Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 344 (2019) 47–65

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

A Formalisation of Nominal C-Matching through Unification with Protected Variables

Mauricio Ayala-Rincón[†,‡] [1,2]
Washington de Carvalho-Segundo[‡][4] Maribel Fernández[*][3]
Daniele Nantes-Sobrinho[†][5]

Departments of [†] *Mathematics and* [‡] *Computer Science*
Universidade de Brasília
Brasília D.F., Brazil

[*] *Department of Informatics*
King's College London
London, U.K.

Abstract

This work adapts a formalisation in Coq of a nominal C-unification algorithm to include "protected variables" that cannot be instantiated during the unification process. By introducing protected variables we are able to reuse the C-unification algorithm to solve nominal C-matching (as well as equality check) problems. From the algorithmic point of view having this extension would be enough to adapt the C-unification algorithm for dealing with equational check as well as with C-matching problems, but the resulting algorithms would not be formally checked by simple reuse of the original formalisation. This paper describes the additional effort necessary in order to adapt the specification and reuse previous formalisations.

Keywords: Nominal Unification, Nominal Matching, Commutative Theory, C-Unification, Formal Methods, Coq

1 Introduction

Nominal unification [24,23] is the problem of solving equations between nominal terms, that is, terms that include binding operators, in which solutions should be defined modulo α-equivalence. *Nominal matching* is a restriction of nominal unification, in which only one side of the equation can be instantiated. This work

[1] Research partially supported by the Brazilian Federal District Foundation FAPDF within grant DE 0193001369/2016.

[2] Email: ayala@unb.br Author partially supported by a high productivity research grant from the Brazilian National Council for Scientific and Technological Development CNPq.

[3] Email: maribel.fernandez@kcl.ac.uk

[4] Email: wtonribeiro@gmail.com

[5] Email: dnantes@unb.br

https://doi.org/10.1016/j.entcs.2019.07.004

is about a formalisation in Coq of a nominal matching algorithm, where terms may include commutative operators. This problem is known as *nominal C-unification.*

In nominal syntax [19], terms include function symbols, abstractions, and two kinds of variables: *atoms* and *unknowns* (or simply variables). *Atoms* are used to represent object-level variables whereas *unknowns* behave like first-order variables, except that they can have "suspended atom permutations" which act when the variable is instantiated by a term. Atoms can be abstracted over terms, the nominal term $[a]s$ represents the abstraction of a in s and α-equivalence is axiomatised by means of a freshness relation $a\#t$ (read: a is fresh in t) and name-swappings $(a\,b)$, which implement renamings. For example, the first-order logic formula $\forall a.a \geq 0$ can be written as a nominal term $\forall([a]\mathtt{geq}(a, 0))$, using function symbols $\forall$ and $\mathtt{geq}$ and an abstracted atom a. Notice that $\forall([a]\mathtt{geq}(\mathtt{a}, \mathtt{X})) \approx^?_\alpha \forall([b]\mathtt{geq}(Y, 0))$ is a nominal unification problem whose solution should be such that a does not occur free in Y.

Nominal unification is a decidable problem and efficient nominal unification algorithms are available [18,12,11], that compute solutions consisting of freshness contexts (containing freshness constraints of the form $a\#X$) and substitutions. Nominal unification modulo commutativity (C), consists of nominal unification problems in which the signature of terms contains commutative function symbols. This algebraic property has to be taken into account during the unification process.

In [2], we proposed a nominal C-unification algorithm based on a set of simplification rules. This algorithm generates, for each solvable nominal C-unification problem, a finite set of *fixed point equations* of the form $\pi \cdot X \approx^?_\alpha X$, where π is a permutation and X is a variable, together with a set of *freshness constraints* and a *substitution*. In contrast, the output of the standard nominal unification algorithm consists only of substitutions and freshness constraints. Fixed point equations can be easily eliminated in the standard unification algorithm (they are replaced by freshness constraints), but this is not the case in the presence of commutative symbols. For instance, the fixed point equation $(a\,b) \cdot X \approx^?_{\alpha,C} X$ has infinite solutions $X/a+b, X/(a+b)+(a+b), \ldots$ (see [3] for a procedure to generate solutions of fixed point equations).

The approach in [2] was specified and its correctness and completeness formalised in Coq. In this paper, we adapt the nominal C-unification algorithm specified in [2] to provide a fully verified specification of a nominal C-matching approach. An additional parameter $\mathcal{X}$ that is a set of *protected variables* is now considered as part of the input problem. These variables will not be instantiated during the unification process and therefore, the domain of solutions will be disjoint from this set. If the set of protected variables is empty, the approach solves general nominal C-unification problems; if the set of protected variables consists of the variables occurring in the right-hand side of equations in the problem given as input, the algorithm will output a C-matching solution of this problem, if such solution exists; and if the set of protected variables consists of all the variables occurring in the input problem, the algorithm becomes a C-equational checker. Although these conclusions are obvious from the operational point of view, one cannot straightforwardly reuse the formalisation of the nominal C-unification specification to verify the derived nominal C-matching algorithm. In this paper, we show how the formalisation of correctness and completeness of the nominal C-matching specification was reached.

To summarise, the contributions of this paper are:

- An extension of the nominal C-unification algorithm proposed in [2] that adds a parameter for *protected variables*. The paper shows how previous formalisations were reused in order to prove termination, soundness and completeness of this extension.
- An algorithm for nominal C-matching that is obtained by defining the set of protected variables to be the set of variables on the right-hand side of equations in the input problem. Necessary specifications of nominal C-matching problems and solutions are required and then the algorithm is proven to be terminating, sound and complete.

Additional products are obtained that also require formalisation. For instance as a by-product, a nominal C-equational checker is obtained, by defining the set protected variables to be the set of all variables in the input problem.

All the proofs presented in this paper were formalised in the proof management system Coq, and are available at `http://ayala.mat.unb.br/publications.html`.

Related work

Equational unification and matching have been studied in automated reasoning and deduction for more than three decades providing interesting problems (some still open) and efficient solutions. For instance, F. Baader, K. Schulz, P. Narendran, M. Schmidt-Schaußand W. Snyder, in [8,9,10], have investigated several aspects related with general unification with equational theories and combinations of disjoint equational theories, whereas F. Fages, D. Kapur, P. Narendran and J.H. Siekmann have studied associative and/or commutative unification, matching and their complexities [14,15,16,17,21,22].

Nominal equational unification was initially investigated in [5] and [20]: in the former paper, the relation of *nominal narrowing* is defined and a *lifting* result relating nominal narrowing and unification is proven, whereas in the latter paper a nominal unification approach for higher order expressions with recursive `let` is presented.

The nominal C-unification algorithm proposed in [2] outputs a triple consisting of a substitution, a freshness context and a set of fixed point problems, and it was noticed that the set of fixed point equations could generate infinite solutions. In order to give explicit solutions for the fixed point problems, in [1], combinatorial solutions based on permutation cycles and pseudo-cycles were generated, and an exhaustive search algorithm was given. Thus, if solutions are expressed only with freshness contexts and substitutions, nominal C-unification is infinitary, in contrast with standard first-order C-unification which is finitary.

In addition to the Coq formalisation of nominal C-unification in [2], there are formalisations of non equational nominal unification in Isabelle [23] and PVS [7]. Regarding formalisations in the (non nominal) standard syntax, Contejean [13] formalised AC-matching in Coq.

Recently, a new axiomatisation of the alpha equivalence relation for nominal terms was presented [6], which is based on fixed point constraints and allows a finite representation of solutions of nominal C-unification problems, consisting of a substitution and a fixed point context.

Organisation

Section 2 presents basic concepts and notations. Section 3 presents the extension of simplification rules for nominal C-unification with protected variables, and discusses how the formalisations of termination, soundness and completeness were adapted. Section 4 introduces the nominal C-matching algorithm, and discusses the formalisation of its termination, soundness and completeness properties. Section 5 concludes the paper.

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 $\mathtt{dom}(\pi) := \{a \in \mathcal{A} \mid \pi \cdot a \neq a\}$. We will assume as in [4] countable sets of function symbols with different equational properties such as associativity, commutativity, idempotence, etc. Function symbols have superscripts that indicate their equational properties; thus, f_k^C will denote the k^{th} function symbol that is commutative and $f_j^{\emptyset}$ the j^{th} function symbol without any equational property.

Definition 2.1 [Nominal Grammar] *Nominal terms* are generated by the following grammar.

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

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

The 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.

Definition 2.2 [Permutation action] The *action of a permutation* π on a term t, denoted as $\pi \cdot t$, is recursively defined as:

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

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: $ds(\pi, \pi') := \{a \in \mathcal{A} \mid \pi \cdot a \neq \pi' \cdot a\}$.

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

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

Definition 2.3 [Substitution action] The *action of a substitution* σ on a term t, denoted $t\sigma$, is defined recursively as follows:

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

Example 2.4 For term $t = \langle (a\ b).X, f(e)\rangle$ and substitution $\sigma = \{X/[a]a\}$, we obtain that $((a\ b).X)\sigma = \langle [b]b, f(e)\rangle \approx_\alpha \langle [a]a, f(e)\rangle = X\sigma$, for some unary function symbol f in the signature.

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

Lemma 2.5 (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 $dom(\Delta)$ is the set of atoms appearing in it; $\Delta|_X$ denotes the restriction of Δ to the freshness constraints on X: $\{a\#X \mid a\#X \in \Delta\}$. The rules in Fig. 1 are used to check if an atom a is fresh in a nominal term t under a freshness context ∇, 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 $(\approx_\alpha \mathbf{[ab]})$.

Remark 2.6 $\mathtt{dom}(\pi)\#X$ and $ds(\pi,\pi')\#X$ denote, respectively, the sets $\{a\#X \mid a \in \mathtt{dom}(\pi)\}$ and $\{a\#X \mid a \in ds(\pi,\pi')\}$. Notice that $\mathtt{dom}(\pi) = ds(\pi, id)$.

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

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

Fig. 1. Rules for the freshness relation

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

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

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

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

Key properties of the nominal freshness and α-equivalence relations have been extensively explored in previous works [4,7,23,24], among them we have *freshness preservation*: If $\nabla \vdash a\,\#\,s$ and $\nabla \vdash s \approx_\alpha t$, then $\nabla \vdash a\,\#\,t$; *equivariance*: for all permutations π, if $\nabla \vdash s \approx_\alpha t$ then $\nabla \vdash \pi \cdot s \approx_\alpha \pi \cdot t$; and, **equivalence**: $\nabla \vdash \_ \approx_\alpha \_$ is an equivalence relation.

3 A nominal C-unification algorithm with protected variables

In [2] we proposed a nominal C-unification algorithm which used sets of transformation rules to deal with equations (Figure 4) and another set of rules to deal with freshness constraints and contexts (Figure 3). These rules act over triples of the form $\langle\nabla,\sigma,P\rangle$, where σ is a substitution. In this work, we will deal with nominal C-equational problems including another parameter that is a set $\mathcal{X}$ of protected variables. This parameter indicates which variables are forbidden to be instantiated. The quadruple that will be associated with a C-equational problem of the form $\langle\nabla,\mathcal{X},P\rangle$ is $\langle\nabla,\mathcal{X},id,P\rangle$. Calligraphic uppercase letters (e.g., $\mathcal{P},\mathcal{Q},\mathcal{R}$, etc) will denote quadruples.

Definition 3.1 [Unification and matching problem] A *unification problem* is a triple $\langle \nabla, \mathcal{X}, P\rangle$, where ∇ is a *freshness context*, $\mathcal{X}$ a set of protected variables, and P is a finite set of *equations* and *freshness constraints* of the form $s \approx_? t$ and $a\#_? s$, respectively, 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. A *matching problem* consists only of a pair $\langle \nabla, P\rangle$.

Definition 3.2 [Solution for a C-unification problem with preserved variables] A *solution* for a quadruple $\mathcal{P} = \langle \Delta, \mathcal{X}, \delta, P\rangle$ is a pair $\langle \nabla, \sigma\rangle$, where the domain of σ has no variables in $\mathcal{X}$, and the following conditions are satisfied:

(i) $\nabla \vdash \Delta\sigma$;

(ii) if $a\#_? t \in P$ then $\nabla \vdash a \# t\sigma$;

(iii) if $s \approx_? t \in P$ then $\nabla \vdash s\sigma \approx_{\{\alpha, C\}} t\sigma$;

(iv) there exists λ such that $\nabla \vdash \delta\lambda \approx \sigma$.

A *solution for a C-equational problem with preserved variables* $\langle \Delta, \mathcal{X}, P\rangle$ is a solution for the associated quadruple $\langle \Delta, \mathcal{X}, id, P\rangle$. The *solution set* for a problem or quadruple $\mathcal{P}$ is denoted by $\mathcal{U}_C(\mathcal{P})$.

We will denote the set of variables occurring in the set P of a problem $\mathcal{P}$ or quadruple $\mathcal{P} = \langle \nabla, \mathcal{X}, \sigma, P\rangle$ as $\mathtt{var}(P)$ or $\mathtt{var}(\mathcal{P})$, respectively.

When $\mathcal{X}$ equals $\emptyset$, the notions of C-unification problem with preserved variables, and its solutions coincide with the corresponding notions for C-unification as given in [2]. For simplicity, C-unification problems and solutions with preserved variables will be called just C-unification problems and solutions.

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

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

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

Fig. 3. Reduction rules for freshness problems

w

These inference rules transform a nominal unification problem into a finite set of unification problems consisting only of *fixed point* equations, i.e., equations of the form $\pi.X \approx_? X$, together with a substitution and a freshness context. P is called a *fixed point problem* if it is a set of fixed point equations. We will denote by $P_{\approx}, P_{\#}$ and $P_{\mathrm{fp}_{\approx}}$ the sets of equations, freshness problems and fixed point equations in the set P of a unification problem $\langle \nabla, \mathcal{X}, P\rangle$.

Also, we use rules described in Fig. 5, called *unification steps*, which give a strategy for application of rules specified as presented in Figs. 4 and 3.

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

$$(\approx_? \textbf{inv})\frac{\langle \nabla, \mathcal{X}, \sigma, P \uplus \{\pi.X \approx_? \pi'.X\}\rangle}{\langle \nabla, \mathcal{X}, \sigma, P \cup \{\pi \oplus (\pi')^{-1}.X \approx_? X\}\rangle}, \text{ if } \pi' \neq \mathtt{id} \qquad (\approx_? \textbf{app})\frac{\langle \nabla, \mathcal{X}, \sigma, P \uplus \{f_k^E\, s \approx_? f_k^E\, t\}\rangle}{\langle \nabla, \mathcal{X}, \sigma, P \cup \{s \approx_? t\}\rangle}, \text{ if } E \neq C$$

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

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

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

Fig. 4. Reduction rules for equational problems

Differently from [2], rule ($\approx_?$ **inst**), checks whether the variable X is a protected variable, before applying the instantiation.

Notice that the set $\mathcal{X}$ has not effect on the rules for freshness. Rules in Fig. 4 will be applied without restrictions by use of rule $(\upsilon_{\approx})$, but freshness constraints are reduced only when all equations were reduced and the problem consists of fixed point equations. This fact is expressed by the condition $P_{\approx} = P_{fp_{\approx}}$ in rule $(\upsilon_{\#})$.

$$(\upsilon_{\approx})\frac{\mathcal{P} \Rightarrow_{\approx} \mathcal{Q}}{\mathcal{P} \Rightarrow_{\upsilon} \mathcal{Q}} \qquad (\upsilon_{\#})\frac{\mathcal{P} \Rightarrow_{\#} \mathcal{Q}}{\mathcal{P} \Rightarrow_{\upsilon} \mathcal{Q}}, P_{\approx} = P_{fp_{\approx}}$$

Fig. 5. Unification step

Derivation with rules of Fig. 4 is denoted by $\Rightarrow_{\approx}$; thus, $\langle \nabla, \mathcal{X}, \sigma, P\rangle \Rightarrow_{\approx} \langle \nabla, \mathcal{X}, \sigma', P'\rangle$ means that the second quadruple is obtained from the first one by application of one rule. We will use the standard rewriting nomenclature, e.g., we will say that $\mathcal{P}$ is a *normal form* or *irreducible* by $\Rightarrow_{\approx}$, denoted by $\Rightarrow_{\approx}$-*nf*, whenever there is no $\mathcal{Q}$ such that $\mathcal{P} \Rightarrow_{\approx} \mathcal{Q}$; $\Rightarrow_{\approx}^{*}$ and $\Rightarrow_{\approx}^{+}$ denote respectively derivations in zero or more and one or more applications of the rules in Fig. 4. Derivation with rules of Fig. 3 is denoted by $\Rightarrow_{\#}$.

The theorem below summarises the termination and correctness results regarding the algorithm proposed.

Theorem 3.3 (Properties of $\Rightarrow_{\approx}$, $\Rightarrow_{\#}$ and $\Rightarrow_{\upsilon}$)

(i) *(Decidability of $\Rightarrow_{\approx}$, $\Rightarrow_{\#}$ and $\Rightarrow_{\upsilon}$)Given a quadruple $\mathcal{P}$, it is possible to decide whether $\mathcal{P}$ is a normal form w.r.t. $\Rightarrow_{\approx}$ (resp. $\Rightarrow_{\#}$) or there exists $\mathcal{Q}$ such that $\mathcal{P} \Rightarrow_{\approx} \mathcal{Q}$ (resp. $\mathcal{P} \Rightarrow_{\#} \mathcal{Q}$).*

(ii) *(Termination of $\Rightarrow_{\approx}$, $\Rightarrow_{\#}$ and $\Rightarrow_{\upsilon}$)The relations $\Rightarrow_{\approx}$, $\Rightarrow_{\#}$ and $\Rightarrow_{\upsilon}$ are terminating.*

(iii) *(Completeness of $\Rightarrow_{\#}$) If $\mathcal{P} \Rightarrow_{\#} \mathcal{Q}$, then $\langle \nabla, \sigma \rangle \in \mathcal{U}_C(\mathcal{P})$ if and only if $\langle \nabla, \sigma \rangle \in \mathcal{U}_C(\mathcal{Q})$.*

Proof. The proofs are obtained by adjusting proofs in [2] taking into account the set of variables $\mathcal{X}$. □

Remark 3.4 We will call $\mathcal{Q}$ a *leaf* if it is a normal form w.r.t. $\Rightarrow_{\upsilon}$.

For completeness, we will restrict ourselves to idempotent solutions, in order to obtain such property, we will restrict the applications of rules to *valid* quadruples.

Definition 3.5 [Valid quadruple] $\mathcal{P} = \langle \nabla, \mathcal{X}, \sigma, P \rangle$ *is valid* if $\mathtt{im}(\sigma) \cap \mathtt{dom}(\sigma) = \emptyset$ and $\mathtt{dom}(\sigma) \cap \mathtt{var}(P) = \emptyset$.

As for Theorem 3.3 , the formalisations of Lemmas 3.6, 3.7, 3.8, and Theorems 3.10 and 3.12 are quite similar to the proofs of corresponding lemmas in [2]. For this reason, they are omitted.

Lemma 3.6 (Preservation of valid quadruples)

(i) *If $\mathcal{P} \Rightarrow_{\#} \mathcal{Q}$ or $\mathcal{P} \Rightarrow_{\approx} \mathcal{Q}$, and $\mathcal{P}$ is valid then $\mathcal{Q}$ is also valid.*

(ii) *If $\mathcal{P} \Rightarrow_{\upsilon} \mathcal{Q}$ and $\mathcal{P}$ is valid then $\mathcal{Q}$ is also valid.*

Lemma 3.7 (Preservation of solutions) *Let $\mathcal{P}$ be a valid quadruple.*

(i) *If $\mathcal{P} \Rightarrow_{\approx} \mathcal{Q}$ and $\langle \nabla, \sigma \rangle \in \mathcal{U}_C(\mathcal{Q})$, then $\langle \nabla, \sigma \rangle \in \mathcal{U}_C(\mathcal{P})$.*

(ii) *If $\mathcal{P} \Rightarrow_{\upsilon} \mathcal{Q}$ and $\langle \nabla, \sigma \rangle \in \mathcal{U}_C(\mathcal{Q})$, then $\langle \nabla, \sigma \rangle \in \mathcal{U}_C(\mathcal{P})$.*

Lemma 3.8 (Completeness of $\Rightarrow_{\approx}$ and $\Rightarrow_{\upsilon}$) *Let $\mathcal{P}$ be a valid quadruple.*

(i) *If $\mathcal{P}$ is not a normal form w.r.t. $\Rightarrow_{\approx}$, then $\langle \nabla, \sigma \rangle \in \mathcal{U}_C(\mathcal{P})$ if and only if there exists $\mathcal{Q}$ such that $\mathcal{P} \Rightarrow_{\approx} \mathcal{Q}$ and $\langle \nabla, \sigma \rangle \in \mathcal{Q}$.*

(ii) *If $\mathcal{P}$ is not a leaf, then $\langle \nabla, \sigma \rangle \in \mathcal{U}_C(\mathcal{P})$ if and only if there exists $\mathcal{Q}$ such that $\mathcal{P} \Rightarrow_{\upsilon} \mathcal{Q}$ and $\langle \nabla, \sigma \rangle \in \mathcal{Q}$.*

Example 3.9 [Nominal C-unification with $\mathcal{X}$ equals to the empty set] This example exhibits the execution of the nominal C-unification algorithm for the initial problem

$$\mathcal{P} = \langle \emptyset,\ \emptyset,\ id,\ \{[a]f\langle [b](X * Y), Z\rangle \approx [b]f\langle [a](\overline{a} * X), Z\rangle\}\rangle,$$

where the set of protected variables ($\mathcal{X}$) is empty; thus, there exists no restriction over the variables of the problem. Notice that the application of rule ($\approx_?$ **C**) generates two branches that are represented by items (i) and (ii) in the example. The algorithm generates the leaves $\langle \{a\#Z\},\ \emptyset,\ \{X/\overline{b}, Y/(a\,b).X\},\ \{(a\,b).Z \approx_? Z\}\rangle$, and $\langle \{a\#Z\},\ \emptyset,\ \{Y/\overline{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z\}\rangle$.

By Theorem 3.10, the union of the solutions of these two leaves is equal to the set of solutions of the initial problem $\mathcal{P}$. As shown in [1], the complete set of solutions of $\langle \{a\#Z\},\ \emptyset,\ \{X/\overline{b}, Y/(a\,b).X\},\ \{(a\,b).Z \approx_? Z\}\rangle$ is unitary whereas the complete set of solutions of $\langle \{a\#Z\},\ \emptyset,\ \{Y/\overline{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z\}\rangle$ is infinite. Figure 3 illustrates the C-unification derivation three for $\mathcal{P}$.

$$\begin{tikzcd}
 & \mathcal{P} = \langle \emptyset, \emptyset, id, \{[a]f\langle [b](X*Y), Z\rangle \approx_? [b]f\langle [a](\overline{a}*X), Z\rangle\}\rangle \arrow[d, Rightarrow, "(v\approx)"', "*"] & \\
 & \langle \emptyset, \emptyset, id, \{X*Y \approx_? \overline{b}*(ab)\cdot X, Z \approx_? (ab)\cdot Z, a\#_? f\langle [a](\overline{a}*X), Z\rangle\}\rangle \arrow[dl, Rightarrow, "(\approx_? \mathrm{C})"'] \arrow[dr, Rightarrow, "(\approx_? \mathrm{C})"] & \\
\text{(i)}\ \langle \emptyset, \emptyset, id, \left\{ \begin{array}{l} X \approx_? \overline{b}, Y \approx_? (a\ b)\cdot X, \\ Z \approx_? (a\ b)\cdot Z, a\#_? f\langle [a](\overline{a}*X), Z\rangle \end{array} \right\}\rangle \arrow[d, Rightarrow, "(v\approx)"', "*"] & & \langle \emptyset, \emptyset, id, \left\{ \begin{array}{l} X \approx_? (a\ b)\cdot X, Y \approx_? \overline{b}, \\ Z \approx_? (a\ b)\cdot Z, a\#_? f\langle [a](\overline{a}*X), Z\rangle \end{array} \right\}\rangle\ \text{(ii)} \arrow[d, Rightarrow, "(v\approx)"', "*"] \\
{} \arrow[d, Rightarrow, "(v\#)"', "*"] & & {} \arrow[d, Rightarrow, "(v\#)"', "*"] \\
\langle \{a\#Z\}, \emptyset, \{X/\overline{b}, Y/(a\ b)\cdot X\}, \{(a\ b)\cdot Z \approx_? Z\}\rangle & & \langle \{a\#Z\}, \emptyset, \{Y/\overline{b}\}, \{(a\ b)\cdot X \approx_? X, (a\ b)\cdot Z \approx_? Z\}\rangle
\end{tikzcd}$$

Fig. 6. Derivation tree for nominal C-unification.

$\mathcal{P} = \langle \emptyset,\ \emptyset,\ id,\ \{[a]f\langle [b](X*Y), Z\rangle \approx [b]f\langle [a](\overline{a}*X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?[\mathbf{ab}])}$ $\langle \emptyset,\ \emptyset,\ id,\ \{f\langle [b](X*Y), Z\rangle \approx_? f\langle [b](\overline{b}*(a\,b).X), (a\,b).Z\rangle,\ a\#_? f\langle [a](\overline{a}* X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{app})}$ $\langle \emptyset,\ \emptyset,\ id,\ \{\langle [b](X*Y), Z\rangle \approx_? \langle [b](\overline{b}*(a\,b).X), (a\,b).Z\rangle,\ a\#_? f\langle [a](\overline{a}* X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{pair})}$ $\langle \emptyset,\ \emptyset,\ id,\ \{[b](X*Y) \approx_? [b](\overline{b}*(a\,b).X),\ Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a}* X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?[\mathbf{aa}])}$ $\langle \emptyset,\ \emptyset,\ id,\ \{X*Y \approx_? (\overline{b}*(a\,b).X),\ Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a}*X), Z\rangle\}\rangle$

(i)

$\Rightarrow_{(\approx_?\mathbf{C})}$ $\langle \emptyset,\ \emptyset,\ id,\ \{X \approx_? \overline{b},\ Y \approx_? (a\,b).X,\ Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a}* X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inst})}$ $\langle \emptyset,\ \emptyset,\ \{X/\overline{b}\},\ \{Y \approx_? (a\,b).X,\ Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a}* X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inst})}$ $\langle \emptyset,\ \emptyset,\ \{X/\overline{b},\ Y/(a\,b).X\},\ \{Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a}*X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle \emptyset,\ \emptyset,\ \{X/\overline{b},\ Y/(a\,b).X\},\ \{(a\,b).Z \approx_? Z,\ a\#_? f\langle [a](\overline{a}*X), Z\rangle\}\rangle$

$\Rightarrow_{(\#_?\mathbf{app})}$ $\langle \emptyset,\ \emptyset,\ \{X/\overline{b},\ Y/(a\,b).X\},\ \{(a\,b).Z \approx_? Z,\ a\#_? \langle [a](\overline{a}*X), Z\rangle\}\rangle$

$\Rightarrow_{(\#_?\mathbf{pair})}$ $\langle \emptyset,\ \emptyset,\ \{X/\overline{b},\ Y/(a\,b).X\},\ \{(a\,b).Z \approx_? Z,\ a\#_? [a](\overline{a}* X),\ a\#_? Z\}\rangle$

$\Rightarrow_{(\#_?\mathbf{a}[\mathbf{a}])}$ $\langle \emptyset,\ \emptyset,\ \{X/\overline{b},\ Y/(a\,b).X\},\ \{(a\,b).Z \approx_? Z,\ a\#_? Z\}\rangle$

$\Rightarrow_{(\#_?\mathbf{var})}$ $\langle \{a\#Z\},\ \emptyset,\ \{X/\overline{b}, Y/(a\,b).X\},\ \{(a\,b).Z \approx_? Z\}\rangle$

(ii)

$\Rightarrow_{(\approx_?\mathbf{C})}$ $\langle \emptyset,\ \emptyset,\ id,\ \{X \approx_? (a\,b).X,\ Y \approx_? \overline{b},\ Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a}* X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle \emptyset,\ \emptyset,\ id,\ \{(a\,b).X \approx_? X,\ Y \approx_? \overline{b},\ Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a}* X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inst})}$ $\langle\emptyset,\ \emptyset,\ \{Y/\bar{b}\},\ \{(a\,b).X \approx_? X,\ Z \approx_? (a\,b).Z,\ a\#_? f\langle[a](\bar{a} * X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle\emptyset,\ \emptyset,\ \{Y/\bar{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z,\ a\#_? f\langle[a](\bar{a} * X), Z\rangle\}\rangle$

$\Rightarrow_{(\#_?\mathbf{app})}$ $\langle\emptyset,\ \emptyset,\ \{Y/\bar{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z,\ a\#_? \langle[a](\bar{a} * X), Z\rangle\}\rangle$

$\Rightarrow_{(\#_?\mathbf{pair})}$ $\langle\emptyset,\ \emptyset,\ \{Y/\bar{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z,\ a\#_? [a](\bar{a} * X),\ a\#_? Z\}\rangle$

$\Rightarrow_{(\#_?\mathbf{a[a]})}$ $\langle\emptyset,\ \emptyset,\ \{Y/\bar{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z,\ a\#_? Z\}\rangle$

$\Rightarrow_{(\#_?\mathbf{var})}$ $\langle\{a\#Z\},\ \emptyset,\ \{Y/\bar{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z\}\rangle$

Theorem 3.10 (Correctness of $\Rightarrow_\upsilon^*$) *Let $\mathcal{P}$ be a valid quadruple.*

(i) *(Soundness of $\Rightarrow_\upsilon^*$) If $\mathcal{P} \Rightarrow_\upsilon^* \mathcal{Q}$, $\mathcal{Q}$ is a leaf and $\langle\nabla,\sigma\rangle \in \mathcal{U}_C(\mathcal{Q})$ then $\langle\nabla,\sigma\rangle \in \mathcal{U}_C(\mathcal{P})$.*

(ii) *(Completeness of $\Rightarrow_\upsilon^*$) $\langle\nabla,\sigma\rangle \in \mathcal{U}_C(\mathcal{P})$ if and only if there exists a leaf $\mathcal{Q}$ such that $\mathcal{P} \Rightarrow_\upsilon^* \mathcal{Q}$ and $\langle\nabla,\sigma\rangle \in \mathcal{Q}$.*

Definition 3.11 [Proper problem] A quadruple $\mathcal{P} = \langle\Delta, \mathcal{X}, \delta, P\rangle$ is called a *proper problem* if every commutative function symbol in P has a tuple as argument.

Theorem 3.12 (Characterisation of successful leaves) *Let $\mathcal{Q} = \langle\Delta, \mathcal{X}, \delta, Q\rangle$ a leaf, if $\mathcal{Q}$ is a proper problem and there exists $\langle\nabla,\sigma\rangle \in \mathcal{U}_C(\mathcal{Q})$ with $\mathtt{dom}(\sigma) \cap \mathcal{X} = \emptyset$, then Q is a fixed point problem.*

4 Nominal C-matching

In this section we restrict our attention to *nominal matching problems*: a nominal unification problem whose solutions should be applied only to the left-hand side of the nominal equations.

We specify a nominal C-matching algorithm that consists of applications of the *matching step* rules presented in Figure 7. These rules basically apply the rules in Figure(s 4, 3 and) 5, but now the set $\mathcal{X}$ of protected variables plays an important role and should be defined as the variables occurring in the right-hand sides of the set of equational constraints P in the input problem.

Definition 4.1 [Protected variables and C-matching problems] The set of protected variables for a matching problem $\langle\nabla, P\rangle$ (see Definition 3.1) is the set of *right-hand side variables* of the equational constraints in P, denoted by $\mathtt{Rvar}(P)$, i.e., $\mathtt{Rvar}(P) = \{X \mid s \approx_? t \in P \text{ and } X \in \mathtt{var}(t)\}$. The quadruple associated with the C-matching problem $\langle\nabla, P\rangle$ is given by $\langle\nabla, \mathtt{Rvar}(P), id, P\rangle$.

When solving a problem according to the rules in Fig. 7 using the protected variables given in Def. 4.1, the domain of a substitution that is a solution will be disjoint from the set of right-hand side variables of the problem.

Derivation with rules of Fig. 7 is denoted by $\Rightarrow_\mu$.

$$(\mu_v)\,\frac{\mathcal{P} \Rightarrow_v \mathcal{Q}}{\mathcal{P} \Rightarrow_\mu \mathcal{Q}} \qquad (\mu_{\mathbf{fp}})\,\frac{\langle \nabla, \mathcal{X}, \sigma, P \uplus \{\pi.X \approx_? X\}\rangle}{\langle \nabla \cup \mathtt{dom}(\pi)\#X,\ \mathcal{X},\ \sigma,\ P\rangle}\,, P = P_{fp_\approx}$$

Fig. 7. Matching step

Definition 4.2 [Solution for a C-matching problem] A *C-matching solution* for a quadruple $\mathcal{P}$ of the form $\langle \Delta, \mathtt{Rvar}(P), \delta, P\rangle$ is a pair $\langle \nabla, \sigma\rangle$, where $\mathtt{dom}(\sigma) \cap \mathtt{Rvar}(P) = \emptyset$, and the following conditions are satisfied:

(i) $\nabla \vdash \Delta\sigma$;

(ii) $\nabla \vdash a\,\#\,t\sigma$, if $a\#_? t \in P$;

(iii) $\nabla \vdash s\sigma \approx_{\{\alpha,C\}} t$, if $s \approx_? t \in P$;

(iv) there is a substitution λ such that $\nabla \vdash \delta\lambda \approx \sigma$.

A *C-matching solution* for the problem $\langle \Delta, P\rangle$ is a solution for $\langle \Delta, \mathtt{Rvar}(P), id, P\rangle$, its associated C-matching problem. The *solution set* for a matching problem $\mathcal{P}$ is denoted by $\mathcal{M}_C(\mathcal{P})$.

Remark 4.3 We will call a quadruple $\mathcal{Q}$ a *matching leaf* if $\mathcal{Q}$ is a normal form w.r.t. $\Rightarrow_\mu$.

4.1 Auxiliary properties of nominal C-matching

We now present the main auxiliary lemmas related with nominal C-unification notions included in the formalisation.

Remark 4.4 In the specification, the set of protected variables of a C-matching problem is given as a parameter of the deductive system that remains unchanged during derivations. For simplicity, in the following results the protected set is denoted as $\mathcal{X}$.

Lemma 4.5 ($\mathcal{U}_C$ and $\mathcal{M}_C$ equivalence) *Let $\mathcal{P} = \langle \Delta, \mathcal{X}, \delta, P\rangle$ be a quadruple. Then, $\langle \nabla, \sigma\rangle \in \mathcal{U}_C(\mathcal{P})$ if and only if $\langle \nabla, \sigma\rangle \in \mathcal{M}_C(\mathcal{P})$.*

Proof. The formalisation follows straightforwardly from the definitions of $\mathcal{U}_C(\mathcal{P})$ and $\mathcal{M}_C(\mathcal{P})$. □

Remark 4.6 Despite the fact that the reduction rules (Figures 3, 4, and 7) preserve the set $\mathcal{X}$ of protected variables given as input, in the following formalised results, for quadruples of the form $\mathcal{P} = \langle \Delta,\ \mathcal{X},\ \delta,\ P\rangle$ and $\mathcal{Q} = \langle \Delta',\ \mathcal{X},\ \delta',\ Q\rangle$, where $\mathcal{P} \Rightarrow_\mu \mathcal{Q}$, the sets $\mathtt{Rvar}(P)$ and $\mathtt{Rvar}(Q)$ will be considered. These sets change after reduction steps using rules such as ($\approx_?$ **refl**), ($\approx_?$ **inst**) and ($\mu_{\mathbf{fp}}$), but as Lemma 4.7 shows, $\mathtt{Rvar}(Q) \subseteq \mathtt{Rvar}(P)$, therefore if $\mathtt{Rvar}(P) \subseteq \mathcal{X}$ then $\mathtt{Rvar}(Q) \subseteq \mathcal{X}$. Since in the matching algorithm $\mathcal{X}$ is the set of right-hand side variables of the input problem, for each quadruple in a derivation $\mathcal{P}_0 \Rightarrow_\mu \ldots \Rightarrow_\mu \mathcal{P}_n$ the right-hand side variables are in the protected set.

Lemma 4.7 (Preservation of Rvar by $\Rightarrow_\mu$) *Let $\mathcal{P} = \langle \Delta, \mathcal{X}, \delta, P\rangle$ and $\mathcal{Q} = \langle \Delta', \mathcal{X}, \delta', Q\rangle$ such that $\mathcal{P} \Rightarrow_\mu \mathcal{Q}$. Then $\mathtt{Rvar}(Q) \subseteq \mathtt{Rvar}(P)$.*

Proof. The formalisation follows by case analysis on the $\Rightarrow_\mu$ reduction. □

Corollary 4.8 (Intersection emptyness preservation with right-hand side variables by $\Rightarrow_\mu$) *Let $\mathcal{P}$ and $\mathcal{Q}$ be two quadruples, $\langle \Delta, \mathcal{X}, \delta, P\rangle$ and $\langle \Delta', \mathcal{X}, \delta', Q\rangle$, respectively, and $\mathcal{Y}$ be an arbitrary set of variables. If $\mathtt{Rvar}(P) \cap \mathcal{Y} = \emptyset$ and $\mathcal{P} \Rightarrow_\mu \mathcal{Q}$, then $\mathtt{Rvar}(Q) \cap \mathcal{Y} = \emptyset$.*

Proof. This is indeed an easy set theoretically based corollary of Lemma 4.7. □

Corollary 4.9 (Preservation of valid quadruples by $\Rightarrow_\mu$) *If $\mathcal{P} \Rightarrow_\mu \mathcal{Q}$ and $\mathcal{P}$ is valid then $\mathcal{Q}$ is also valid.*

Proof. The formalisation follows from Lemma 3.6. □

Lemma 4.10 (Decidability of $\Rightarrow_\mu$) *For all quadruple $\mathcal{P}$ it is possible to decide whether there exists $\mathcal{Q}$ such that $\mathcal{P} \Rightarrow_\mu \mathcal{Q}$. Thus, it is also possible to decide whether $\mathcal{P}$ is a leaf.*

Proof. The formalisation is obtained by decidability of the relation $\Rightarrow_\upsilon$ (item (i) of Theorem 3.3.) □

4.2 Main formalised properties for nominal C-matching

Theorem 4.11 (Termination of $\Rightarrow_\mu$) *The relation $\Rightarrow_\mu$ is terminating.*

Proof. The proof is by case analysis on the derivation rules of the relation $\Rightarrow_\mu$, and uses a lexicographic measure over sets of equation and freshness constraints. The measure is given by

$$\left\langle |\mathtt{var}(P)|,\ \sum_{s \approx_? t \in P} |s| + |t|,\ |P_\approx / P_{fp_\approx}|,\ \sum_{a \#_? s \in P} |s| \right\rangle$$

Let $\mathcal{P} = \langle \Delta, \mathcal{X}, \delta, P\rangle$ and $\mathcal{Q} = \langle \nabla, \mathcal{X}, \sigma, Q\rangle$ such that $\mathcal{P} \Rightarrow_\mu \mathcal{Q}$.

For the case of rule (μ_υ), Theorem 3.3 item (ii) is applied.

For the case of an application of rule $(\mu_{\mathbf{fp}})$, one observes that:

(i) $|\mathtt{var}(Q)| \leq |\mathtt{var}(P)|$,

(ii) $\sum_{s \approx_? t \in Q} |s| + |t| < \sum_{s \approx_? t \in P} |s| + |t|$,

(iii) $|Q_\approx / Q_{fp_\approx}| = |P_\approx / P_{fp_\approx}|$ and

(iv) $\sum_{a \#_? s \in Q} |s| = \sum_{a \#_? s \in P} |s|$.

Therefore the measure also decreases in this case, which concludes the proof. □

Lemma 4.12 (Preservation of solutions by $\Rightarrow_\mu$) *Let $\mathcal{P} = \langle \Delta, \mathcal{X}, \delta, P\rangle$ be a valid quatrule and $\mathcal{Q} = \langle \Delta', \mathcal{X}, \delta', Q\rangle$. If $\mathtt{Rvar}(P) \cap \mathtt{dom}(\sigma) = \emptyset$, $\mathcal{P} \Rightarrow_\mu \mathcal{Q}$ and $\langle \nabla, \sigma\rangle \in \mathcal{M}_C(\mathcal{Q})$, then $\langle \nabla, \sigma\rangle \in \mathcal{M}_C(\mathcal{P})$.*

Proof. The proof is by case analysis on the derivation rules of $\Rightarrow_\mu$. According Definition 4.2, one has $\mathtt{Rvar}(Q) \cap \mathtt{dom}(\sigma) = \emptyset$. From this, the hypothesis $\mathtt{Rvar}(P) \cap \mathtt{dom}(\sigma) = \emptyset$ and using Lemmas 3.7 (item (i)) and 4.5 one concludes the case of rule (μ_υ). For the case of rule $(\mu_{\mathbf{fp}})$, one needs to conclude the conditions of Definition 4.2

for the pair $\langle\nabla,\sigma\rangle$ w.r.t. $\mathcal{P}$. Condition (iv) is trivially satisfied. The first condition is proved just observing that every constraint $a\#X$ in Δ is also in $\Delta\cup\mathtt{dom}(\pi)$. The second condition is easy proved from the fact that if $a\#_?s\in P\uplus\{\pi.X\approx_? X\}$ then $a\#_?s\in P$. Then, one applies the hypothesis $\langle\nabla,\sigma\rangle\in\mathcal{M}_C(\mathcal{Q})$ using Definition 4.2, item (ii), to conclude. The third condition is proved by analysis of two cases. The first case is when $s\approx_? t\in P\uplus\{\pi.X\approx_? X\}$ being the equation $s\approx_? t$ equal to $\pi.X\approx_? X$. In this case, one starts proving the statement $X\cap\mathtt{dom}(\sigma)=\emptyset$ using the hypothesis $\mathtt{Rvar}(P\uplus\{\pi.X\approx_? X\})\cap\mathtt{dom}(\sigma)=\emptyset$. Form this, $(\pi.X)\sigma$ can be replaced by $\pi.X$ in the objective $\nabla\vdash\pi X\sigma\approx_{\{\alpha,C\}}X$, remaining to prove that $\nabla\vdash\pi.X\approx_{\{\alpha,C\}}X$. Then, using the condition (i) of Definition 4.2 of hypothesis $\langle\nabla,\sigma\rangle\in\mathcal{M}_C(\mathcal{Q})$ one has that $\nabla\vdash(\Delta\cup\mathtt{dom}(\pi)\#X)\sigma$. Since $X\notin\mathtt{dom}(\sigma)$, one concludes that $\mathtt{dom}(\pi)\#X\subseteq\nabla$ and then the objective is proved using the definition of $\approx_{\{\alpha,C\}}$ for the case of suspensions. The second case is when $s\approx_? t\in P$. This case is trivial, and uses hypothesis $\langle\nabla,\sigma\rangle\in\mathcal{M}_C(\mathcal{Q})$ with Definition 4.2, item (iii).□

Theorem 4.13 (Completeness of $\Rightarrow_\mu$) *Let $\mathcal{P}=\langle\Delta,\mathcal{X},\delta,P\rangle$ a valid quadruple that is not a matching leaf, if $\mathtt{Rvar}(P)\cap\mathtt{dom}(\sigma)=\emptyset$, then $\langle\nabla,\sigma\rangle\in\mathcal{M}_C(\mathcal{P})$ if and only if there exists $\mathcal{Q}$ such that $\mathcal{P}\Rightarrow_\mu\mathcal{Q}$ and $\langle\nabla,\sigma\rangle\in\mathcal{M}_C(\mathcal{Q})$.*

Proof. Necessity is proved by case analysis on the derivation rules of $\Rightarrow_\mu$. Lemma 4.10 is applied to the premise that $\mathcal{P}$ is not a matching leaf to obtain that there exists $\mathcal{Q}'$ such that $\mathcal{P}\Rightarrow_\mu\mathcal{Q}'$. Then for the case of rule (μ_v), using Lemmas 3.8 (item (iii)) and 4.5 it is proved the assertion that there exists $\mathcal{Q}''$ such that $\mathcal{P}\Rightarrow_v\mathcal{Q}''$ and $\langle\nabla,\sigma\rangle\in\mathcal{U}_\mathcal{C}(\mathcal{Q}')$. From this, using again Lemma 4.5, applying rule (μ_v) and using Corollary 4.8 one concludes. For the case of rule $(\mu_{\mathbf{fp}})$, $\mathcal{P}=\langle\Delta,\mathcal{X},\delta,P'\uplus\{\pi.X\approx_? X\}\rangle$ with $P'=P'_{fp_\approx}$. The quadruple $\mathcal{Q}=\langle\Delta\cup\mathtt{dom}(\pi)\#X,\mathcal{X},\delta,P\rangle$ will be a witness. Thus, $\mathcal{P}\Rightarrow_\mu\mathcal{Q}$ follows by an application of rule $(\mu_{\mathbf{fp}})$. To prove that $\langle\nabla,\sigma\rangle\in\mathcal{M}_C(\mathcal{Q})$, one has to show that the conditions of Definition 4.2 are satisfied, having as hypothesis that $\langle\nabla,\sigma\rangle\in\mathcal{M}_C(\mathcal{P})$. Conditions (ii), (iii) and (iv) are trivially verified and intersection emptiness is proved using Corollary 4.8. For condition (i), a constraint $a\#X$ is chosen that is in $\Delta\cup\mathtt{dom}(\pi)\#X$ to analyse if it is either in Δ or $\mathtt{dom}(\pi)\#X$. If $a\#X$ is in Δ the proof is trivial, otherwise one first proves the assertion that $\{X\}\cap\mathtt{dom}(\sigma)=\emptyset$ from the hypotheses that $\mathcal{P}$ is valid and $\mathtt{Rvar}(P)\cap\mathtt{dom}(\sigma)=\emptyset$. This allows to replace every $X\sigma$ and every $(\pi.X)\sigma$, respectively, just by X and $\pi.X$, because $X\notin\mathtt{dom}(\sigma)$. Since $\pi.X\approx_? X$ is in $P\uplus\{\pi.X\approx_? X\}$ and $\langle\nabla,\sigma\rangle\in\mathcal{M}_C(\mathcal{P})$, we have that $\nabla\vdash(\pi.X)\sigma\approx_{\{\alpha,C\}}X$, therefore $\nabla\vdash\pi.X\approx_{\{\alpha,C\}}X$ and then $\mathtt{dom}(\pi)\#X\subseteq\nabla$. On the other hand, having $a\in\mathtt{dom}(\pi)$ as hypothesis, one has to prove that $\nabla\vdash a\#X\sigma$, which is the same as $\nabla\vdash a\#X$. Using the fact that $\mathtt{dom}(\pi)\#X\subseteq\nabla$, one concludes.

Sufficiency is formalised as a direct consequence of Lemma 4.12. □

Theorem 4.14 (Soundness of $\Rightarrow_\mu^*$) *Let $\mathcal{P}=\langle\Delta,\mathcal{X},\delta,P\rangle$ be a valid quadruple and $\mathcal{P}\Rightarrow_\mu^*\mathcal{Q}$. If $\mathcal{Q}$ is a matching leaf and $\langle\nabla,\sigma\rangle\in\mathcal{M}_C(\mathcal{Q})$ such that $\mathtt{Rvar}(P)\cap\mathtt{dom}(\sigma)=\emptyset$ then $\langle\nabla,\sigma\rangle\in\mathcal{M}_C(\mathcal{P})$.*

Proof. The proof uses Corollaries 4.9 and 4.8 and Lemma 4.12, and it is done by induction on the number of steps of $\Rightarrow_\mu$. If $\mathcal{P}=\mathcal{Q}$ the proof is trivial. In the case were $\mathcal{P} \Rightarrow_\mu \mathcal{Q}$, Lemma 4.12 is applied to conclude. When $\mathcal{P} \Rightarrow_\mu \mathcal{R}$ and $\mathcal{R} \Rightarrow_\mu^* \mathcal{Q}$, one uses Lemma 4.12, IH and Lemmas 4.9 and 4.8 to conclude. □

Theorem 4.15 (Completeness of $\Rightarrow_\mu^*$) *Let $\mathcal{P} = \langle \Delta, \mathcal{X}, \delta, P\rangle$ be a valid quadruple and $\langle \nabla, \sigma\rangle \in \mathcal{M}_C(\mathcal{P})$. Then there exists a matching leaf $\mathcal{Q}$ such that $\mathcal{P} \Rightarrow_\mu^* \mathcal{Q}$ and $\langle \nabla, \sigma\rangle \in \mathcal{M}_C(\mathcal{Q})$.*

Proof. The formalisation follows by well-founded induction on the number of applications of $\Rightarrow_\mu$. Also, Lemma 4.10 is applied in the analysis of the cases where either $\mathcal{P}$ is a matching leaf or there exists Q' such that $\mathcal{P} \Rightarrow_\mu \mathcal{Q}'$. If $\mathcal{P}$ is a matching leaf then $\mathcal{P} = \mathcal{Q}$ and the proof is completed. If there exists Q' such that $\mathcal{P} \Rightarrow_\mu \mathcal{Q}'$, one applies Lemma 4.10 to obtain that $\mathcal{P}$ is not a matching leaf. Lemma 4.13 is applied to the premise that $\mathcal{P}$ is not a matching leaf. Form this and the hypothesis $\langle \nabla, \sigma\rangle \in \mathcal{M}_C(\mathcal{P})$ one obtains that there exists $\mathcal{Q}'$ such that $\mathcal{P} \Rightarrow_\mu \mathcal{Q}'$.

The IH is established as the following statement: $\forall \mathcal{R}$ valid, if $\mathcal{P} \Rightarrow_\mu \mathcal{R}$, $\mathtt{Rvar}(\mathcal{R}) \cap \mathtt{dom}(\sigma)$ and $\langle \nabla, \sigma\rangle \in \mathcal{M}_C(\mathcal{R})$, then there exists $\mathcal{S}$, such that $\mathcal{R} \Rightarrow_\mu^* \mathcal{S}$ and $\langle \nabla, \sigma\rangle \in \mathcal{M}_C(\mathcal{S})$.

This is applied to the hypothesis $\mathcal{P} \Rightarrow_\mu \mathcal{Q}'$ to conclude that there exists $\mathcal{Q}$, such that $\mathcal{Q}' \Rightarrow_\mu^* \mathcal{Q}$ and $\langle \nabla, \sigma\rangle \in \mathcal{M}_C(\mathcal{Q}')$. The other premises of IH are achieved with the auxiliary results given by Lemma 4.9 and Corollary 4.8. Finally, by case analysis on the statement $\mathcal{Q}' \Rightarrow_\mu^* \mathcal{Q}$, one concludes. □

Theorem 4.16 (Characterisation of successful matching leaves) *Let $\mathcal{Q} = \langle \Delta, \mathcal{X}, \delta, Q\rangle$ a matching leaf, if $\mathcal{Q}$ is a proper problem and there exists $\langle \nabla, \sigma\rangle \in \mathcal{M}_C(\mathcal{Q})$, then $Q = \emptyset$.*

Proof. First one proves the assertion that Q is a fixed point problem. This statement is proved using Theorem 3.12, Lemma 4.5 and rule (μ_v) of the definition of matching step (Figure 7). Therefore, if Q is a fixed problem it must be equal to the empty set, otherwise $\mathcal{Q}$ could be reduced by an application of rule $(\mu_{\mathbf{fp}})$ of Figure 7, which contradicts the fact that $\mathcal{Q}$ is a matching leaf. □

Example 4.17 [Nominal C-matching] This example is similar to Example 3.9, but now the set of protected variables is equal to the right-hand side variables of the initial problem, that is $\{X, Z\}$. This results in the execution of the nominal C-matching algorithm that provides the matching leaves

$$\langle \emptyset,\ \{X, Z\},\ \{\ Y/(a\,b).X\},\ \{X \approx_? \overline{b},\ (a\,b).Z \approx_? Z,\ a\#_? f\langle [a](\overline{a} * X), Z\rangle\}\rangle,$$

$$\text{and, } \langle \{a\#X, b\#X, a\#Z, b\#Z\},\ \{X, Z\},\ \{Y/\overline{b}\},\ \emptyset\rangle.$$

Since X is a protect variable, the former problem has no solution due the fact that the equation $X \approx_? \overline{b}$ cannot be solved since X cannot be instantiated. The latter problem has just one solution given by $\langle \{a\#X, b\#X, a\#Z, b\#Z\}, \{Y/\overline{b}\}\rangle$. Theorems 4.14 and 4.15 show that this solution is the unique C-matching solution for the initial problem.

$$\mathcal{P} = \langle \emptyset,\ \{X, Z\},\ id,\ \{[a]f\langle [b](X * Y), Z\rangle \approx [b]f\langle [a]((\overline{a} * X)), Z\rangle\}\rangle$$

$\Rightarrow_{(\approx_?[\mathbf{ab}])}$ $\langle\emptyset,\ \{X,Z\},\ id,\ \{f\langle[b](X*Y),Z\rangle \approx_? f\langle[b](\overline{b}*(a\,b).X),(a\,b).Z\rangle,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{app})}$ $\langle\emptyset,\ \{X,Z\},\ id,\ \{\langle[b](X*Y),Z\rangle \approx_? \langle[b](\overline{b}*(a\,b).X),(a\,b).Z\rangle,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{pair})}$ $\langle\emptyset,\ \{X,Z\},\ id,\ \{[b](X*Y) \approx_? [b](\overline{b}*(a\,b).X),\ Z \approx_? (a\,b).Z,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?[\mathbf{aa}])}$ $\langle\emptyset,\ \{X,Z\},\ id,\ \{X*Y \approx_? (\overline{b}*(a\,b).X),\ Z \approx_? (a\,b).Z,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$

(i)

$\Rightarrow_{(\approx_?\mathbf{C})}$ $\langle\emptyset,\ \{X,Z\},\ id,\ \{X \approx_? \overline{b},\ Y \approx_? (a\,b).X,\ Z \approx_? (a\,b).Z,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inst})}$ $\langle\emptyset,\ \{X,Z\},\ \{Y/(a\,b).X\},\ \{X \approx_? \overline{b},\ Z \approx_? (a\,b).Z,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle\emptyset,\ \{X,Z\},\ \{\ Y/(a\,b).X\},\ \{X \approx_? \overline{b},\ (a\,b).Z \approx_? Z,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$

(ii)

$\Rightarrow_{(\approx_?\mathbf{C})}$ $\langle\emptyset,\ \{X,Z\},\ id,\ \{X \approx_? (a\,b).X,\ Y \approx_? \overline{b},\ Z \approx_? (a\,b).Z,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle\emptyset,\ \{X,Z\},\ id,\ \{(a\,b).X \approx_? X,\ Y \approx_? \overline{b},\ Z \approx_? (a\,b).Z,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inst})}$ $\langle\emptyset,\ \{X,Z\},\ \{Y/\overline{b}\},\ \{(a\,b).X \approx_? X,\ Z \approx_? (a\,b).Z,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle\emptyset,\ \{X,Z\},\ \{Y/\overline{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$

$\Rightarrow_{(\#_?\mathbf{app})}$ $\langle\emptyset,\ \{X,Z\},\ \{Y/\overline{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z,\ a\#_?\langle[a](\overline{a}*X),Z\rangle\}\rangle$

$\Rightarrow_{(\#_?\mathbf{pair})}$ $\langle\emptyset,\ \{X,Z\},\ \{Y/\overline{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z,\ a\#_?[a](\overline{a}*X),\ a\#_?Z\}\rangle$

$\Rightarrow_{(\#_?\mathbf{a}[\mathbf{a}])}$ $\langle\emptyset,\ \{X,Z\},\ \{Y/\overline{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z,\ a\#_?Z\}\rangle$

$\Rightarrow_{(\#_?\mathbf{var})}$ $\langle\{a\#Z\},\ \{X,Z\},\ \{Y/\overline{b}\},\ \{(a\,b).X \approx_? X,\ (a\,b).Z \approx_? Z\}\rangle$

$\Rightarrow_{(\mu_{\mathbf{fp}})}$ $\langle\{a\#X,b\#X,a\#Z\},\ \{X,Z\},\ \{Y/\overline{b}\},\ \{(a\,b).Z \approx_? Z\}\rangle$

$\Rightarrow_{(\mu_{\mathbf{fp}})}$ $\langle\{a\#X,b\#X,a\#Z,b\#Z\},\ \{X,Z\},\ \{Y/\overline{b}\},\ \emptyset\rangle$

Example 4.18 [Nominal C-equivalence checking] This example exhibits the execution of the nominal C-unification algorithm applied to nominal C-equivalence check. In item (a), the set of protected variables, $\{X,Y,Z\}$, consists now of all variables in the input problem. The algorithm generates two leaves

$$\langle\emptyset,\ \{X,Y,Z\},\ id,\ \{X \approx_? \overline{b},\ Y \approx_? (a\,b).X,(a\,b).Z \approx_? Z,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle$$

and

$$\langle\emptyset,\ \{X,Y,Z\},\ id,\ \{(a\,b).X \approx_? X,\ Y \approx_? \overline{b},\ (a\,b).Z \approx_? Z,\ a\#_?f\langle[a](\overline{a}*X),Z\rangle\}\rangle.$$

Both are quadruples that have equations without solutions. In the former one, the X cannot be instantiated to solve $X \approx_? \overline{b}$, and in the latter one, Y cannot be instantiated to solve $Y \approx_? \overline{b}$.

In item (b), the set of protected variables, $\{X, Y\}$, consists also of all variables in the input problem, but the generated leaves are

$$\langle \emptyset,\ \{X, Y\},\ id,\ \{X \approx_? \overline{b},\ \overline{b} \approx_? (a\,b).X, (a\,b).Y \approx_? Y,\ a\#_? f\langle [a](\overline{a} * X), Y\rangle\}\rangle$$

$$\text{and, } \langle \{a\#X, b\#X, a\#Y, b\#Y\},\ \{X, Y\},\ id,\ \emptyset\rangle$$

The first leaf has also equations with the protected variable X. Namely, in equations $X \approx_? \overline{b}$ and $\overline{b} \approx_? (a\,b).X$ X cannot be instantiated. Thus, neither equation has solutions. On the other branch, the second leaf provides a solution given by the freshness context $\{a\#X, b\#X, a\#Y, b\#Y\}$.

(a) $\langle \emptyset,\ \{X, Y, Z\},\ id,\ \{[a]f\langle [b](X * Y), Z\rangle \approx [b]f\langle [a](\overline{a} * X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?[\mathbf{ab}])}$ $\langle \emptyset,\ \{X, Y, Z\},\ id,\ \{f\langle [b](X * Y), Z\rangle \approx_? f\langle [b](\overline{b} * (a\,b).X), (a\,b).Z\rangle,\ a\#_? f\langle [a](\overline{a} * X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{app})}$ $\langle \emptyset,\ \{X, Y, Z\},\ id,\ \{\langle [b](X * Y), Z\rangle \approx_? \langle [b](\overline{b} * (a\,b).X), (a\,b).Z\rangle,\ a\#_? f\langle [a](\overline{a} * X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{pair})}$ $\langle \emptyset,\ \{X, Y, Z\},\ id,\ \{[b](X * Y) \approx_? [b](\overline{b} * (a\,b).X),\ Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a} * X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?[\mathbf{aa}])}$ $\langle \emptyset,\ \{X, Y, Z\},\ id,\ \{X * Y \approx_? (\overline{b} * (a\,b).X),\ Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a} * X), Z\rangle\}\rangle$

(i)

$\Rightarrow_{(\approx_?\mathbf{C})}$ $\langle \emptyset,\ \{X, Y, Z\},\ id,\ \{X \approx_? \overline{b},\ Y \approx_? (a\,b).X,\ Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a} * X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle \emptyset,\ \{X, Y, Z\},\ id,\ \{X \approx_? \overline{b},\ Y \approx_? (a\,b).X, (a\,b).Z \approx_? Z,\ a\#_? f\langle [a](\overline{a} * X), Z\rangle\}\rangle$

(ii)

$\Rightarrow_{(\approx_?\mathbf{C})}$ $\langle \emptyset,\ \{X, Y, Z\},\ id,\ \{X \approx_? (a\,b).X,\ Y \approx_? \overline{b},\ Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a} * X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle \emptyset,\ \{X, Y, Z\},\ id,\ \{(a\,b).X \approx_? X,\ Y \approx_? \overline{b},\ Z \approx_? (a\,b).Z,\ a\#_? f\langle [a](\overline{a} * X), Z\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle \emptyset,\ \{X, Y, Z\},\ id,\ \{(a\,b).X \approx_? X,\ Y \approx_? \overline{b},\ (a\,b).Z \approx_? Z,\ a\#_? f\langle [a](\overline{a} * X), Z\rangle\}\rangle$

(b) $\langle \emptyset,\ \{X, Y\},\ id,\ \{[a]f\langle [b](X * \overline{b}), Y\rangle \approx [b]f\langle [a](\overline{a} * X), Y\rangle\}\rangle$

$\Rightarrow_{(\approx_?[\mathbf{ab}])}$ $\langle \emptyset,\ \{X, Y\},\ id,\ \{f\langle [b](X * \overline{b}), Y\rangle \approx_? f\langle [b](\overline{b} * (a\,b).X), (a\,b).Y\rangle,\ a\#_? f\langle [a](\overline{a} * X), Y\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{app})}$ $\langle \emptyset,\ \{X, Y\},\ id,\ \{\langle [b](X * \overline{b}), Y\rangle \approx_? \langle [b](\overline{b} *$

$(a\,b).X), (a\,b).Y\rangle,\ a\#_? f\langle [a](\overline{a} * X), Y\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{pair})}$ $\langle\emptyset,\ \{X, Y\},\ id,\ \{[b](X * \overline{b}) \approx_? [b](\overline{b} * (a\,b).X),\ Y \approx_? (a\,b).Y,\ a\#_? f\langle [a](\overline{a} * X), Y\rangle\}\rangle$

$\Rightarrow_{(\approx_?[\mathbf{aa}])}$ $\langle\emptyset,\ \{X, Y\},\ id,\ \{X * \overline{b} \approx_? (\overline{b} * (a\,b).X),\ Y \approx_? (a\,b).Y,\ a\#_? f\langle [a](\overline{a} * X), Y\rangle\}\rangle$

(i)

$\Rightarrow_{(\approx_?\mathbf{C})}$ $\langle\emptyset,\ \{X, Y\},\ id,\ \{X \approx_? \overline{b},\ \overline{b} \approx_? (a\,b).X,\ Y \approx_? (a\,b).Y,\ a\#_? f\langle [a](\overline{a} * X), Y\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle\emptyset,\ \{X, Y\},\ id,\ \{X \approx_? \overline{b},\ \overline{b} \approx_? (a\,b).X, (a\,b).Y \approx_? Y,\ a\#_? f\langle [a](\overline{a} * X), Y\rangle\}\rangle$

(ii)

$\Rightarrow_{(\approx_?\mathbf{C})}$ $\langle\emptyset,\ \{X, Y\},\ id,\ \{X \approx_? (a\,b).X,\ \overline{b} \approx_? \overline{b},\ Y \approx_? (a\,b).Y,\ a\#_? f\langle [a](\overline{a} * X), Y\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle\emptyset,\ \{X, Y\},\ id,\ \{(a\,b).X \approx_? X,\ \overline{b} \approx_? \overline{b},\ Y \approx_? (a\,b).Y,\ a\#_? f\langle [a](\overline{a} * X), Y\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{refl})}$ $\langle\emptyset,\ \{X, Y\},\ id,\ \{(a\,b).X \approx_? X,\ Y \approx_? (a\,b).Y,\ a\#_? f\langle [a](\overline{a} * X), Y\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{inv})}$ $\langle\emptyset,\ \{X, Y\},\ id,\ \{(a\,b).X \approx_? X,\ (a\,b).Y \approx_? Y,\ a\#_? f\langle [a](\overline{a} * X), Y\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{app})}$ $\langle\emptyset,\ \{X, Y\},\ id,\ \{(a\,b).X \approx_? X,\ (a\,b).Y \approx_? Y,\ a\#_? \langle [a](\overline{a} * X), Y\rangle\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{pair})}$ $\langle\emptyset,\ \{X, Y\},\ id,\ \{(a\,b).X \approx_? X,\ (a\,b).Y \approx_? Y,\ a\#_? [a](\overline{a} * X),\ a\#_? Y\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{a}[\mathbf{a}])}$ $\langle\emptyset,\ \{X, Y\},\ id,\ \{(a\,b).X \approx_? X,\ (a\,b).Y \approx_? Y,\ a\#_? Y\}\rangle$

$\Rightarrow_{(\approx_?\mathbf{var})}$ $\langle\{a\#Y\},\ \{X, Y\},\ id,\ \{(a\,b).X \approx_? X,\ (a\,b).Y \approx_? Y\}\rangle$

$\Rightarrow_{(\mu_{\mathbf{fp}})}$ $\langle\{a\#X, b\#X, a\#Y\},\ \{X, Y\},\ id,\ \{(a\,b).Y \approx_? Y\}\rangle$

$\Rightarrow_{(\mu_{\mathbf{fp}})}$ $\langle\{a\#X, b\#X, a\#Y, b\#Y\},\ \{X, Y\},\ id,\ \emptyset\rangle$

5 Conclusion and future work

This paper presents an extension of the nominal C-unification algorithm proposed in [2], which permits the use of *protected variables*. When the set of protected variables is the set of the variables in the right-hand side of nominal equational problems given as input, the algorithm outputs a nominal C-matcher for the input problem if one exists. If all the variables of a nominal unification problem are protected, the algorithm becomes a nominal C-equality checker.

The nominal C-matching algorithm was checked through a formalisation in Coq which reused a formalisation of the unification algorithm in [2] plus additional formalisations related with the main desired properties of the C-matching algorithm that are termination, soundness and completeness.

Work in progress investigates formalisations of nominal AC-unification and matching, and future work will deal with restricted cases such as linear AC-matching, as well as unification modulo more general theories.

References

[1] 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.

[2] M. Ayala-Rincón, W. Carvalho-Segundo, M. Fernández, and D. Nantes-Sobrinho. *Nominal C-Unification*. In *Post-proc. of the 27th Int. Symp. Logic-based Program Synthesis and Transformation (LOPSTR 2017)*, volume 10855 of *LNCS*, pages 235–251. Springer, 2018.

[3] M. Ayala-Rincón, W. Carvalho-Segundo, M. Fernández, and D. Nantes-Sobrinho. A Formalisation of Nominal α-equivalence with A, C, and AC Function Symbols. *Theor. Comput. Sci.*, Accepted 2019.

[4] M. Ayala-Rincón, W. de Carvalho Segundo, M. Fernández, and D. Nantes-Sobrinho. A formalisation of nominal α-equivalence with A and AC function symbols. *ENTCS*, 332:21–38, 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 D. Nantes-Sobrinho. *Fixed Point Constraints for Nominal Equational Unfication*. In *Proc. of the 3rd Int. Conf. Formal Structures for Computation and Deduction (FSCD)*, volume 108 of *LIPIcs*, pages 7:1–7:16, 2018.

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

[8] F. Baader. The Theory of Idempotent Semigroups is of Unification Type Zero. *J. of Autom. Reasoning*, 2(3):283–286, 1986.

[9] F. Baader and Klaus U. Schulz. Unification in the Union of Disjoint Equational Theories: Combining Decision Procedures. *J. of Sym. Computation*, 21(2):211–243, 1996.

[10] F. Baader and W. Snyder. Unification Theory. In *Handbook of Automated Reasoning (in 2 volumes)*, pages 445–532. Elsevier and MIT Press, 2001.

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

[12] 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.

[13] 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.

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

[15] D. Kapur and P. Narendran. NP-Completeness of the Set Unification and Matching Problems. In *8th International Conference on Automated Deduction (CADE)*, volume 230 of *LNCS*, pages 489–495. Springer, 1986.

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

[17] D. Kapur and P. Narendran. Complexity of Unification Problems with Associative-Commutative Operators. *J. of Autom. Reasoning*, 9(2):261–288, 1992.

[18] 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.

[19] A. M. Pitts. *Nominal Sets*. Number 57 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2013.

[20] M. Schmidt-Schauß, T. Kutsia, J. Levy, and M. Villaret. *Nominal Unification of Higher Order Expressions with Recursive Let*. In *Post-proc. of the 26th Int. Sym. on Logic-Based Program Synthesis and Transformation (LOPSTR 2016)*, volume 10184 of *LNCS*, pages 328–344. Springer, 2017.

[21] J. H. Siekmann. Matching under commutativity. In *Proc. of the Int. Symposium on Symbolic and Algebraic Manipulation (EUROSAM)*, volume 72 of *LNCS*, pages 531–545. Springer, 1979.

[22] J. H. Siekmann. Unification Theory. *J. of Sym. Computation*, 7(3-4):207–274, 1989.

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

[24] C. Urban, A. M. Pitts, and M. J. Gabbay. *Nominal Unification*. *Theor. Comput. Sci.*, 323(1-3):473–497, 2004.