)$, such that $\pi \cdot t \approx_{\{\alpha,C\}} t$. For convenience, we 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 [27].

Example 4. (Continuing Example 3) The permutations $(a\,b)::(e\,f)::(c\,d)::(a\,c)::$ nil and $(a\,b)::(c\,d)::(a\,c)::nil$ are respectively represented as the product of permutation cycles $(a\,b\,c\,d)(e\,f)$ and $(a\,b\,c\,d)(e)(f)$.

Permutation cycles of length one are omitted. In general the cyclic representation of a permutation consists of the product of all its cycles.

Let π be a permutation with $dom(\pi) = n$. Given $a \in dom(\pi)$ 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: $(a\,b\,c\,d) = (b\,c\,d\,a) = (c\,d\,a\,b) = (d\,a\,b\,c)$.

Def. 12 establishes the notion of a *pseudo-cycle w.r.t. a k-cycle κ*. Intuitively, given a k-cycle κ and a commutative function symbol $*$, a pseudo-cycle w.r.t κ, $(A_0 \ldots A_l)$, is a cycle whose elements are either atom terms built from the atoms in κ or terms of the form $A'_i * A'_j$, for A'_i, A'_j elements of a pseudo-cycle w.r.t κ.

Definition 12 (Pseudo-cycle). *Let $\kappa = (a_0\ a_1\ \ldots\ a_{k-1})$ be a k-cycle of a permutation π. A* pseudo-cycle *w.r.t. κ is inductively defined as follows:*

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

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

Example 5. A (Continuing Example 2) The unitary pseudo-cycles of $\kappa = (a\,b)$ are of the form $(\overline{a} * \overline{b})$ for $*$ any commutative symbol in the signature. These pseudo-cycles are the basis for a more elaborated construction used to build infinite independent solutions for the leaf $\langle \emptyset, id, \{X \approx_? (a\,b).X\} \rangle$. Examples of these solutions are: $\langle \emptyset, \{X/\overline{a}*\overline{b}\} \rangle$, $\langle \emptyset, \{X/(\overline{a}*\overline{a})*(\overline{b}*\overline{b})\} \rangle$, $\langle \emptyset, \{X/(\overline{a}*\overline{b})*(\overline{a}*\overline{b})\} \rangle$, $\langle \emptyset, \{X/((\overline{a} * \overline{a}) * \overline{a}) * ((\overline{b} * \overline{b}) * \overline{b})\} \rangle$, $\langle \emptyset, \{X/(\overline{a} * (\overline{a} * \overline{a})) * (\overline{b} * (\overline{b} * \overline{b}))\} \rangle$, etc.

B (Continuing Examples 3 and 4) In $\mathcal{Q}_1$ and $\mathcal{Q}_2$ we have the occurrences of the 4-cycle $\kappa = (a\,b\,c\,d)$. Suppose $*, \oplus, +$ are commutative operators in the signature. The following are pseudo-cycles w.r.t. κ: $\overline{\kappa} = (\overline{a}\ \overline{b}\ \overline{c}\ \overline{d})$; $\kappa_1 = ((\overline{a}*\overline{b})\ (\overline{b}*\overline{c})\ (\overline{c}*\overline{d})\ (\overline{d}*\overline{a}))$; $\kappa_2 = ((\overline{a}\oplus\overline{c})\ (\overline{b}\oplus\overline{d}))$; $\kappa_{11} = (((\overline{a}*\overline{b})+(\overline{b}*\overline{c}))\ ((\overline{b}*\overline{c})+(\overline{c}*\overline{d}))((\overline{c}*\overline{d})+(\overline{d}*\overline{a}))\ ((\overline{d}*\overline{a})+(\overline{a}*\overline{b})))$; $\kappa_{12} = (((\overline{a}*\overline{b})*(\overline{c}*\overline{d}))\ ((\overline{b}*\overline{c})*(\overline{d}*\overline{a})))$;

$\kappa_{21} = (((\overline{a} \oplus \overline{c}) * (\overline{b} \oplus \overline{d})))$; $\kappa_{121} = (((\overline{a} * \overline{b}) * (\overline{c} * \overline{d})) * ((\overline{b} * \overline{c}) * (\overline{d} * \overline{a})))$. κ_1 and κ_2 are first-instance pseudo-cycles of $\overline{\kappa}$, and κ_{11} and κ_{12} of κ_1 and κ_{21} of κ_2. Notice that, $|\overline{\kappa}| = |\kappa_1| = |\kappa_{11}| = 4$, $|\kappa_{12}| = 2$, and $|\kappa_{21}| = |\kappa_{121}| = 1$. Also, κ_1 corresponds to $((\overline{a} * \overline{d})\ (\overline{b} * \overline{a})\ (\overline{c} * \overline{b})\ (\overline{d} * \overline{c}))$, a first-instance pseudo-cycle of $\overline{\kappa}$. Finally, observe that for the elements of the unitary pseudo-cycles κ_{21} and κ_{121}, say $s = (\overline{a} \oplus \overline{c}) * (\overline{b} \oplus \overline{d})$ and $t = ((\overline{a} * \overline{b}) * (\overline{c} * \overline{d})) * ((\overline{b} * \overline{c}) * (\overline{d} * \overline{a}))$, $\{X/s\}$ and $\{X/t\}$ (resp. $\{Y/s\}$ and $\{Y/t\}$) are solutions of the FP equation $(a\ b\ c\ d)(e\ f).X \approx_? X$ (resp. $(a\ b\ c\ d).Y \approx_? Y$).

Let κ be a pseudo-cycle. Notice that only item 2 of Def. 12 may build a first-instance pseudo-cycle κ' w.r.t. κ with fewer elements. If $|\kappa'| < |\kappa|$ then, due to algebraic properties of cycles and commutativity of the operator applied $(*)$, one must have that $|\kappa'| = |\kappa|/2$. Thus, unitary pseudo-cycles can only be generated from cycles of length a power of two. This is the intuition behind the next theorem, proved by induction on the size of the cycle κ.

Theorem 3. *A pseudo-cycle κ generates unitary pseudo-cycles iff $|\kappa|$ is a power of two.*

Notice that, according item 2.c of Def. 12, if $\kappa' = (A_0 \ldots A_{k'-1})$ is a pseudo-cycle w.r.t. π then $\pi \cdot A_{k'-1} \approx_{\{\alpha,C\}} A_0$; particularly, if $k' = 1$ then $\pi \cdot A_0 \approx_{\{\alpha,C\}} A_0$. Below, given $\mathcal{P} = \langle \emptyset, \{\pi.X \approx_? X\} \rangle$ a fixed point equational problem, we call a *combinatory solution* of $\mathcal{P}$, a substitution $\{X/t\}$, such that $\pi \cdot t \approx_C t$, and t contains only atoms from π and commutative function symbols, built as unary pseudo-cycles w.r.t. κ a cycle in π.

The next theorem is proved by contradiction, supposing that κ has an odd factor and using Theorem 3.

Theorem 4. *Let $\mathcal{P} = \langle \emptyset, \{\pi.X \approx_? X\} \rangle$ be a fixed point problem. $\mathcal{P}$ has a combinatory solution iff there exists a unitary pseudo-cycle κ w.r.t. π.*

Remark 7. Since one can generate infinitely many unitary pseudo-cycles from a given 2^n-cycle κ in π, $n \in \mathbb{N}$, there exist infinite independent solutions for the fixed point problem $\langle \emptyset, \{\pi.X \approx_? X\} \rangle$.

General solutions for fixed point problems. To compute the set of solutions for a FP equation, we use a method described in [4], which is based on the computation of unitary *extended pseudo-cycles* (`epc`). We refer to [4] for the definition of extended pseudo-cycles and an algorithm to enumerate all the solutions of a successful leaf in the derivation tree.

Pseudo-cycles are built just from atom terms in $dom(\pi)$ and commutative function symbols, while `epc`'s consider all nominal syntactic elements including new variables, and also non commutative function symbols. The soundness and completeness of the generator of solutions described in [4] relies on the properties of pseudo-cycles described above, in particular the fact that only unitary pseudo-cycles generate solutions.

5 Formal Proofs

In the Coq formalisation, nominal terms are specified inductively, which permits to use induction to formalise properties of terms (to check nominal α-equality modulo C we use the rules given in [3]; see Fig. 3). The relations $\Rightarrow_{\#}$ and $\Rightarrow_{\approx}$ are inductivelty specified, as propositions from problems to problems, resp. as fresh_sys and equ_sys, and normal forms and their reflexive-transitive closures are specified using abstract relations as shown below.

```
Definition NF (T:Type) (R:T→T→Prop) (s:T) := ∀ t, ¬ R s t.

Inductive tr_clos (T:Type) (R:T→T→Prop) : T→T→Prop :=
 | tr_rf : ∀ s, tr_clos T R s s
 | tr_os : ∀ s t, R s t → tr_clos T R s t
 | tr_ms : ∀ s t u, R s t → tr_clos T R t u → tr_clos T R s u
```

A unification step, unif_step, is a reduction step either with the relation equ_sys or with the relation fresh_sys, the latter restricted to fixed point problems; and a leaf is a normal form for this relation.

```
Inductive unif_step : Triple → Triple → Prop :=
 | equ_unif_step : ∀ T T', equ_sys T T' → unif_step T T'
 | fresh_unif_step : ∀ T T', fixpoint_Problem (equ_proj (snd T)) →
                             fresh_sys T T' → unif_step T T' .

Definition leaf (T : Triple) := NF _ unif_step T .
```

Unification paths are derivations with the relation unif_step to a leaf:

```
Definition unif_path (T T' : Triple) := tr_clos _ unif_step T T' ∧ leaf T'.
```

Soundness is specified as the Theorem below, which reads: for any unification problem T that reduces into a problem T' with the relation unif_path, and such that Sl is a solution of T', Sl is also a solution of T.

```
Theorem c_unif_path_soundness : ∀ T T' Sl,
        valid_triple T →unif_path T T' → sol_c Sl T' → sol_c Sl T.
```

The formalisation of soundness is given in a theory that consists of 902 lines or 35KB. This theory also includes lemmas that characterise successful leaves and their solutions. The theorem uses three auxiliary lemmas, also proved by induction. A lemma expresses preservation of the set of solutions of unification problems under reduction by the relation $\Rightarrow_{\#}$:

```
Lemma fresh_sys_compl : ∀ T T' Sl, fresh_sys T T' → (sol_c Sl T ↔ sol_c Sl T') .
```

Another lemma, the longer one, states that the solutions of a unification problem obtained from a given problem through application of the relation $\Rightarrow_{\approx}$ are solutions of the given problem:

```
Lemma equ_sol_preserv : ∀ T T' Sl, valid_triple T →
                          equ_sys T T' → sol_c Sl T' → sol_c Sl T .
```

Finally, the last auxiliary lemma applied to prove soundness states that solutions are preserved in each unification step:

```
Lemma unif_step_preserv : ∀ T T' Sl,
        valid_triple T → unif_step T T' → sol_c Sl T' → sol_c Sl T.
```

Since except ($\approx_{\{\alpha,\mathbf{C}\}}$ **C**) unification rules are invertible, the formalisation of the proof of completeness is shorter, consisiting only of 351 lines or 13KB. The additional element to be considered is the nondeterminism of ($\approx_{\{\alpha,\mathbf{C}\}}$ **C**), indeed implemented as two rules. The key theorem states that *Sl* is a solution for *T* iff there exists a unification path form *T* to some *T'* with solution *Sl* .

```
Theorem unif_path_compl : ∀ T Sl,
        valid_triple T → (sol_c Sl T ↔ ∃ T', unif_path T T' ∧ sol_c Sl T').
```

Excluding formalisation of nominal terms and E-equivalence, subject of [3], the whole theory consists of theories Completeness, Soundness, Termination, C-Unif, Substs, Problems and C-Equiv, which consist of 5474 lines or 204KB.

6 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, which may contain FP equations, represent a complete set of unifiers. We have shown that nominal C-unification is infinitary and NP-complete. An OCaml implementation of the simplification phase has been developed, which outputs derivation trees. Extensions to deal with different equational theories will be considered in 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 the 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.* *ENTCS*, 332:21–38, 2017.
4. M. Ayala-Rincón, W. Carvalho-Segundo, M. Fernández, and D. Nantes-Sobrinho. *On Solving Nominal Fixpoint Equations.* In *Proc. of the 11th Int. Symp. on Frontiers of Combining Systems (FroCoS)*, volume 10483 of *LNCS*, pages 209–226. Springer, 2017.
5. M. Ayala-Rincón, M. Fernández, and D. Nantes-Sobrinho. *Nominal Narrowing.* In *Proc. of the 1st Int. Conf. on Formal Structures for Computation and Deduction (FSCD)*, volume 52 of *LIPIcs*, pages 11:1–11:17, 2016.
6. M. Ayala-Rincón, M. Fernández, and A. C. Rocha-oliveira. *Completeness in PVS of a Nominal Unification Algorithm.* *ENTCS*, 323:57–74, 2016.
7. F. Baader and T. Nipkow. *Term Rewriting and* All That. Cambridge UP, 1998.
8. T. Braibant and D. Pous. *Tactics for Reasoning Modulo AC in Coq.* In *In Proc. of the 1st. Int. Conf. on Certified Programs and Proofs (CPP)*, volume 7086 of *LNCS*, pages 167–182. Springer, 2011.
9. C. F. Calvès. *Complexity and implementation of nominal algorithms.* PhD Thesis, King's College London, 2010.
10. C. F. Calvès and M. Fernández. *Implementing Nominal Unification.* *ENTCS*, 176(1):25–37, 2007.

11. C. F. Calvès and M. Fernández. *The First-order Nominal Link*. In *Proc. of the 20th Int. Symp. Logic-based Program Synthesis and Transformation (LOPSTR)*, volume 6564 of *LNCS*, pages 234–248. Springer, 2011.
12. J. Cheney. *αProlog Users Guide & Language Reference Version 0.3 DRAFT*, 2003.
13. J. Cheney. *Equivariant unification. J. of Autom. Reasoning*, 45(3):267–300, 2010.
14. R. A. Clouston and A. M. Pitts. *Nominal Equational Logic. ENTCS*, 172:223–257, 2007.
15. E. Contejean. *A Certified AC Matching Algorithm*. In *Proc. of the 15th Int. Conf. on Rewriting Techniques and Applications, (RTA)*, volume 3091 of *LNCS*, pages 70–84. Springer, 2004.
16. M. Fernández and M. J. Gabbay. *Nominal Rewriting. Information and Computation*, 205(6):917–965, 2007.
17. M. Fernández and M. J. Gabbay. *Closed nominal rewriting and efficiently computable nominal algebra equality*. In *Proc. of the 5th Int. Work. on Logical Frameworks and Meta-languages: Theory and Practice (LFMTP)*, volume 34 of *EPTCS*, pages 37–51, 2010.
18. M. Fernández, M. J. Gabbay, and I. Mackie. *Nominal Rewriting Systems*. In *Proc. of the 6th Int. Conf. on Principles and Practice of Declarative Programming (PPDP)*, pages 108–119. ACM Press, 2004.
19. M. J. Gabbay and A. Mathijssen. *Nominal (Universal) Algebra: Equational Logic with Names and Binding. J. of Logic and Computation*, 19(6):1455–1508, 2009.
20. 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.
21. D. Kapur and P. Narendran. *Matching, Unification and Complexity. SIGSAM Bulletin*, 21(4):6–9, 1987.
22. R. Kumar and M. Norrish. *(Nominal) Unification by Recursive Descent with Triangular Substitutions*. In *Proc. of Interactive Theorem Proving, 1st Int. Conf. (ITP)*, volume 6172 of *LNCS*, pages 51–66. Springer, 2010.
23. T. Kutsia, J. Levy, M. Schmidt-Schauß, and M. Villaret. *Nominal Unification of Higher Order Expressions with Recursive Let*. In *Proc. of the 26th Int. Sym. on Logic-Based Program Synthesis and Transformation (LOPSTR)*, volume 10184 of *LNCS*, pages 328–344. Springer, 2016.
24. J. Levy and M. Villaret. *An Efficient Nominal Unification Algorithm*. In *Proc. of the 21st Int. Conf. on Rewriting Techniques and Applications (RTA)*, volume 6 of *LIPIcs*, pages 209–226, 2010.
25. T. Nipkow. *Equational Reasoning in Isabelle. Science of Computer Programming*, 12(2):123–149, 1989.
26. A. M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge UP, 2013.
27. Sagan, B. E. *The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions*. Springer, 2nd edition, 2001.
28. J. H. Siekmann. *Unification of Commutative Terms*. In *Proc. of the Int. Symposium on Symbolic and Algebraic Manipulation*, volume 72 of *LNCS*, pages 22–29. Springer, 1979.
29. C. Urban. *Nominal Unification Revisited*. In *Proc. of the 24th Int. Work. on Unification (UNIF)*, volume 42 of *EPTCS*, pages 1–11, 2010.
30. 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

All the proofs in section 3 have been formalised in Coq.

Lemma 2 (Inversion). *The inference rules of* $\approx_{\{\alpha,C\}}$ *are* invertible*:*

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* $ds(\pi,\pi')\#X \subseteq \nabla$

Proof. (Sketch) The proof is by case analysis in the derivation rules of the relation $\_ \vdash \_ \approx_{\{\alpha,C\}} \_$. It is necessary to evaluate the possible scenarios where the hypothesis can be considered valid. For example, in 2., $\nabla \vdash f_k^C\, \langle s_0, s_1\rangle \approx_{\{\alpha,C\}} f_k^C\, \langle t_0, t_1\rangle$ is valid only if we have either $\nabla \vdash s_0 \approx_{\{\alpha,C\}} t_0$ and $\nabla \vdash s_1 \approx_{\{\alpha,C\}} t_1$, or $\nabla \vdash s_0 \approx_{\{\alpha,C\}} t_1$ and $\nabla \vdash s_1 \approx_{\{\alpha,C\}} t_0$. In the formalisation this proof strategy is implemented in a very simple way by the use of the tactic `inverts` applied to the hypothesis.

Lemma 7 (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 σ 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.

Lemma 9 (Characterisation of leaves of $\mathcal{T}_{\langle\Delta,P\rangle}$). *Let* $\langle \Delta, P\rangle$ *be a unification problem. If* $\mathcal{P}' = \langle \nabla, \sigma', P'\rangle$ *is the label of a leaf in* $\mathcal{T}_{\langle\Delta,P\rangle}$*, then* P' *can be partitioned as follows:* $P' = P'' \cup P_\perp$*, where* P'' *is the set of all FP 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 FP equation in $P_\perp$, say $s \approx_? t$, then by definition of $\mathcal{T}_{\langle\Delta,P\rangle}$, $\mathcal{P}'$ 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 $\langle \nabla', \delta \rangle$ such that $\nabla' \vdash s\delta \approx_{\{\alpha,C\}} t\delta$, 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 $\mathcal{Q} = \langle \Delta, \sigma, Q \rangle \in \mathcal{P}_{\Rightarrow_\approx}$ and $\mathcal{P}'$ is a $\Rightarrow_\#$-nf of $\mathcal{Q}$. The set Q can be split into sets of freshness constraints, $Q_\perp$, and FP 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. 8), and there is no $\langle \nabla', \delta \rangle$ such that $\nabla' \vdash a \,\#\, \bar{a}\delta$, that is $\nabla' \vdash a \,\#\, \bar{a}$, one concludes that also in this case $\mathcal{U}_C(\mathcal{P}') = \emptyset$.

Lemma A1 (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 4) 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 λ 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 2) $\nabla' \vdash s\sigma' \approx_{\{\alpha,C\}} t\sigma'$ 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. 4 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. 4. 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 1, one has $((a\,b)\cdot t)\sigma' = (a\,b)\cdot(t\sigma')$. Hence, by application of the α-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. 2) can be written as $\nabla' \vdash ([a]s)\sigma' \approx_{\{\alpha,C\}} ([b]t)\sigma'$. Thus, $\mathcal{U}_C(\mathcal{P}') \subseteq \mathcal{U}_C(\mathcal{P})$.

Notice that in this case $\mathcal{U}_C(\mathcal{P}) \subseteq \mathcal{U}_C(\mathcal{P}')$ too; indeed, if $\nabla' \vdash ([a]s)\sigma' \approx_{\{\alpha,C\}} ([b]t)\sigma'$, by Def. 2, reverse application of the α-equivalence rule ($\approx_\alpha$ [**ab**]) (Lemma 2) and Lemma 1, 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. 4. 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 λ. 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 7, $dom(\sigma)$ 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. 6) one concludes that $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$.

The first condition in Def. 4 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 $u = f\,v$ 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 3) and the fact that $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$ (proved in the analysis of the third condition of Def. 4) 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. 4 uses the hypothesis that there exists a λ 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. 4 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. 2), this holds if and only if $\nabla' \vdash \pi \oplus (\pi')^{-1} \cdot (X\sigma') \approx_{\{\alpha,C\}} X\sigma'$, and by equivariance (Lemma 5) and the definition of action of substitutions (Def. 2), 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 A2 (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 $(\#_?\mathbf{var})$. For rules $(\#_?\langle\rangle)$, $(\#_?\mathbf{a}\bar{\mathbf{b}})$, $(\#_?\mathbf{app})$, $(\#_?\mathbf{a}[\mathbf{a}])$ $(\#_?\mathbf{a}[\mathbf{b}])$ and $(\langle\#\rangle\mathbf{pair})$ 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. 4, 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. 4 hold for $\mathcal{P}$. For the second condition in Def. 4, 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. 2), 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 $(\#_?\mathbf{var})$:

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

On the one hand, if $\langle\nabla', \sigma'\rangle \in \mathcal{U}_C(\mathcal{P})$, the second, third and fourth conditions in Def. 4 hold trivially for $\mathcal{P}'$. To prove the first condition for $\mathcal{P}'$, by the second condition in Def. 4 for $\mathcal{P}$ one has that $\nabla' \vdash a \,\#\, \pi.X\sigma'$, which, by nominal properties and substitution action (Def. 2), implies that $\nabla' \vdash \pi^{-1} \cdot a \,\#\, X\sigma'$. Since by hypothesis $\nabla' \vdash \nabla\sigma'$ (first condition of Def. 4 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. 4 hold trivially for $\mathcal{P}$. For proving the second condition for $\mathcal{P}$, by the first condition in Def. 4 one has that $\nabla' \vdash (\pi^{-1} \cdot a) \,\#\, X\sigma'$, which again by nominal properties and Def. 2 implies that $\nabla' \vdash a \,\#\, (\pi.X)\sigma'$. Thus, by the last and the second condition in Def. 4 for $\mathcal{P}'$, one obtains the second condition for $\mathcal{P}$. Therefore, $\mathcal{U}_C(\mathcal{P}') \subseteq \mathcal{U}_C(\mathcal{P})$.

Theorem 1 (Soundness of $\mathcal{T}_{\langle\Delta,P\rangle}$). *$\mathcal{T}_{\langle\Delta,P\rangle}$ is correct, i.e., if $\mathcal{P}' = \langle\nabla, \sigma, P'\rangle$ is the label of a leaf in $\mathcal{T}_{\langle\Delta,P\rangle}$, then*

1. *$\mathcal{U}_C(\mathcal{P}') \subseteq \mathcal{U}_C(\langle\Delta, id, P\rangle)$, and*
2. *if P' contains non FP 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 A1 and A2. The second is a direct application of Lemma 9.

Theorem 2 (Completeness of $\mathcal{T}_{\langle\Delta,P\rangle}$). *Let $\mathcal{T}_{\langle\Delta,P\rangle}$ be a derivation tree for the unification problem $\langle\Delta,P\rangle$, where $\mathcal{P}=\langle\Delta,id,P\rangle$. Then*

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

Proof. From soundness (Theorem 1), $\mathcal{U}_C(\mathcal{P}) \supseteq \bigcup_{\mathcal{Q}\in ST(\mathcal{T}_{\langle\Delta,P\rangle})} \mathcal{U}_C(\mathcal{Q})$. The other inclusion is proved by induction on the size of subtrees of $\mathcal{T}_{\langle\Delta,P\rangle}$. This is done verifying that in the preservation lemmas for $\Rightarrow_{\approx}$ and $\Rightarrow_{\#}$ (Lemmas A1 and A2) 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 $\mathcal{T}_{\langle\Delta,P\rangle}$, one has two sibling nodes labelled with triples $\mathcal{Q}_1$ and $\mathcal{Q}_2$ such that

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

If $\langle\nabla',\sigma'\rangle\in\mathcal{U}_C(\mathcal{Q})$ then it satisfies the four conditions in Def. 4 for $\mathcal{Q}$. It can be easily checked that $\langle\nabla',\sigma'\rangle$ 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 $\nabla'\vdash f^C\langle s_0,s_1\rangle\sigma' \approx_{\{\alpha,C\}} f^C\langle t_0,t_1\rangle\sigma'$. From the last, by substitution action one has that $\nabla'\vdash f^C\langle s_0\sigma',s_1\sigma'\rangle \approx_{\{\alpha,C\}} f^C\langle t_0\sigma',t_1\sigma'\rangle$. Then, by Lemma 2 either $\nabla'\vdash s_0\sigma'\approx_{\{\alpha,C\}} t_0\sigma'$ and $\nabla'\vdash s_1\sigma'\approx_{\{\alpha,C\}} t_1\sigma'$, or $\nabla'\vdash s_0\sigma'\approx_{\{\alpha,C\}} t_1\sigma'$ and $\nabla'\vdash s_1\sigma'\approx_{\{\alpha,C\}} t_0\sigma'$. Thus, the third condition holds for $\mathcal{Q}_1$ or for $\mathcal{Q}_2$, and it can be concluded that $\langle\nabla',\sigma'\rangle\in\mathcal{U}_C(\mathcal{Q})$ if and only if $\langle\nabla',\sigma'\rangle\in\mathcal{U}_C(\mathcal{Q}_1)$ or $\langle\nabla',\sigma'\rangle\in\mathcal{U}_C(\mathcal{Q}_2)$. 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 $\mathcal{T}_{\langle\Delta,P\rangle}$, one has a sibling node labelled with a triple $\mathcal{Q}'$ such that

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

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

Suppose that $\langle\nabla',\sigma'\rangle\in\mathcal{U}_C(\mathcal{Q})$. We will check that $\langle\nabla',\sigma'\rangle$ satisfies the four conditions in Def. 4 for $\mathcal{Q}'$.

− The **first condition**, that is $\nabla'\vdash\nabla\sigma'$, 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 $\nabla'\vdash a\#(u\{X/\pi^{-1}\cdot t\})\sigma'$ and
2. for all $Y\in dom(\sigma'')$ such that $a\#Y\in\nabla$, $\nabla'\vdash a\#Y\sigma''\sigma'$.

For the first subcase we use induction in u similarly to the case of rule ($\approx_?$ **inst**) in Lemma A1, being the interesting case, when $u=\phi.X$, for which the hypotheses of interest are that $\nabla'\vdash a\#\phi.X\sigma'$ and $\nabla'\vdash\pi.X\sigma'\approx_{\{\alpha,C\}} t\sigma'$, by the second and third conditions in Def. 4 for $\mathcal{Q}$, respectively. From the first hypothesis, by nominal properties of the freshness relation one has that $\nabla' \vdash \phi^{-1} \cdot a\#X\sigma'$; by nominal properties of the freshness relation, also it follows that $\nabla' \vdash \pi \cdot \phi^{-1} \cdot a\#\pi \cdot X\sigma'$; by the second hypothesis and freshness preservation (Lemma 3) one obtains $\nabla' \vdash \pi \cdot \phi^{-1} \cdot a\#t\sigma'$; again by nominal properties the last gives $\nabla' \vdash a\#\phi \cdot \pi^{-1} \cdot t\sigma'$, which by substitution application gives finally $\nabla' \vdash a\#(\phi.X\{X/\pi^{-1} \cdot t\})\sigma'$.

For the second subcase above, observe initially that for a $Y \in dom(\sigma'')$ such that $a\#Y \in \nabla$, $\nabla' \vdash a\#Y\sigma''\sigma'$ equals $\nabla' \vdash a\#Y\sigma\{X/\pi^{-1} \cdot t\}\sigma'$. 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. 4 for $\mathcal{Q}$: namely, it contradicts $\nabla' \vdash a\#Y\sigma'$ that is $\nabla' \vdash a\#\bar{a}$ since $\nabla' \vdash \sigma\lambda \approx \sigma'$ (fourth condition in Def. 4 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 $\nabla' \vdash a\#Y\sigma'$, that by the fourth condition gives $\nabla' \vdash a\#Y\sigma\lambda$; thus, $\nabla' \vdash a\#\phi.Z\lambda$ from which one has (since we are dealing with valid triples, Lemma 7) $\nabla' \vdash a\#\phi.Z\sigma\lambda$ and then $\nabla' \vdash a\#\phi.Z\{X/\pi^{-1} \cdot t\}\sigma\lambda$, and finally, $\nabla' \vdash a\#Y\sigma\{X/\pi^{-1} \cdot t\}\sigma\lambda$ that is $\nabla' \vdash a\#Y\sigma''\sigma'$. Otherwise, if $Z = X$, that is $Y\sigma = \phi.X$, by the first and fourth conditions for $\mathcal{Q}$ one has that $\nabla' \vdash a\#Y\sigma'$ and then that $\nabla' \vdash a\#Y\sigma\lambda$; thus, $\nabla' \vdash a\#\phi.X\lambda$. On the other side, by the third and fourth conditions for $\mathcal{Q}$, one has that $\nabla' \vdash \pi.X\sigma\lambda \approx_{\{\alpha,C\}} t\sigma\lambda$, which by equivariance (Lemma 5) and since $\mathcal{Q}$ is a valid triple gives $\nabla' \vdash X\lambda \approx_{\{\alpha,C\}} \pi^{-1} \cdot t\lambda$. Again, by equivariance and substitution and permutation properties one obtains $\nabla' \vdash \phi.X\lambda \approx_{\{\alpha,C\}} \phi \cdot \pi^{-1} \cdot t\lambda$. From the last and $\nabla' \vdash a\#\phi.X\lambda$, by freshness preservation (Lemma 3) one obtains $\nabla' \vdash a\#\phi \cdot \pi^{-1} \cdot t\lambda$ and, by freshness properties this gives $\nabla' \vdash \pi \cdot \phi^{-1} \cdot a\#t\lambda$. Then, $\nabla' \vdash \pi \cdot \phi^{-1} \cdot a\#\pi \cdot X\{X/\pi^{-1} \cdot t\}\lambda$ which, since $\mathcal{Q}$ is a valid triple, gives $\nabla' \vdash \pi \cdot \phi^{-1} \cdot a\#\pi \cdot X\{X/\pi^{-1} \cdot t\}\sigma\lambda$ and by permutation and freshness properties $\nabla' \vdash a\#\phi.X\{X/\pi^{-1} \cdot t\}\sigma\lambda$. The last gives the desired property: $\nabla' \vdash a\#Y\sigma\{X/\pi^{-1} \cdot t\}\sigma\lambda$, that is $\nabla' \vdash a\#Y\sigma''\sigma'$.

– Now, we consider the **third condition** in Def. 4 for $\mathcal{Q}'$. It should be proved that for any equation $u \approx_? v$ in Q, except $\pi.X \approx_? t$, $\nabla' \vdash u\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} v\{X/\pi^{-1} \cdot t\}\sigma'$. This is done by induction in u and v.

- Case $u = \langle\rangle$. Since $\nabla' \vdash \langle\rangle\sigma' \approx_{\{\alpha,C\}} v\sigma'$ (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 $X \neq Y$, $\nabla' \vdash \langle\rangle\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1}\cdot t\}\sigma'$. If $X = Y$, we have the following chain using nominal properties and the hypothesis that $\nabla' \vdash \langle\rangle\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$: $\nabla' \vdash \pi \cdot \phi^{-1} \cdot \langle\rangle\sigma' \approx_{\{\alpha,C\}} \pi.X\sigma'$ implies $\nabla' \vdash \pi \cdot \phi^{-1} \cdot \langle\rangle\sigma' \approx_{\{\alpha,C\}} t\sigma'$, since $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$ (by the third property for $\mathcal{Q}$), implies $\nabla' \vdash \phi^{-1}\cdot\langle\rangle\sigma' \approx_{\{\alpha,C\}} \pi^{-1}\cdot t\sigma'$ iff $\nabla' \vdash \phi^{-1}\cdot\langle\rangle\sigma' \approx_{\{\alpha,C\}} X\{X/\pi^{-1}\cdot t\}\sigma'$, implies $\nabla' \vdash \langle\rangle\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1}\cdot t\}\sigma'$ iff $\nabla' \vdash \langle\rangle\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1}\cdot t\}\sigma'$.
- Case $u = \bar{a}$ Since $\nabla' \vdash \bar{a}\sigma' \approx_{\{\alpha,C\}} v\sigma'$ (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 $X \neq Y$, $\nabla' \vdash \bar{a}\{X/\pi^{-1}\cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1}\cdot$

$t\}\sigma'$ since $\nabla' \vdash \bar{a}\sigma' \approx_{\{\alpha,C\}} \phi.Y\sigma'$ holds. If $X = Y$, we have the following chain using nominal properties and the hypothesis that $\nabla' \vdash \bar{a}\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$: $\nabla' \vdash \pi \cdot \phi^{-1} \cdot \bar{a}\sigma' \approx_{\{\alpha,C\}} \pi.X\sigma'$ implies $\nabla' \vdash \pi \cdot \phi^{-1} \cdot \bar{a}\sigma' \approx_{\{\alpha,C\}} t\sigma'$, since $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$ (by the third property for $\mathcal{Q}$), implies $\nabla' \vdash \phi^{-1} \cdot \bar{a}\sigma' \approx_{\{\alpha,C\}} \pi^{-1} \cdot t\sigma'$ iff $\nabla' \vdash \phi^{-1} \cdot \bar{a}\sigma' \approx_{\{\alpha,C\}} X\{X/\pi^{-1} \cdot t\}\sigma'$, implies $\nabla' \vdash \bar{a}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1} \cdot t\}\sigma'$ iff $\nabla' \vdash \bar{a}\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1} \cdot t\}\sigma'$.

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

  For the subcase in which $v = \phi.Y$, notice that $Y\sigma' = [b]w$. If $X = Y$, the hypothesis that $\nabla' \vdash [a]s\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$ implies that $X \notin Var(s)$, and since also $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$, we obtain $\nabla' \vdash [a]s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1} \cdot t\}\sigma'$. Otherwise, let Y' be a new variable and extend $\mathcal{Q}$ with the equation $Y \approx_? [b]Y'$, ∇' with the constraints $c\#Y'$ for all $c\#Y \in \nabla'$ except $b\#Y'$, and σ' with the bind $\{Y'/w\}$, so that $Y'\sigma' = w$. Thus, $\nabla' \vdash [a]s\sigma' \approx_{\{\alpha,C\}} [\phi \cdot b]\phi.Y'\sigma'$. If $\phi \cdot b = a$, by rule ($\approx_\alpha$ [**aa**]) we have $\nabla' \vdash s\sigma' \approx_{\{\alpha,C\}} \phi.Y'\sigma'$ and by i.h. $\nabla' \vdash s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi \cdot Y'\{X/\pi^{-1} \cdot t\}\sigma'$, which again by rule ($\approx_\alpha$ [**aa**]) gives $\nabla' \vdash [a]s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1} \cdot t\}\sigma'$. If $\phi \cdot b = c$, by rule ($\approx_\alpha$ [**ab**]) we have $\nabla' \vdash s\sigma' \approx_{\{\alpha,C\}} (a\,c)\phi.Y'\sigma'$ and $\nabla' \vdash a\#\phi.Y'\sigma'$. Thus, $\nabla' \vdash a\#\phi.Y'\{X/\pi^{-1} \cdot t\}\sigma'$ and since by i.h. $\nabla' \vdash s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} (a\,c)\phi.Y'\{X/\pi^{-1} \cdot t\}\sigma'$, by application of rule ($\approx_\alpha$ [**ab**]) we can conclude that $\nabla' \vdash [a]s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} [\phi \cdot b]\phi.Y'\{X/\pi^{-1} \cdot t\}\sigma'$ or equivalently that $\nabla' \vdash [a]s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1} \cdot t\}\sigma'$.
- 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 $\nabla' \vdash \langle s, w\rangle\sigma' \approx_{\{\alpha,C\}} \langle v_1, v_2\rangle\sigma'$, and by rule ($\approx_\alpha$ **pair**), this holds iff $\nabla' \vdash s\sigma' \approx_{\{\alpha,C\}} v_1\sigma'$ and $\nabla' \vdash w\sigma' \approx_{\{\alpha,C\}} v_2\sigma'$; this gives, by i.h., $\nabla' \vdash s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} v_1\{X/\pi^{-1} \cdot t\}\sigma'$ and $\nabla' \vdash w\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} v_2\{X/\pi^{-1} \cdot t\}\sigma'$ and finally, again by rule ($\approx_\alpha$ **pair**), $\nabla' \vdash \langle s, w\rangle\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \langle v_1, v_2\rangle\{X/\pi^{-1} \cdot t\}\sigma'$.

  When $v = \phi.X$, since $\nabla' \vdash \langle s, w\rangle\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$, $X \notin Var\langle s, w\rangle$; thus, since also $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$, one has that $\nabla' \vdash \langle s, w\rangle\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1} \cdot t\}\sigma'$. 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 $Y = \langle Y_1, Y_2\rangle$, ∇' with constraints $a\#Y_i$ for $i = 1, 2$ for all $a\#Y \in \nabla'$, and σ' 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, $\nabla' \vdash s\sigma' \approx_{\{\alpha,C\}} \phi.Y_1\sigma'$ and $\nabla \vdash w\sigma' \approx_{\{\alpha,C\}} \phi.Y_2\sigma'$ and by i.h. $\nabla' \vdash s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y_1\{X/\pi^{-1} \cdot t\}\sigma'$ and $\nabla \vdash w\sigma'\{X/\pi^{-1} \cdot t\} \approx_{\{\alpha,C\}} \phi.Y_2\{X/\pi^{-1} \cdot t\}\sigma'$; thus, again by rule ($\approx_\alpha$ **pair**), one has $\nabla' \vdash \langle s, w\rangle\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi \cdot \langle v_1, v_2\rangle\{X/\pi^{-1} \cdot t\}\sigma'$, that is $\nabla' \vdash \langle s, w\rangle\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1} \cdot t\}\sigma'$.

- 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: $\nabla' fs\sigma' \approx_{\{\alpha,C\}} fw\sigma'$ iff $\nabla' \vdash s\sigma \approx_{\{\alpha,C\}} w\sigma'$, by rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **app**); then, by i.h. $\nabla' \vdash s\{X/\pi^{-1} \cdot t\}\sigma \approx_{\{\alpha,C\}} w\{X/\pi^{-1} \cdot t\}\sigma'$, and, again by rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **app**) one has that $\nabla' \vdash fs\{X/\pi^{-1} \cdot t\}\sigma \approx_{\{\alpha,C\}} fw\{X/\pi^{-1} \cdot t\}\sigma'$.
  For the case in which $v = \phi.X$, notice that $X \notin Var(s)$ since $\nabla \vdash fs\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$; thus, $\nabla \vdash fs\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.X\{X/\pi^{-1} \cdot t\}\sigma'$. 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 $Y \approx_? fY'$, ∇' including the constraint $a\#Y'$ for all $a\#Y \in \nabla'$ and σ' with the bind $\{Y'/w\}$; so $Y'\sigma' = w$. First, $\nabla' \vdash s\sigma' \approx_{\{\alpha,C\}} \phi.Y'\sigma'$ by the hypothesis and application of rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **app**); then, by i.h. one has that $\nabla' \vdash s\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi.Y'\{X/\pi^{-1} \cdot t\}\sigma'$ and by application of the rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **app**) one has that $\nabla' \vdash fs\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} f\phi.Y'\{X/\pi^{-1} \cdot t\}\sigma'$ or equivalently $\nabla' \vdash fs\{X/\pi^{-1} \cdot t\}\sigma \approx_{\{\alpha,C\}} \phi.Y\{X/\pi^{-1} \cdot t\}\sigma'$.
- 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 $\nabla' \vdash u_0\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} v_0\{X/\pi^{-1} \cdot t\}\sigma'$ and $\nabla' \vdash u_1\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} v_1\{X/\pi^{-1} \cdot t\}\sigma'$, or $\nabla' \vdash u_0\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} v_1\{X/\pi^{-1} \cdot t\}\sigma'$ and $\nabla' \vdash u_1\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} v_0\{X/\pi^{-1} \cdot t\}\sigma'$ and, by application of rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **pair**), one obtains $\nabla' \vdash f\langle u_0, u_1\rangle\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} f\langle v_0, v_1\rangle\{X/\pi^{-1} \cdot t\}\sigma'$.
  For the case in which $v = \phi.X$, one has that $X \notin Var(u)$ since $\nabla' \vdash f\langle u_0, u_1\rangle\sigma' \approx_{\{\alpha,C\}} \phi.X\sigma'$. Thus, since $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$, we obtain that $\nabla' \vdash f\langle u_0, u_1\rangle\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \pi.X\{X/\pi^{-1} \cdot t\}\sigma'$. For the case in which $v = \phi.Y$ and $X \neq Y$, let Y_0 and Y_1 be new variables and extend ∇' with $a\#Y_i$ for $i = 0$ and $i = 1$ and all $a\#Y \in \nabla'$, and σ' with the binds $\{Y_0/u_i, Y_2/u_{i+1}\}$ for $i = 0$ or $i = 1$. Then, one has that $\nabla' \vdash f\langle u_0, u_1\rangle\sigma' \approx_{\{\alpha,C\}} f\langle Y_0, Y_1\rangle\sigma'$ which by rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **pair**) and i.h. implies that either $\nabla' \vdash u_0\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} Y_0\{X/\pi^{-1} \cdot t\}\sigma'$ and $\nabla' \vdash u_1\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} Y_1\{X/\pi^{-1} \cdot t\}\sigma'$ or, $\nabla' \vdash u_0\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} Y_1\{X/\pi^{-1} \cdot t\}\sigma'$ and $\nabla' \vdash u_1\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} Y_0\{X/\pi^{-1} \cdot t\}\sigma'$. From the last and by application of rule ($\approx_{\{\alpha,\mathbf{C}\}}$ **pair**) we conclude $\nabla' \vdash f\langle u_0, u_1\rangle\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} f\langle Y_0, Y_1\rangle\{X/\pi^{-1} \cdot t\}\sigma'$.
- 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 $\nabla' \vdash \phi.X\sigma \approx_{\{\alpha,C\}}$

$\phi'.X\sigma'$, since $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$, we conclude that $\nabla' \vdash \phi.X\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi'.X\{X/\pi^{-1} \cdot t\}\sigma'$. Case $Y \neq X = Y'$, we also use the hypotheses that $\nabla' \vdash \phi.X\sigma \approx_{\{\alpha,C\}} \phi'.X\sigma'$ and $\nabla' \vdash \pi.X\sigma' \approx_{\{\alpha,C\}} t\sigma'$ to conclude that $\nabla' \vdash \phi.Y\{X/\pi^{-1} \cdot t\}\sigma' \approx_{\{\alpha,C\}} \phi'.X\{X/\pi^{-1} \cdot t\}\sigma'$; the case $Y = X \neq Y'$ follows analogously. The case $Y \neq X \neq Y'$ requires only the hypothesis that $\nabla' \vdash \phi.X\sigma \approx_{\{\alpha,C\}} \phi'.X\sigma'$.

– To prove the **fourth condition** in Def. 4 for $\mathcal{Q}'$, select $\lambda' = \lambda \setminus \{X/X\lambda\}$. We have to prove that for any variable Y, $\nabla' \vdash Y\sigma''\lambda' \approx_{\{\alpha,C\}} Y\sigma'$ assuming that $\nabla' \vdash Y\sigma\lambda \approx_{\{\alpha,C\}} Y\sigma'$.

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

B Proofs of Section 4

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

Definition B1 (Pseudo-cycle C equivalence) *Let κ_1 and κ_2 be two pseudo-cycles w.r.t. the pseudo-cycle κ. They are said to be* equivalent modulo commutativity of $*$*, denoted by $\kappa_1 \approx_{(C,*)} \kappa_2$, if each element A of κ_1 is equivalent modulo commutativity of $*$ to one element B of κ_2, i.e., $A \approx_{(C,*)} B$ and they have the same successor, that is, $\kappa \cdot A \approx_{(C,*)} \kappa \cdot B$.*

Example B1 $((A * B)\ (C * D)\ (E * F)) \approx_C ((F * E)\ (A * B)\ (D * C))$

$[\kappa]_{(C,*)}$ denotes the equivalence class modulo commutativity of $*$ of the pseudo-cycle κ, 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 κ by $[\kappa]_C$. So, pseudo-cycle equivalence modulo C can be determined by a single element:

Lemma B1 *If κ_1 and κ_2 are two pseudo-cycles w.r.t. the cycle κ, 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 8. The elements of pseudo-cycles can be represented by the coefficients of κ. Thus, if one chooses an element A_0 of κ, 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 κ represented by its coefficients:

Definition B2 *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 $a_{ij} := \overline{j-1} * \overline{i+j-1}$.*

$$\mathcal{M}_{(k-1)\times k} = \begin{bmatrix} \overline{0} * \overline{1} & \overline{1} * \overline{2} & \cdots & \overline{k-1} * \overline{0} \\ \overline{0} * \overline{2} & \overline{1} * \overline{3} & \cdots & \overline{k-1} * \overline{1} \\ & & & \\ \ddots & \ddots & \ddots & \ddots \\ & & & \\ \overline{0} * \overline{k-2} & \overline{1} * \overline{k-1} & \cdots & \overline{k-1} * \overline{k-3} \\ \overline{0} * \overline{k-1} & \overline{1} * \overline{0} & \cdots & \overline{k-1} * \overline{k-2} \end{bmatrix}_{(k-1)\times k}$$

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 κ*.*

Remark 9. We want to establish a relationship between the rows of the matrix $\mathcal{M}_{(k-1)\times k}$ related to κ and the first instance pseudo-cycles of κ: for each i, $1 \leq i \leq k-1$, we map the i-th row $[a_{i1}\ a_{i2} \dots a_{ik}]$ of $\mathcal{M}_{(k-1)\times k}$ to the k-cycle: $\kappa_i = (\overline{0} * \overline{i} \quad \overline{1} * \overline{i+1} \quad \dots \quad \overline{k-1} * \overline{i-1})$.

Example B2 *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 κ: the second row doesn't, since it contradicts condition 2.b) of the Definition 12; however, it contains two first instance pseudo-cycles of κ, 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 κ.

Example B3 *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 $\kappa = (\overline{0}\quad \overline{1}\quad \overline{2}\quad \overline{3})$ of κ 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 $((\kappa^0 \cdot \overline{a})\quad (\kappa^1 \cdot \overline{a})\quad (\kappa^2 \cdot \overline{a})\quad (\kappa^3 \cdot \overline{a})) = (\overline{a}\ \overline{b}\ \overline{c}\ \overline{d})$.*
- *if $A_0 = \overline{b}$, then $(\overline{0}\,\overline{1}\,\overline{2}\,\overline{3})$ corresponds to $((\kappa^0 \cdot \overline{b})\quad (\kappa^1 \cdot \overline{b})\quad (\kappa^2 \cdot \overline{b})\quad (\kappa^3 \cdot \overline{b})) = (\overline{b}\ \overline{c}\ \overline{d}\ \overline{a})$.*

and so on.

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

- *if $A_0 = (a*c)$ then $(\overline{0}\ \overline{1})$ corresponds to the κ_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 B2 *Let $\mathcal{M} = (a_{ij})_{k-1\times k}$ be a first-instance pseudo-cycle matrix for a pseudo-cycle κ. 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 κ, both with length $\frac{k}{2}$.

Proof. The proof of all items follows by algebraic manipulation, using the commutativity of $*$ and Definition B2:

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

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

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

4. (a) By Definition B2, 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 B1 and the mapping from the rows of $M_{(k-1)\times k}$ to cycles.

Example B4 *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 κ, 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 ρ:*

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

  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 κ, if there exist two elements equivalent modulo C, in a row n, then k is even.

Lemma B3 *Let $\mathcal{M} = (a_{ij})_{k-1\times k}$ be a first-instance pseudo-cycle matrix for a pseudo-cycle κ . 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 κ with odd length, is also a first instance pseudo-cycle of κ.

Theorem B1 *Let κ 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 κ, 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 9, to obtain the candidate a pseudo-cycle of $\kappa : \kappa_i = (\overline{0} * \overline{i} \quad \overline{1} * \overline{i+1} \quad \dots \overline{k-1} * \overline{i-1})$. whose elements clearly satisfy the conditions **b.** and **c.** of Definition 12. Also, since k is odd, it follows from Lemma B2, that the elements of κ_i satisfy condition **a.** of the Definition 12.

Lemma B4 *Let $\mathcal{M}_{(k-1)\times k}$ be first instance matrix of the pseudo-cycle κ, with $k = 2n+1$ for some positive integer n. If κ_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 B2, $a_{(k-i)(i+j)} \approx_C a_{ij}$, for each $1 \leq i \leq k-1$, therefore, by Lemma B1, $\kappa_i \approx_C \kappa_{k-i}$.

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

Theorem B2 *κ' is a first-instance pseudo-cycle of κ 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 κ.*

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 B1, κ'' is a pseudo-cycle in a row of $\mathcal{M}_{(k-1)\times k}$, with k elements. Besides, by Lemma B1, $\kappa' \approx_C \kappa''$. 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 B2, either κ'' is equivalent to a row i for $1 \leq i \leq \lfloor \frac{k-1}{2} \rfloor$, and therefore, κ'' is equivalent to a first instance pseudo-cycle of κ with k elements, or κ'' can be split in two pseudo-cycles κ''_1 and κ''_2 of length $\frac{k}{2}$, both containing an element equivalent to A_0, by Lemma B1, $\kappa' \approx_C \kappa''_i$ $(i = 1, 2)$, and therefore, κ' is in a row of $M_{(k-1)\times k}$.

Theorem B3 *Let κ 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 κ 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 κ 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 B2, it follows that rows 1 to $\lfloor \frac{k-1}{2} \rfloor$ of $\mathcal{M}$ are first instance pseudo-cycles of κ 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 B2 and B3, the n-th row of $\mathcal{M}$ contains two equivalent first instance pseudo-cycles of κ both with $\frac{k}{2}$ elements.

2. Suppose $k = 2n + 1$, for some non-negative integer n. By Theorem B1 the $k - 1$ rows of $\mathcal{M}$ are first instance pseudo-cycles of κ with length k. From Lemma B2, 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 3. *A pseudo-cycle κ contains a unitary pseudo-cycle iff $|\kappa|$ is power of two.*

Proof. Let κ be of the form $\kappa = (a_0 \quad a_1 \quad \ldots \quad a_{k-1})$.
($\Leftarrow$) 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. κ and $*$ is

$$\mathcal{M}_{12\times} = \begin{bmatrix} \overline{0} * \overline{1} \; \overline{1} * \overline{0} \end{bmatrix}$$

  Notice that $\overline{0} * \overline{1} \approx_C \overline{1} * \overline{0}$, and this single row of $\mathcal{M}_{12\times}$ 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 κ_l be a pseudo-cycle of length $l = 2^{n+1} = 2.(2^n)$. By Theorem B3, κ_l has a first instance pseudo-cycle of length $\frac{l}{2} = 2^n$, and by IH, the result follows.

($\Rightarrow$) Let κ_1 and κ_2 be pseudo-cycles w.r.t. a pseudo-cycle κ such that κ_2 a first-instance pseudo-cycle of κ_1. By Lemma B2 , $|\kappa_2| < |\kappa_1|$ 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 κ with k elements.

- If $k = 1 = 2^0$, the result follows.
- Suppose that $k > 1$. Then, there exists a pseudo-cycle κ_p regarding to κ, with $|\kappa_p| = 1$ only if one has a chain of pseudo-cycles $\kappa, \kappa_1, \cdots, \kappa_{p-1}, \kappa_p$, where κ_1 is a first-instance pseudo-cycle of κ, and κ_{i+1} is a first-instance pseudo-cycle of κ_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.

Theorem 4. *Let $\mathcal{P} = \langle \emptyset, \{\pi.X \approx_? X\} \rangle$ be a fixed point problem. $\mathcal{P}$ has a combinatory solution iff there exists a unitary pseudo-cycle κ w.r.t. π.*

Proof. ($\Leftarrow$) Suppose that π 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 $\mathcal{P}$.

($\Rightarrow$) Suppose that π does not have a unitary pseudo-cycle, then by Theorem 3, every pseudo-cycle κ w.r.t. π has length $k = 2^n(2r+1)$, for some positive integer r.

Suppose, by contradiction, that there is a combinatory solution for $\mathcal{P}$, say $\{X/t\}$. If any atom from $dom(\kappa)$ is in t, then all atoms of κ 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 κ 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.

C Examples generated with the OCaml implementation

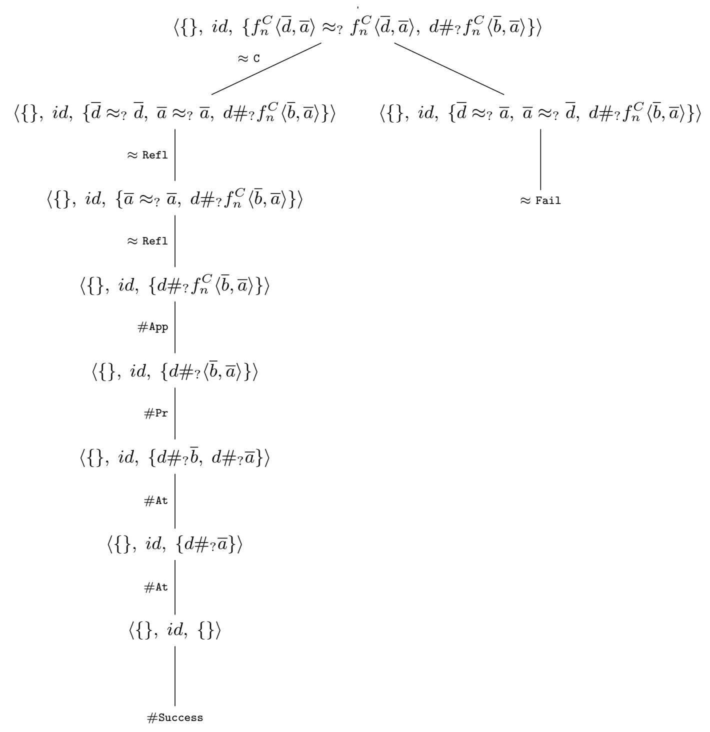
$\langle\{\},\ id,\ \{f_n^C\langle\overline{d},\overline{a}\rangle \approx_? f_n^C\langle\overline{d},\overline{a}\rangle,\ d\#_?f_n^C\langle\overline{b},\overline{a}\rangle\}\rangle$
≈ C
$\langle\{\},\ id,\ \{\overline{d} \approx_? \overline{d},\ \overline{a} \approx_? \overline{a},\ d\#_?f_n^C\langle\overline{b},\overline{a}\rangle\}\rangle$
$\langle\{\},\ id,\ \{\overline{d} \approx_? \overline{a},\ \overline{a} \approx_? \overline{d},\ d\#_?f_n^C\langle\overline{b},\overline{a}\rangle\}\rangle$
≈ Refl
≈ Fail
$\langle\{\},\ id,\ \{\overline{a} \approx_? \overline{a},\ d\#_?f_n^C\langle\overline{b},\overline{a}\rangle\}\rangle$
≈ Refl
$\langle\{\},\ id,\ \{d\#_?f_n^C\langle\overline{b},\overline{a}\rangle\}\rangle$
#App
$\langle\{\},\ id,\ \{d\#_?\langle\overline{b},\overline{a}\rangle\}\rangle$
#Pr
$\langle\{\},\ id,\ \{d\#_?\overline{b},\ d\#_?\overline{a}\}\rangle$
#At
$\langle\{\},\ id,\ \{d\#_?\overline{a}\}\rangle$
#At
$\langle\{\},\ id,\ \{\}\rangle$
#Success

Fig. 6. Derivation tree for Example $\langle\{\},\ id,\ \{[d]f_n^C\langle\overline{d},\overline{a}\rangle \approx_? [b]f_n^C\langle\overline{b},\overline{a}\rangle\}\rangle$

The OCaml implementation is based on a straightforward algorithmic strategy in which simplification paths are built choosing equations and freshness constraints in the order in which they appear in the unification problem (using

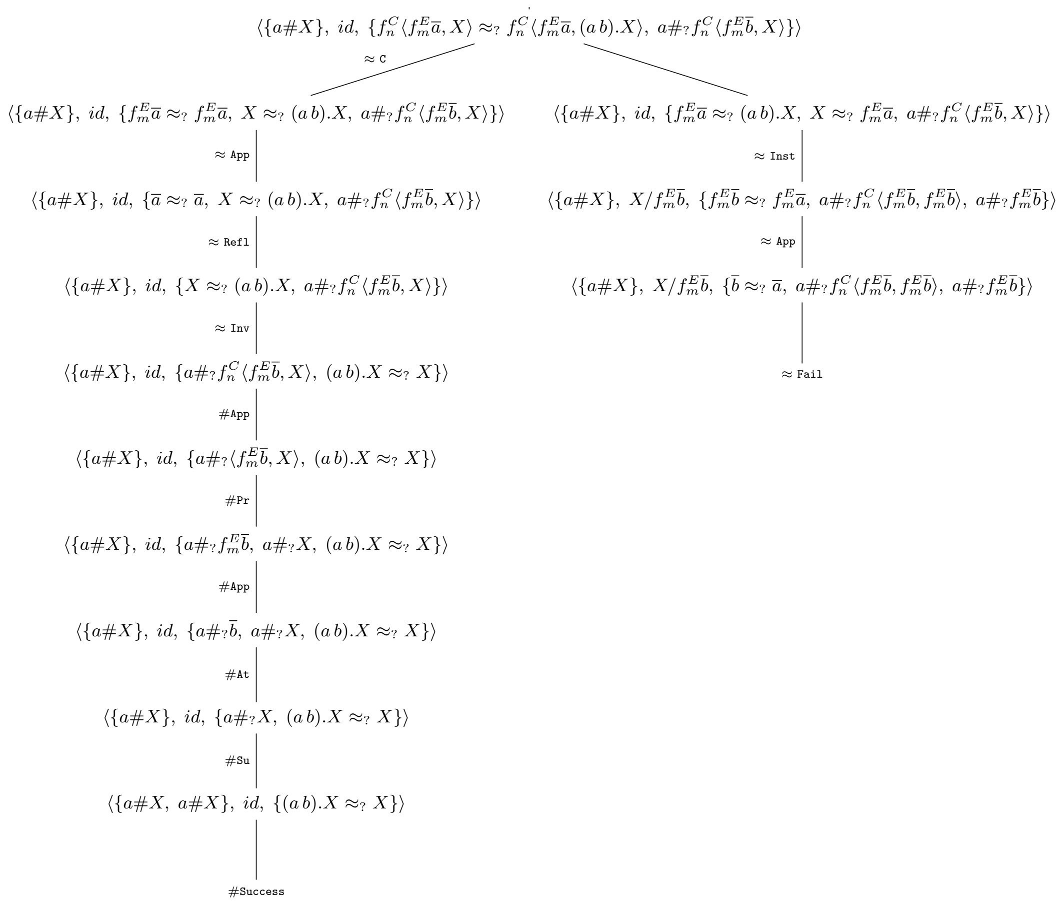
$\langle\{a\#X\},\ id,\ \{f_n^C\langle f_m^E\overline{a}, X\rangle \approx_? f_n^C\langle f_m^E\overline{a}, (a\,b).X\rangle,\ a\#_? f_n^C\langle f_m^E\overline{b}, X\rangle\}\rangle$
$\approx$ C
$\langle\{a\#X\},\ id,\ \{f_m^E\overline{a} \approx_? f_m^E\overline{a},\ X \approx_? (a\,b).X,\ a\#_? f_n^C\langle f_m^E\overline{b}, X\rangle\}\rangle$
$\approx$ App
$\langle\{a\#X\},\ id,\ \{\overline{a} \approx_? \overline{a},\ X \approx_? (a\,b).X,\ a\#_? f_n^C\langle f_m^E\overline{b}, X\rangle\}\rangle$
$\approx$ Refl
$\langle\{a\#X\},\ id,\ \{X \approx_? (a\,b).X,\ a\#_? f_n^C\langle f_m^E\overline{b}, X\rangle\}\rangle$
$\approx$ Inv
$\langle\{a\#X\},\ id,\ \{a\#_? f_n^C\langle f_m^E\overline{b}, X\rangle,\ (a\,b).X \approx_? X\}\rangle$
#App
$\langle\{a\#X\},\ id,\ \{a\#_?\langle f_m^E\overline{b}, X\rangle,\ (a\,b).X \approx_? X\}\rangle$
#Pr
$\langle\{a\#X\},\ id,\ \{a\#_? f_m^E\overline{b},\ a\#_? X,\ (a\,b).X \approx_? X\}\rangle$
#App
$\langle\{a\#X\},\ id,\ \{a\#_?\overline{b},\ a\#_? X,\ (a\,b).X \approx_? X\}\rangle$
#At
$\langle\{a\#X\},\ id,\ \{a\#_? X,\ (a\,b).X \approx_? X\}\rangle$
#Su
$\langle\{a\#X,\ a\#X\},\ id,\ \{(a\,b).X \approx_? X\}\rangle$
#Success
$\langle\{a\#X\},\ id,\ \{f_m^E\overline{a} \approx_? (a\,b).X,\ X \approx_? f_m^E\overline{a},\ a\#_? f_n^C\langle f_m^E\overline{b}, X\rangle\}\rangle$
$\approx$ Inst
$\langle\{a\#X\},\ X/f_m^E\overline{b},\ \{f_m^E\overline{b} \approx_? f_m^E\overline{a},\ a\#_? f_n^C\langle f_m^E\overline{b}, f_m^E\overline{b}\rangle,\ a\#_? f_m^E\overline{b}\}\rangle$
$\approx$ App
$\langle\{a\#X\},\ X/f_m^E\overline{b},\ \{\overline{b} \approx_? \overline{a},\ a\#_? f_n^C\langle f_m^E\overline{b}, f_m^E\overline{b}\rangle,\ a\#_? f_m^E\overline{b}\}\rangle$
$\approx$ Fail

Fig. 7. Derivation tree for $\langle\{a\#X\},\ id,\ \{[a]f_n^C\langle f_m^E\overline{a}, X\rangle \approx_? [b]f_n^C\langle f_m^E\overline{b}, X\rangle\}\rangle$

a list data structure). The rules in $\Rightarrow_{\approx}$ and $\Rightarrow_{\#}$ are applied in this order, according to the construction of derivation trees in Definition 10. Each set of rules is applied in the order in which the rules appear in Fig. 4 and Fig. 5. As soon as an equation or a freshness constraint can not be simplified the algorithm fails.

The output for Example 3 and other interesting ones are included.

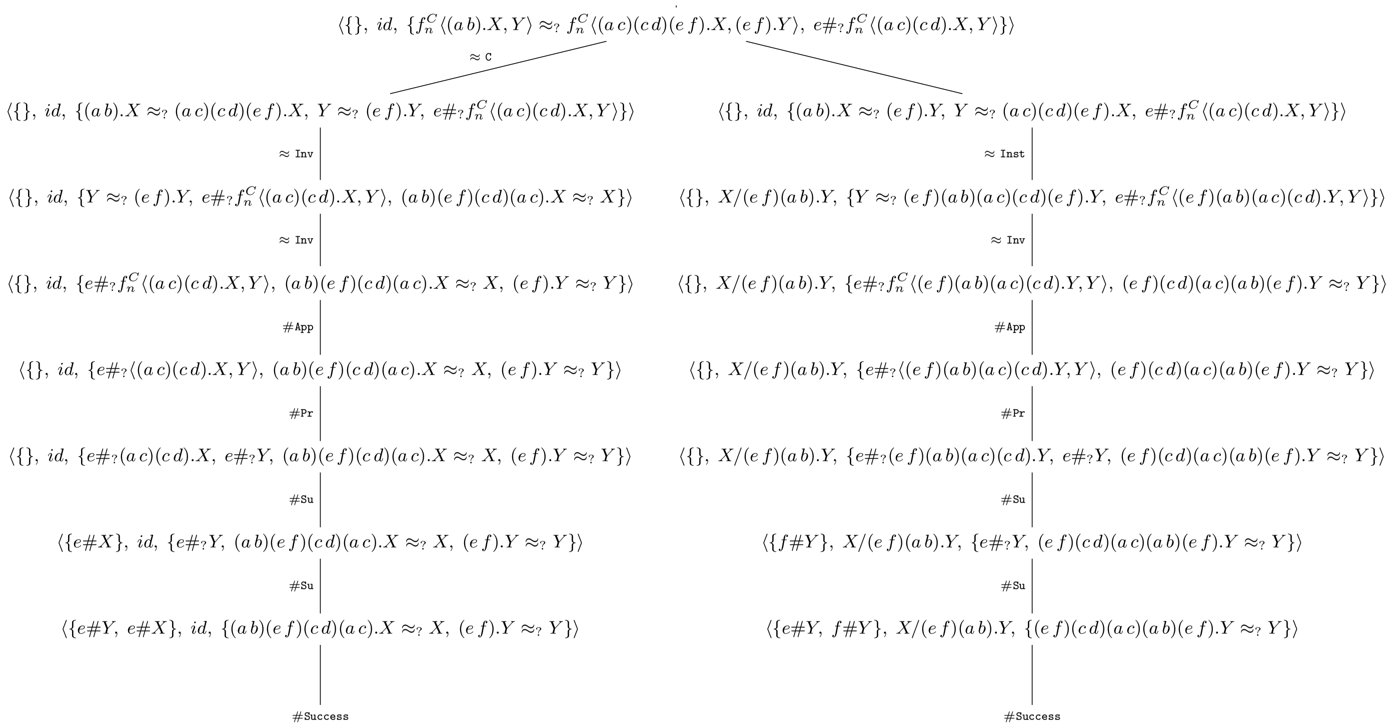
$\langle\{\},\ id,\ \{f_n^C\langle(a\,b).X,Y\rangle \approx_? f_n^C\langle(a\,c)(c\,d)(e\,f).X,(e\,f).Y\rangle,\ e\#_? f_n^C\langle(a\,c)(c\,d).X,Y\rangle\}\rangle$
$\approx$ C
$\langle\{\},\ id,\ \{(a\,b).X \approx_? (a\,c)(c\,d)(e\,f).X,\ Y \approx_? (e\,f).Y,\ e\#_? f_n^C\langle(a\,c)(c\,d).X,Y\rangle\}\rangle$
$\approx$ Inv
$\langle\{\},\ id,\ \{Y \approx_? (e\,f).Y,\ e\#_? f_n^C\langle(a\,c)(c\,d).X,Y\rangle,\ (a\,b)(e\,f)(c\,d)(a\,c).X \approx_? X\}\rangle$
$\approx$ Inv
$\langle\{\},\ id,\ \{e\#_? f_n^C\langle(a\,c)(c\,d).X,Y\rangle,\ (a\,b)(e\,f)(c\,d)(a\,c).X \approx_? X,\ (e\,f).Y \approx_? Y\}\rangle$
#App
$\langle\{\},\ id,\ \{e\#_?\langle(a\,c)(c\,d).X,Y\rangle,\ (a\,b)(e\,f)(c\,d)(a\,c).X \approx_? X,\ (e\,f).Y \approx_? Y\}\rangle$
#Pr
$\langle\{\},\ id,\ \{e\#_?(a\,c)(c\,d).X,\ e\#_?Y,\ (a\,b)(e\,f)(c\,d)(a\,c).X \approx_? X,\ (e\,f).Y \approx_? Y\}\rangle$
#Su
$\langle\{e\#X\},\ id,\ \{e\#_?Y,\ (a\,b)(e\,f)(c\,d)(a\,c).X \approx_? X,\ (e\,f).Y \approx_? Y\}\rangle$
#Su
$\langle\{e\#Y,\ e\#X\},\ id,\ \{(a\,b)(e\,f)(c\,d)(a\,c).X \approx_? X,\ (e\,f).Y \approx_? Y\}\rangle$
#Success
$\langle\{\},\ id,\ \{(a\,b).X \approx_? (e\,f).Y,\ Y \approx_? (a\,c)(c\,d)(e\,f).X,\ e\#_? f_n^C\langle(a\,c)(c\,d).X,Y\rangle\}\rangle$
$\approx$ Inst
$\langle\{\},\ X/(e\,f)(a\,b).Y,\ \{Y \approx_? (e\,f)(a\,b)(a\,c)(c\,d)(e\,f).Y,\ e\#_? f_n^C\langle(e\,f)(a\,b)(a\,c)(c\,d).Y,Y\rangle\}\rangle$
$\approx$ Inv
$\langle\{\},\ X/(e\,f)(a\,b).Y,\ \{e\#_? f_n^C\langle(e\,f)(a\,b)(a\,c)(c\,d).Y,Y\rangle,\ (e\,f)(c\,d)(a\,c)(a\,b)(e\,f).Y \approx_? Y\}\rangle$
#App
$\langle\{\},\ X/(e\,f)(a\,b).Y,\ \{e\#_?\langle(e\,f)(a\,b)(a\,c)(c\,d).Y,Y\rangle,\ (e\,f)(c\,d)(a\,c)(a\,b)(e\,f).Y \approx_? Y\}\rangle$
#Pr
$\langle\{\},\ X/(e\,f)(a\,b).Y,\ \{e\#_?(e\,f)(a\,b)(a\,c)(c\,d).Y,\ e\#_?Y,\ (e\,f)(c\,d)(a\,c)(a\,b)(e\,f).Y \approx_? Y\}\rangle$
#Su
$\langle\{f\#Y\},\ X/(e\,f)(a\,b).Y,\ \{e\#_?Y,\ (e\,f)(c\,d)(a\,c)(a\,b)(e\,f).Y \approx_? Y\}\rangle$
#Su
$\langle\{e\#Y,\ f\#Y\},\ X/(e\,f)(a\,b).Y,\ \{(e\,f)(c\,d)(a\,c)(a\,b)(e\,f).Y \approx_? Y\}\rangle$
#Success

Fig. 8. Derivation tree for Example 3