Nominal Narrowing

Mauricio Ayala-Rincón, Maribel Fernández, and Daniele Nantes-Sobrinho

Abstract
Nominal unification is a generalisation of first-order unification that takes \(\alpha\)-equivalence into account. In this paper, we study nominal unification in the context of equational theories. We introduce nominal narrowing and design a general nominal E-unification procedure, which is sound and complete for a wide class of equational theories. We give examples of application.

1 Introduction
This is a paper about nominal unification in the context of equational theories.

Nominal techniques [16] facilitate reasoning in systems with binding operators, where \(\alpha\)-equivalence must be taken into account. In nominal syntax [11, 29], atoms, which are used to represent object-level variables in the intended applications, can be abstracted: \([a]t\) denotes the abstraction of the atom \(a\) in the term \(t\). Variables in nominal terms represent unknown parts of terms and behave like first-order variables, but nominal variables may be decorated with atom permutations. Permutations act on terms, swapping atoms (e.g., \((a\ b)\cdot t\) means that \(a\) and \(b\) are swapped everywhere in \(t\)).

Nominal syntax has interesting properties. Nominal unification [29], that is, unification of nominal terms modulo \(\alpha\)-equivalence, is decidable and unitary. Efficient nominal unification algorithms are available [3, 19]. Nominal matching, a key ingredient in the definition of nominal rewriting [11], is a particular case of nominal unification that can be solved in linear time [4]. Nominal rewriting [11] can be used to reason in nominal equational theories (see [12]; a completion procedure is described in [14]).

However, to our knowledge, the concept of nominal E-unification, i.e., nominal unification in the context of an equational theory \(E\), has not been addressed in previous works. Nominal E-unification is needed to solve equations between nominal terms where the function symbols...
satisfy properties defined by an equational theory. Nominal E-unification has applications in, e.g., functional-logical programming languages and analysis of cryptographic protocols.

The main contributions of this paper are:
- We define nominal E-unification problems, and the nominal narrowing relation, and study the relationship between nominal rewriting and nominal narrowing.
- We show that Hullot’s results [17] (with the corrections from [1, 24]) relating first-order narrowing derivations and first-order E-unifiers can be transferred to nominal systems. Thus, we obtain a nominal E-unification procedure that is sound and complete for the class of convergent closed equational theories. We give examples to illustrate these results.
- We define basic nominal narrowing and provide sufficient conditions for termination of nominal narrowing derivations, which can be used to prove the decidability of nominal E-unification for certain equational theories.

Related Work. Narrowing has traditionally been used to solve equations in initial and free algebras modulo a set of equations. It is well-known that narrowing is a programming feature that allows integration of functional and logical programming languages [8, 20]. Narrowing was originally introduced for theorem proving [17], but nowadays it is used in type inference [27] and verification of cryptographic protocols [23], amongst other areas. Narrowing gives rise to a complete E-unification procedure if E is defined by a convergent rewrite system, but it is generally inefficient. Several strategies have been designed to make narrowing-based E-unification procedures more efficient by reducing the search space (e.g., basic narrowing [17] and variant narrowing [9], the latter inspired by the notion of E-variant [6]) and sufficient conditions for termination have been obtained [17, 9, 1]. In this paper we develop basic nominal narrowing strategies and associated termination conditions, and leave the study of other complete strategies for future work.

Nominal unification is closely related to higher-order pattern unification [18] and there is previous work addressing higher-order pattern E-unification: Prehofer [26] introduced higher-order narrowing and some variants (such as lazy narrowing, conditional narrowing, pattern narrowing), and considered applications of narrowing as an inference rule in logic and functional programming. Nominal extensions of logic and functional programming languages are already available (see, e.g., [28, 5]), and nominal narrowing could play a similar role in the definition of a functional-logic programming language.

Overview of the paper: Section 2 recalls basic concepts in nominal unification and rewriting. Section 3 introduces the notion of nominal narrowing, presents results relating nominal narrowing and nominal equational unification, and gives examples of application. Section 4 introduces basic nominal narrowing and the results regarding the termination of narrowing. Section 5 contains the conclusions and directions for future work.

2 Nominal Rewriting

We recall below the definitions of nominal unification and nominal rewriting; for more details we refer the reader to [11, 29].

2.1 Nominal terms and \(\alpha\)-equivalence

A nominal signature \(\Sigma\) is a set of function symbols \(f, g, \ldots\), each with a fixed arity \(n \geq 0\). Fix a countably infinite set \(X\) of variables \(X, Y, Z, \ldots\); these represent meta-level unknowns. Also,
fix a countably infinite set \( \mathcal{A} \) of atoms \( a, b, c, n, x, \ldots \); these represent object-level variables. We assume that \( \Sigma, \mathcal{X} \) and \( \mathcal{A} \) are pairwise disjoint.

Nominal terms are generated by the grammar: \( t ::= a \mid \pi \cdot X \mid [a]t \mid f(t_1, \ldots, t_n) \).

Terms are called respectively atoms, suspensions, abstractions and function applications. We write \( V(t) \) for the set of variables occurring in \( t \), \( A(t) \) for the set of atoms mentioned in \( t \), and \( atm(t) \) for the set of atoms that occur as subterms in \( t \). For example, \( A([a]b) = \{ a, b \} \), \( b \in atm([a]b) \), \( a \notin atm([a]b) \). Ground terms are terms without variables, they may still contain atoms. The occurrences of \( a \) in a term are said to be bound (or abstracted) if they occur in the scope of an abstraction, otherwise they are said to be free (or unabstracted).

A permutation \( \pi \) is a bijection on atoms, with finite domain. \( \pi \cdot \pi' \) denotes functional composition of permutations and \( \pi^{-1} \) denotes the inverse of \( \pi \). A permutation action \( \pi \cdot t \) is defined by induction: \( \pi \cdot \alpha \equiv \pi(\alpha) \), \( \pi \cdot [a]t \equiv [\pi(a)](\pi \cdot t) \), \( \pi \cdot (\pi' \cdot X) \equiv (\pi \circ \pi') \cdot X \) and \( \pi \cdot f(t_1, \ldots, t_n) \equiv f(\pi \cdot t_1, \ldots, \pi \cdot t_n) \). We write \( \langle a \rangle b \) for the swapping permutation that maps \( a \) to \( b \), \( b \) to \( a \) and all other atoms \( c \) to themselves, and \( Id \) for the identity permutation, so \( Id(a) = a \). Note that \( X \) is not a term, but \( Id \cdot X \) is. We abbreviate \( Id \cdot X \) as \( X \) when there is no ambiguity.

A substitution \( \sigma \) is a mapping from variables to terms, with a finite domain denoted by \( \text{dom}(\sigma) \); the image is denoted \( \text{Im}(\sigma) \). Henceforth, if \( X \notin \text{dom}(\sigma) \) then \( \sigma(X) \) denotes \( Id \cdot X \). Substitutions are generated by the grammar: \( \sigma ::= \text{Id} \mid \{ X \mapsto V \} \sigma \), where \( Id \) denotes the substitution with \( \text{dom}(Id) = \emptyset \). We use the same notation for the identity permutation and the identity substitution, as there will be no ambiguity. For every substitution \( \sigma \), we define \( \sigma|_V \) (the restriction of \( \sigma \) to \( V \)) as the substitution that maps \( X \) to \( \sigma(X) \) if \( X \in V \) and to \( Id \cdot X \) otherwise. The substitution action \( \sigma t \) is defined as follows: \( a\sigma \equiv a \), \( ([a]t)\sigma \equiv [a](\sigma(t)) \), \( f(t_1, \ldots, t_n)\sigma \equiv f(\sigma(t_1), \ldots, \sigma(t_n)) \) and \( (\pi \cdot X)\sigma \equiv \pi \cdot \sigma(X) \). If \( \sigma \) and \( \theta \) are substitutions, \( \theta \circ \sigma \) is the substitution that maps each \( X \) to \( (X\sigma)\theta \). Note that substitution allows capture of free atoms (it behaves like first-order substitution, except that when instantiating \( \pi \cdot X \), \( \pi \) applies).

On nominal terms, \( \alpha \)-equivalence is defined using swappings and a notion of freshness. A freshness constraint is a pair \( a \# t \) (read “\( a \) fresh in \( t \)”) of an atom \( a \) and a term \( t \). Intuitively, \( a \# t \) means that if \( a \) occurs in \( t \) then it must be abstracted. An \( \alpha \)-equality constraint is a pair \( s \approx_{\alpha} t \) of two terms \( s \) and \( t \). A freshness context is a set of freshness constraints of the form \( a \# t \). \( \Delta, \Gamma \) and \( \nabla \) will range over freshness contexts. A freshness judgement is a tuple of the form \( \Delta \vdash a \# t \) whereas an \( \alpha \)-equivalence judgement is a tuple of the form \( \Delta \vdash s \approx_{\alpha} t \). The derivable freshness and \( \alpha \)-equivalence judgements are defined by the rules in Figure 1. A set \( Pr \) of constraints is called a problem. We write \( \Delta \vdash Pr \) when proofs exist for each \( P \in Pr \), using the derivation rules given in Figure 1. The minimal \( \Delta \) such that \( \Delta \vdash Pr \), denoted by \( \langle Pr \rangle_{nf} \), can be obtained by using a system of simplification rules [11, 29], which, given \( Pr \), outputs \( \Delta \) or fails.

### 2.2 Unification, Matching and Nominal Rewriting

Unification is about finding a substitution that makes two terms equal. For nominal terms the notion of equality is \( \approx_{\alpha} \), which is defined in a freshness context; nominal unification takes this into account.

**Definition 1.** A solution for a problem \( Pr \) is a pair \( (\Gamma, \sigma) \) such that \( \Gamma \vdash Pr \sigma \), where \( Pr \sigma \) is the problem obtained by applying the substitution \( \sigma \) to the terms in \( Pr \).

We follow [11], defining nominal matching/unification problems in context. A term-in-context is a pair \( \Delta \vdash t \) of a freshness context and a term. We may write \( \vdash t \) or simply \( t \) if \( \Delta = \emptyset \).
The action of substitutions extends to freshness contexts, instantiating the variables in freshness constraints.

Definition 2. A unification problem (in context) is a pair \((\nabla \vdash l) \approx s\) where \(\nabla, \nabla\) are freshness contexts and \(l, s\) are nominal terms. The solution to this unification problem, if it exists, is a pair \((\Delta', \theta)\) that solves the problem \(\nabla, \nabla, l \approx s\), that is, \(\Delta' \vdash \Delta, \theta, \nabla, l \approx_s s\).

A matching problem (in context) is a particular kind of unification problem, written \((\nabla \vdash l) \approx_s (\Delta \vdash s)\) where \(s\) is ground, or contains variables not occurring in \(\nabla, l\). The solution \((\Delta', \theta)\) is such that \(X\theta \equiv X\) for \(X \in V(\Delta, s)\) (i.e., \(\theta\) can only instantiate variables in \(\nabla, l\), therefore, \(\Delta' \vdash \Delta, \nabla\theta\) and \(\Delta' \vdash t \approx_s s\).

Example 3. \((\vdash [a][b]X') \approx (\vdash [b][a]X)\) has solution \((\emptyset, \{X' \mapsto (a b) \cdot X\})\).

Definition 4. Let \(\Gamma_1, \Gamma_2\) be contexts, and \(\sigma_1, \sigma_2\) substitutions. Then \((\Gamma_1, \sigma_1) \leq (\Gamma_2, \sigma_2)\) if there exists some \(\sigma'\) such that: \(\forall X, \Gamma_2 \vdash X\sigma_1 \sigma' \approx_{\sigma_1} X\sigma_2\) and \(\Gamma_2 \vdash \Gamma_1\sigma'\). If we want to be more specific, we may write \((\Gamma_1, \sigma_1) \leq_{\sigma'} (\Gamma_2, \sigma_2)\). The relation \(\leq\) is a partial order.

Nominal unification is decidable and unitary [29]: a solvable problem has a unique least solution according to \(\leq\), called principal solution or most general unifier, denoted by \(mgu(Pr)\).

Below we recall the definitions of nominal equational reasoning [15] and nominal rewriting [11] from [12], where a position \(C\) is defined as a pair \((s, \_\_\_\) of a term and a distinguished variable \(\_\_\_\in X\) that occurs precisely once in \(s\), with permutation \(Id\). \(C\) is also called a context. When there is no ambiguity, we equate \(C\) with \(s\) and write \(C[t]\) for the result of applying the substitution \(\{\_\_\_\_r \mapsto t\}\) to \(s\). \(Pos(u)\) denotes the set of positions of the nominal term \(u\), that is, all the positions \(C\) such that \(u = C[t]\) for some \(t\). \(\overline{Pos}(u) = \{C \in Pos(u) | u = C[t] \text{ and } t \neq \pi \cdot X\}\) is the set of non-variable positions.

An equality judgement (resp. rewrite judgement) is a tuple \(\Delta \vdash s = t\) (resp. \(\Delta \vdash s \rightarrow t\)) of a freshness context \(\Delta\) and two nominal terms \(s, t\). An equational theory \(\mathbb{E} = (\Sigma, Ax)\) is a pair of a signature \(\Sigma\) and a possibly infinite set of equality judgements \(Ax\) in \(\Sigma\); they are called axioms. A rewrite theory \(\mathbb{R} = (\Sigma, Rw)\) is a pair of a signature \(\Sigma\) and a possibly infinite set of rewrite judgements \(Rw\) in \(\Sigma\); they are called rewrite rules. \(\Sigma\) may be omitted, identifying \(\mathbb{E}\)

---

1 The \(\approx\) indicates that the variables being instantiated occur in the left-hand side term.
2 This definition of position is equivalent to the standard notion of a position as a path in a tree; here we exploit the fact that nominal substitution corresponds to the informal notion of replacement of a ‘hole’ in a context by a term.
with $Ax$ and $Rw$ when the signature is clear from the context. See Figure 2 for an example of a rewrite theory for the $\lambda$-calculus.

**Definition 5.**

- Nominal rewriting: The one-step rewrite relation $\Delta \vdash s \xrightarrow{R, R, \theta, \pi} t$ is the least relation such that for any $R = (\nabla \vdash l \rightarrow r) \in R$, position $C$, term $s'$, permutation $\pi$, and substitution $\theta$,

\[
\begin{align*}
C[s'] & \vdash (\nabla \theta, s', \approx_{\alpha} \pi \cdot (\theta), C[\pi \cdot (r \theta)] \approx_{\alpha} t) \\
\Delta \vdash s & \xrightarrow{R, R, \theta, \pi} t
\end{align*}
\]

- Nominal algebra) equality: $\Delta \vdash s = t$ is the least transitive reflexive symmetric relation such that for any $(\nabla \vdash l = r) \in E$, position $C$, permutation $\pi$, substitution $\theta$, and fresh $\Gamma$ (so if $a \# X \in \Gamma$ then $a$ is not mentioned in $\Delta, s, t$),

\[
\begin{align*}
\Delta, \Gamma & \vdash (\nabla \theta, s \approx_{\alpha} C[\pi \cdot (l \theta)], C[\pi \cdot (r \theta)] \approx_{\alpha} t) \\
\Delta \vdash s & = t
\end{align*}
\]

Given an equational theory $E$ and a rewrite theory $R$, we say that $R$ is a presentation of $E$ if: $\nabla \vdash s = t \in E \iff (\nabla \vdash s \rightarrow t \in R \lor \nabla \vdash t \rightarrow s \in R)$.

Nominal rewriting is not complete for equational reasoning in general; however, closed nominal rewriting is complete for equational reasoning with closed axioms (see [12]). Intuitively, no free atom occurs in a closed term, and closed axioms do not allow abstracted atoms to become free (a natural assumption). Closedness of a term can be easily checked by matching the term with a freshened copy of itself. For example, the term $f(a)$ is not closed (it is not possible to match $f(a)$ with a freshened variant $f(a')$); however, $f([a][a])$ is closed ($f([a][a]) \approx_{\alpha} f([a'][a'])$). If there are variables, freshness contexts have to be taken into account. We recall below the definitions of freshened variant, closed rewrite rule and closed rewriting relation from [12].

\[\vdash \text{app}(\text{lam}([a][X]), X') \rightarrow \text{sub}([a][X], X') \quad (\text{Beta})\]
\[\begin{align*}
\vdash \text{sub}([a][a], X) & \rightarrow X \\
\vdash \text{sub}([a][Y], X) & \rightarrow Y \\
\vdash \text{sub}([a][\text{app}(X,X'), Y]) & \rightarrow \text{app}([\text{sub}([a][X]), \text{sub}([a][X'), Y)]) \\
b \# Y & \vdash \text{sub}([a][\text{lam}(b)[X]), Y') \rightarrow \text{lam}([b][\text{sub}([a][X], Y)])
\end{align*}\]
11:6 Nominal Narrowing

If \( t \) is a term, we say that \( t' \) is a \textit{freshened variant} of \( t \) when \( t' \) has the same structure as \( t \), except that the atoms and unknowns have been replaced by ‘fresh’ atoms and unknowns. Similarly, if \( \nabla \) is a freshness context then \( \nabla' \) will denote a freshened variant of \( \nabla \) (so if \( a\#X \in \nabla \) then \( a'\#X' \in \nabla' \), where \( a' \) and \( X' \) are chosen fresh for the atoms and unknowns appearing in \( \nabla \)). We may extend this to other syntax, like equality and rewrite judgements. For example, \([a][b]X\) is a freshened variant of \([a][b]X\), \( a'\#X' \) is a freshened variant of \( a\#X \), and \( \emptyset \vdash f([a][b]X) \rightarrow [a']X' \) is a freshened variant of \( \emptyset \vdash f([a]X) \rightarrow [a]X \).

\[ \nabla \vdash t \] is \textit{closed} if there exists a solution for the matching problem \( (\nabla \vdash l') \approx (\nabla, A(\nabla', l') \# V(\nabla, l) \vdash l) \).

Call \( R = (\nabla \vdash l \to r) \) and \( \Delta x = (\nabla \vdash l = r) \) \textit{closed} when \( \nabla \vdash (l, r) \) is closed\(^5\). Given a rewrite rule \( R = (\nabla \vdash l \to r) \) and a term-in-context \( \Delta \vdash s \), write \( \Delta \vdash s \rightarrow^*_R t \) when there is some \( R^e \) a freshened variant of \( R \) (so fresh for \( R, \Delta, s, \) and \( t \)), position \( C \) and substitution \( \theta \) such that \( s \equiv C[s'] \) and \( \Delta, A(R^e) \# V(\Delta, s, t) \vdash (\nabla \theta, s' \approx \alpha \theta, C[r^e \theta] \approx \alpha t) \). We call this (one-step) \textit{closed rewriting}. The \textit{closed-rewrite relation} \( \Delta \vdash s \rightarrow^*_R t \) is the reflexive transitive closure as in Definition 5.

All the rewrite rules in Figure 2 are closed. Closed rewriting is an efficient mechanism to generate rewriting steps for closed rules (closed-rewriting steps can be generated simply using nominal matching; it is not necessary to find a permutation \( \pi \) to apply a rule). We refer the reader to [11, 12] for examples.

3 Nominal E-Unification and Narrowing

We start by generalising the notion of solution.

\[ \nabla \vdash t \] is \textit{closed} if there exists a solution for the matching problem \( (\nabla \vdash l') \approx (\nabla, A(\nabla', l') \# V(\nabla, l) \vdash l) \).

Call \( R = (\nabla \vdash l \to r) \) and \( \Delta x = (\nabla \vdash l = r) \) \textit{closed} when \( \nabla \vdash (l, r) \) is closed\(^5\). Given a rewrite rule \( R = (\nabla \vdash l \to r) \) and a term-in-context \( \Delta \vdash s \), write \( \Delta \vdash s \rightarrow^*_R t \) when there is some \( R^e \) a freshened variant of \( R \) (so fresh for \( R, \Delta, s, \) and \( t \)), position \( C \) and substitution \( \theta \) such that \( s \equiv C[s'] \) and \( \Delta, A(R^e) \# V(\Delta, s, t) \vdash (\nabla \theta, s' \approx \alpha \theta, C[r^e \theta] \approx \alpha t) \). We call this (one-step) \textit{closed rewriting}. The \textit{closed-rewrite relation} \( \Delta \vdash s \rightarrow^*_R t \) is the reflexive transitive closure as in Definition 5.

All the rewrite rules in Figure 2 are closed. Closed rewriting is an efficient mechanism to generate rewriting steps for closed rules (closed-rewriting steps can be generated simply using nominal matching; it is not necessary to find a permutation \( \pi \) to apply a rule). We refer the reader to [11, 12] for examples.

3 Nominal E-Unification and Narrowing

We start by generalising the notion of solution.

\textbf{Definition 7 (Nominal E-unification).} An \textit{E-solution}, or \textit{E-unifier}, of a problem \( Pr \) is a pair \((\Gamma, \sigma)\) of a freshness context and a substitution such that

1. \( \Gamma \vdash_E Pr \sigma \) where \( Pr' \) is obtained from \( Pr \) by replacing each \( \approx \) by \( = \), and \( \Gamma \vdash_E a\#t \) coincides with \( \Gamma \vdash a\#t \).
2. \( X\sigma = X\sigma \) for all \( X \) (i.e., \( \sigma \) is idempotent).

If there is no such \((\Gamma, \sigma)\) then \( Pr \) is \textit{unsolvable}. \( U_E(Pr) \) is the \textit{set of E-solutions} of \( Pr \).

The notion of \( E \)-unification extends to terms-in-context in the natural way.

\textbf{Definition 8.} A \textit{nominal E-unification problem (in context)} is a pair \((\nabla \vdash l) \vdash_E (\Delta \vdash s) \).

The pair \((\Delta', \sigma)\) is an \textit{E-solution}, or \textit{E-unifier}, of \((\nabla \vdash l) \vdash_E (\Delta \vdash s) \) if \((\Delta', \sigma)\) is an \textit{E-solution} of the problem \( \nabla, \Delta, l \approx s \), that is, \( \Delta' \vdash_E \nabla \sigma, \Delta \sigma, l \sigma = s \sigma \).

\( U_E(\nabla \vdash l, \Delta \vdash s) \) denotes the set of all the \textit{E-solutions} of \((\nabla \vdash l) \vdash_E (\Delta \vdash s) \). If \( \nabla \) and \( \Delta \) are empty we write \( U_E(l, s) \) for the set of \textit{E-unifiers} of \( l \) and \( s \).

\textit{Nominal E-matching problems in context} are defined similarly, except that \( s \) is a ground term (or, if it has variables, the solution cannot instantiate them). \textit{E-matching problems in context} are written \((\nabla \vdash l) \vdash_E (\Delta \vdash s) \).

\textbf{Definition 9.} The ordering \( \leq_E \) is the extension of \( \leq \) with respect to \( E \): \( (\Gamma_1, \sigma_1) \leq_E (\Gamma_2, \sigma_2) \) iff there exists a substitution \( \rho \) such that \( \forall X, \Gamma_2 \vdash_E X\sigma_2 = (X\sigma_1)\rho \) and \( \Gamma_2 \vdash E \Gamma_1 \rho \). We write \( \leq_E^{V} \) for the restriction of \( \leq_E \) to the set \( V \) of variables.

---

\(^{4}\) \( A(\nabla', l') \# V(\nabla, l) = \{a\#X \mid a \in A(\nabla', l'), X \in V(\nabla, l)\} \).

\(^{5}\) Here we use the pair constructor as a term former and apply the definition above.
Definition 10 (Complete set of E-solutions of Pr). Let \( W \) be a finite set of variables containing \( V = V(Pr) \). We say that \( S = \{(\Gamma_1, \theta_1), \ldots, (\Gamma_n, \theta_n)\} \) is a complete set of E-solutions of Pr of \( W \) if:

1. \( \forall (\Gamma, \theta) \in S, \text{dom}(\theta) \subseteq V \) and \( \text{Im}(\theta) \cap W = \emptyset \),
2. \( S \subseteq \mathcal{U}_{E}(Pr) \) (correctness),
3. \( \forall (\Gamma, \sigma) \in \mathcal{U}_{E}(Pr) \) \( \exists (\Gamma_i, \theta_i) \in S, (\Gamma_i, \theta_i) \preceq^E (\Gamma, \sigma) \) (completeness).

We are now ready to define the nominal narrowing relation generated by \( R \). The definition of nominal narrowing is similar to nominal rewriting, but we need to solve unification problems instead of matching problems.

Definition 11 (Nominal Narrowing). The one-step narrowing relation \( (\Delta \vdash s) \sim_{R} (\Delta' \vdash t) \) is the least relation such that for any \( R = (\nabla \vdash l \rightarrow r) \in R \), position \( C \), term \( s' \), permutation \( \pi \), and substitution \( \theta \),

\[
\begin{align*}
\text{if } & \exists C[s'], \Delta', \theta, s'\theta \approx_{\alpha} \pi \cdot (\theta s) \text{ and } (C[\pi \cdot r]\theta \approx_{\alpha} t) \\
& \text{then } (\Delta \vdash s) \sim_{\{C, \pi, \theta, \pi\}} (\Delta' \vdash t).
\end{align*}
\]

We may omit subindices if they are clear from the context.

The narrowing relation \( (\Delta \vdash s) \sim_{R} (\Delta' \vdash t) \) is the reflexive transitive closure of the one-step narrowing relation, that is, the least relation that includes the one-step narrowing relation and such that: for all \( \Delta, s, s' : (\Delta \vdash s) \sim_{R} (\Delta \vdash s') \) if \( \Delta \vdash s \approx_{\alpha} s' \); for all \( \Delta, \Delta', S, t, u : (\Delta \vdash s) \sim_{R} (\Delta' \vdash t) \) and \( (\Delta' \vdash t) \sim_{R} (\Delta'' \vdash u) \) implies \( (\Delta \vdash s) \sim_{R} (\Delta'' \vdash u) \).

The Lifting Theorem given below relates nominal narrowing and nominal rewriting. It is an extension of Hullot’s Theorem 1 [17], taking into account freshness contexts and equivalence. The notions of normalised substitution-in-context and satisfiability of freshness contexts play a key role. A substitution \( \sigma \) is normalised in \( \Delta \) w.r.t. a rewrite theory \( R \) if \( \Delta \vdash X \sigma \) is a normal form in \( R \) for every \( X \). A substitution \( \sigma \) satisfies the freshness context \( \nabla \) if there exists a freshness context \( \nabla \) such that \( \nabla \vdash a\#X \sigma \) for each \( a\#X \in \Delta \); the minimal such \( \nabla \) is \( (\Delta \sigma)_{n.f} \).

Theorem 12 (Lifting). Let \( R = \{\nabla_i \vdash l_i \rightarrow r_i\} \) be a convergent rewrite theory. Let \( \Delta_0 \vdash s_0 \) be a nominal term-in-context and \( V_0 \) a finite set of variables containing \( V = V(\Delta_0, s_0) \). Let \( \eta \) be a substitution with \( \text{dom}(\eta) \subseteq V_0 \) and satisfying \( \Delta_0 \), that is, there exists \( \Delta \) such that \( \Delta \vdash \Delta_0 \eta \). Assume moreover that \( \eta \) is normalised in \( \Delta \). Consider a rewrite derivation:

\[
\begin{align*}
\Delta \vdash s_0 & \eta \quad t_0 \xrightarrow{\sim_{\{\Gamma_0, R_0, \sigma_0\}} \cdots \sim_{\{\Gamma_{n-1}, R_{n-1}, \sigma_{n-1}\}}} t_n
\end{align*}
\]

There exists an associated nominal narrowing derivation:

\[
\begin{align*}
(\Delta_0 \vdash s_0) & \sim_{\{\Gamma_0, R_0, \sigma_0\}} \cdots \sim_{\{\Gamma_{n-1}, R_{n-1}, \sigma_{n-1}\}} (\Delta_0 \vdash s_n)
\end{align*}
\]

for each \( i, 0 \leq i \leq n \), a substitution \( \eta_i \) and a finite set of variables \( V_i \supseteq V(s_i) \) such that:

1. \( \text{dom}(\eta_i) \subseteq V_i \),
2. \( \eta_i \) is normalised in \( \Delta_i \),
3. \( \Delta \vdash \eta_i \mid V \approx_{\alpha} \theta_i \eta_i \mid V \),
4. \( \Delta \vdash s_i \eta_i \approx_{\alpha} t_i \),
5. \( \Delta \vdash \Delta_i \eta_i \)

where \( \theta_0 = 1d \) and \( \theta_{i+1} = \theta_i \sigma_i \).

Conversely, to each nominal narrowing derivation of the form (**) and every \( \eta \) such that \( (\Delta_0, \theta_n) \preceq^V (\Delta, \eta) \) and \( \Delta \vdash s_i \eta \approx_{\alpha} t_i \) we can associate a nominal rewriting derivation of the form (**).
Nominal Narrowing

\[
\Delta \vdash s \theta = t_0 \to \ldots \to t_i \to \ldots \to t_{n-1} \to t_n \quad \Delta \vdash s_0 \sigma \quad t_i \sigma \approx t_{i+1} \sigma \quad \ldots \quad t_n \sigma = t_n \\
\Delta_0 \vdash s_0 \sigma_0 \quad \ldots \quad \Delta_i \vdash s_i \sigma \quad (\Delta_{i+1} \vdash s_{i+1} \sigma) \quad \ldots \quad \Delta_n \vdash s_n \sigma
\]

**Figure 3** Corresponding Rewriting and Narrowing Steps.

**Proof.**

\((\implies)\) The proof is by induction on the length of the derivation. Figure 3 illustrates the relation between the two derivations.

**Base Case.** For \(n = 0\), take \(\eta_0 = \eta\), \(V_0 = V \cup \text{dom}(\eta)\). By assumption, \(\Delta \vdash \Delta_0 \eta_0\).

\[
\Delta_0 \vdash s_0 \eta_0 \quad \text{and} \quad \Delta_0 \vdash s_0 \eta = s_0
\]

**Induction Step.** Assume conditions (1)-(5) hold for \(i\), and \(\Delta \vdash t_i \rightarrow_{[C_i, R_i]} t_{i+1}\) (see Figure 3). We have:

(a) \(R_i = \nabla_i \vdash l_i \rightarrow r_i \in R, V(R_i) \cap V(\Delta, t_i) = \emptyset\).
(b) \(t_i \equiv C_i[t'_i]\) for some position \(C_i[\_]\) and \(\Delta \vdash \nabla_i \tau, \pi \cdot (l_i \sigma) \approx_i t_i\).
(c) \(\Delta \vdash C_i[\pi \cdot (r_i \sigma)] \approx_i t_{i+1}\)

Also, \(\text{dom}(\sigma) \cap V_i = \emptyset\) since \(V(R_i) \cap V(\Delta, t_i) = \emptyset\).

By IH, it follows from assumptions 2., 4. and 5. that \(s_i \equiv C_i[t'_i] \in \text{Pos}(s_i)\) and \(\Delta \vdash s_i \eta \approx_i t'_i \approx_i \pi \cdot (l_i \sigma)\) (if \(C_i[\_]\) were a variable position the term \(t'_i\) would be a variable, from (4), \(\Delta \vdash s_i \eta \approx_i t'_i \approx_i \pi \cdot (l_i \sigma) \rightarrow_{R_i} \pi \cdot (r_i \sigma)\), contradicting that \(\eta_i\) is a normalised substitution).

Let us consider \(\rho = \eta \cup \sigma\), we have \(\Delta \vdash s'_i \rho \approx_i \pi \cdot (l_i \rho)\). The pair \((\Delta, \rho)\) is a solution for \((\Delta_i \vdash s'_i \rho) \approx_i (\nabla_i \vdash l_i)\):

(i) \(\Delta \vdash \Delta_i \rho\), because, by hypothesis, \(\Delta \vdash \Delta_i \eta_i\) and \(\sigma\) does not affect \(\Delta_i\) (\(\text{dom}(\sigma) \subseteq V(R_i)\)).

(ii) \(\nabla_i \vdash l_i \rho\).

(iii) \(\Delta \vdash s'_i \rho \approx_i \pi \cdot (l_i \rho)\).

Now, take the principal solution \((\Delta_{i+1}, \sigma_i)\) of \((\Delta_i \vdash s'_i) \approx_i (\nabla_i \vdash \pi \cdot l_i)\). Then, \(\Delta_{i+1} \vdash \Delta_i \sigma_i, \nabla_i \sigma_i, s'_i \sigma_i \approx_i \pi \cdot (l_i \sigma_i)\). Let \(s_{i+1}\) be a nominal term such that \(\Delta_{i+1} \vdash C_i[\pi \cdot r_i \sigma_i] \approx_i s_{i+1}\). Therefore, \((\Delta_i \vdash s_i) \rightarrow_{[C_i, R_i, \sigma_i]} (\Delta_{i+1} \vdash s_{i+1})\).

Since \((\Delta_{i+1}, \sigma_i)\) is the least unifier of \((\Delta_i \vdash s'_i) \approx_i (\nabla_i \vdash \pi \cdot l_i)\), \((\Delta_{i+1}, \sigma_i) \leq (\Delta, \rho)\) and thus there exists a substitution \(\eta'\) such that for all \(X\), \(\Delta \vdash X \sigma_i \eta' \approx_\sigma X \rho\) and \(\Delta \vdash \Delta_{i+1} \eta'\).

That is, \(\Delta \vdash \sigma_i \eta' \approx_\sigma\rho\). Since \(\rho = \eta_i \cup \sigma\) and \(\text{dom}(\sigma) \cap V_i = \emptyset\), \(\eta_i\) is such that \(\Delta \vdash \eta_i \approx_\sigma \sigma_i \eta' | V_i\).

Now let \(V_{i+1} = (V_i \cup \text{Im}(\sigma_i)) \setminus \text{dom}(\sigma_i)\) and let \(\eta_{i+1}\) be such that \(\Delta \vdash \eta_{i+1} \approx_\sigma \eta'_i | V_{i+1}\).

We get condition 1., that is, \(\text{dom}(\eta_{i+1}) \subseteq V_{i+1}\) and from 3.: \(\Delta \vdash \eta_{i+1} \approx_\sigma \sigma_i \eta' | V_{i+1}\) (1).

(By hypothesis, \(\Delta \vdash \eta_i | V_i = \eta_i \approx_\sigma \eta_i\). To illustrate, take \(i = 4\), then \(\eta_4 | V_i = \eta_i \approx_\sigma \eta_i\). Using the definition of \(\eta_i\), it follows that \(\eta_0 = \sigma_0 \sigma_1 \ldots \sigma_3 \) and \(\eta_5 = \sigma_5 \sigma_4 \ldots \sigma_{11}\). Thus, \(\eta_4 \approx_\sigma \sigma_0 \sigma_1 \ldots \sigma_3 \sigma_4 \eta_5 \approx_\sigma \eta_5 \approx_\sigma \eta_5\).)

Recall that we impose \(\text{dom}(\sigma_i) \cap V_i = \emptyset\).

To prove 5. for \(i + 1\), notice that from \(\Delta \vdash \Delta_{i+1} \eta'\) it follows that \(\Delta \vdash \Delta_{i+1} \eta_{i+1}\), since \(\Delta \vdash \eta_{i+1} \approx_\sigma \eta_i\).

To prove 2. for \(i + 1\), let us consider \(X \in V_{i+1}\). There are two cases:

(i') \(X \in V_i \setminus \text{dom}(\sigma_i)\) then \(\Delta \vdash X \eta_i \approx_\sigma X \sigma_i \eta' \approx_\sigma X \eta_i \approx_\sigma X \eta_{i+1}\). Since \(\eta_i\) is a normalised substitution, by hypothesis, it follows that \(\eta_{i+1}\) is also a normalised substitution.
X ∈ V(Im(σ)), then there exists Y ∈ dom(σi) such that X ∈ V(Yσi). Then, Xηi+1 is a subterm of Yηi since Δ ⊢ Xηi+1 ≈α Xη′, Yσiη′ ≈α Yηi, and since, by hypothesis, ηi is a normalised substitution, it follows that ηi+1 is also normalised. This proves (2) for i + 1.

We now prove 3. for i + 1, assuming it for i, i.e., Δ ⊢ ηi|V ≈α θiηi|V.

From equation (1) we get Δ ⊢ θiηi|V ≈α θi(σiηi+1|V)|V. From the definition of θi, we have Im(θi) ⊆ V and V′ ⊆ V ∪ dom(θi). Therefore, Δ ⊢ θiσiηi+1|V ≈α θiηi|V ≈α ηi|V proving condition 3) for i + 1. Notice that, by 3), θi is normalised.

Finally, on the one hand Δ ⊢ t+i ≈α C1[π · r · σ] ≈α C1[π · r · ηi] ≈α C1[π · r · ηi]. On the other hand, Δ ⊢ s+i1ηi+1 ≈α (C1[π · r · σ]ηi+1) ≈α (C1[π · r · σi]ηi) ≈α (C1[σiηi]) ≈α C1[π · r · ηi]. Therefore, Δ ⊢ s+i1ηi+1 ≈α t+i, proving (4).

(⇐) Conversely, let us consider a derivation (**): (Δ₀ ⊢ s₀) ≈α (C_n−1[σ_n−1] · ... · C_0[σ_0]) (Δ₀ ⊢ s₀), and a substitution η such that (Δ₀, η) ≤(Δ, η), that is, there exists ρ such that Δ ⊢ Xη|V ≈α (Xη|V) and Δ ⊢ Δ₀ρ. We define substitutions ηi for 0 ≤ i ≤ n − 1 by:

Δ ⊢ ηi ≈α σ₀ · ... · σ_{n-1}ρ (2), and a normalised substitution η_{n−1} ≡ ρ. By hypothesis, Δ ⊢ Δ₀ρ, and by definition of narrowing step, it follows that Δ_{i+1} ⊢ Δ_iσ_i (0 ≤ i ≤ n − 1). Hence Δ ⊢ Δₙηₙ, and in particular Δ ⊢ Δ₀η₀. We define s₁η₀ ≡ t₀ for 0 ≤ i ≤ n, and show, by induction on i, that:

Δ ⊢ sᵢηᵢ ≈α tᵢ.

Base Case. When i = 0: Δ ⊢ s₀η₀ ≈α s(θ₀η₀) ≈α sη. By definition, η₀ = σ₀, i.e., η₀ ≈α σ₀. 

Induction Step. Suppose that (Δ, s_i) ≈α (C_i[σ_i]) (Δ_{i+1} ⊢ s_{i+1}). By the definition of nominal narrowing we have

1. R_i ⊢ θ_i ⊢ t_i ∈ R, V(R_i) ∩ V(Δ_i, s_i) = ∅.
2. s_i ≡ C_i[σ_i], for a non-variable position C_i, and such that (Δ_{i+1}, σ_i) is the least solution for (Δ ⊢ s_i) ≈α (Δ ⊢ π · l · θ_i). That is, Δ_{i+1} ⊢ s_iσ_i ≈α π · l · θ_i ∧ Δ_{i+1} ⊢ Δ_iσ_i, V(Δ_i).
3. Δ_{i+1} ⊢ C_i[π · r · σ_i] ≈α s_{i+1}.

By definition, Δ ⊢ s_iη_i ≈α t_i. Since C_i is a non-variable position and η_i is a normalised substitution, we have that Δ ⊢ s_iη_i ≈α t_i. In addition, define η′ ≡ σ_{i+1} · ... · σ_{n−1}ρ, by equation (2) Δ ⊢ η_{i+1} ≈α η′ |V_{i+1}, Δ ⊢ t'_i ≈α s'_iη_i ≈α s'_i(σ′) ≈α (π · l · σ_i) →_{R_i} (π · r · θ_i)η′. Therefore, Δ ⊢ t_i ≡ C_i[t_i'] →_{R_i} C_i[π · r · σ_i] ≈α s_{i+1}η_{i+1} ≈α t_{i+1}.

In a similar way, we can associate closed nominal rewriting derivations (see Definition 6) with closed nominal narrowing derivations, where closed narrowing is defined as follows.

**Definition 13** (Closed narrowing). Given a rewrite rule R = (Δ ⊢ l → r) and a term-in-context Δ ⊢ s, write (Δ ⊢ s) →_{R} (Δ ⊢ t) when there is some R′ a freshened variant of R (so fresh for R, Δ, s, and t), position C and substitution θ such that s ≡ C[s′] and Δ', A(R') ⊨ V(Δ', s, t) ⊨ (Δ' ⊢ Δ', s ⊢ θ, s'θ ≈α t'θ, (C[r']) ≈α s,t). We call this (one-step) closed narrowing. The *closed narrowing relation* Δ ⊢ s →_{α} Δ' ⊢ t is the reflexive transitive closure as in Definition 5.

See Example 17 in Section 3.1 for examples of closed narrowing steps.

**Remark.** We can state a “closed lifting” theorem by replacing nominal rewriting/narrowing for closed rewriting/narrowing. The proof is similar.
Nominal Narrowing

In the following we consider a closed nominal equational theory \( \mathcal{E} \), presented by a convergent set \( \mathcal{R} \) of closed rules.

Let us consider an \( \mathcal{E} \)-unification problem \( (\Delta \vdash s) \stackrel{\mathcal{E}}{\in} (\nabla \vdash t) \). To find a solution, we will apply closed narrowing on \( \Delta \vdash s \) and \( \nabla \vdash t \) in parallel. It will simplify matters to narrow the single term \( u = (s,t) \) under \( \Delta, \nabla \).

**Lemma 14 (Soundness).** Let \( \Delta \vdash s \) and \( \nabla \vdash t \) be two nominal terms-in-context and \( \Delta, \nabla \vdash (s,t) = u_0 \leadsto \cdots \leadsto u_n \) a closed narrowing derivation such that \( \Delta_n, s_n \approx_n t_n \) has a solution, say \( (\Gamma, \sigma) \). Then \( (\Gamma, \theta_n \sigma) \) is an \( \mathcal{E} \)-solution of the problem \( \Delta, \nabla, s \approx_n t \), where \( \theta_n \) is the composition of substitutions along the narrowing derivation, as defined in Theorem 12.

**Proof.** Using the \( (\approx) \) part of the previous theorem with \( \eta = \theta_n \), we can associate this narrowing derivation with the following rewriting derivation:
\[ \Gamma \vdash u_0 \theta_n = v_0 \leadsto v_1 \leadsto \cdots \leadsto v_n \theta_n = (v_n, v_n') \, . \]
Thus, \( \Gamma \vdash \theta_n \, \Gamma \vdash t \theta_n \leadsto v_n' \). Moreover, since \( \eta_n = Id \) (because \( \eta = \eta_n \theta_n \)) it follows that \( \Gamma \vdash v_n \approx_n s_n \) and \( \Gamma \vdash v_n' \approx_n t_n \), thus: \( \Gamma \vdash \theta_n \sigma = \theta_n \theta_n \sigma \) and therefore, \( (\Gamma, \theta_n \sigma) \) is an \( \mathcal{E} \)-solution for \( \Delta, \nabla \vdash s \approx_n t \).

**Lemma 15 (Completeness).** Let \( \Delta \vdash s \) and \( \nabla \vdash t \) be two nominal terms-in-context, such that the problem \( (\Delta \vdash s) \stackrel{\mathcal{E}}{\in} (\nabla \vdash t) \) has an \( \mathcal{E} \)-solution, \( (\Delta', \rho) \), and let \( V \) be a finite set of variables containing \( V(\Delta, \nabla, s, t) \). Then there exists a closed narrowing derivation:
\[ \nabla \vdash u = (s,t) \leadsto \cdots \leadsto u_n = (s_n, t_n) \, , \] such that \( \Gamma_n, s_n \approx_n t_n \) has a solution. Let \( (\Gamma, \rho) = mgu(\Gamma_n, s_n \approx_n t_n) \), and \( \theta_n \) the composition of the narrowing substitutions. Then, \( (\Gamma, \theta_n \mu) \preceq_E (\Delta', \rho) \). Moreover, we are allowed to restrict our attention to \( \leadsto \)-derivations such that: \( \forall i, 0 \leq i \leq n, \theta_i | \sigma \) is normalised.

**Proof.** By Definition 8, \( \Delta' \vdash \mathcal{E} s_p = t_p \vdash \nabla \vdash \Delta \rho \). Take \( \eta = \rho \downarrow \), that is, \( \rho \)’s normal form in \( \Delta' \):
\[ \Delta' \vdash X \eta \approx \alpha (X \rho) \downarrow \, . \] It follows that \( \Delta' \vdash s \eta = t \eta \vdash \nabla \eta \vdash \Delta \eta \) since the rules are closed.

Since \( \mathcal{E} \) is a closed nominal theory presented by a convergent rewrite system \( \mathcal{R} \), and since closed rewriting is complete for equational reasoning in this case, \( s \eta \) and \( t \eta \) have the same normal form in \( \Delta' \), which we will call \( r \). Then, \( \Delta' \vdash u \eta = (s \eta, t \eta) = u_0' \leadsto \cdots \leadsto u_n' = (r, r) \). By Theorem 12 there exists a corresponding \( \leadsto \)-derivation ending with \( \Gamma_n \vdash (s_n \eta, t_n \eta) \) such that:
\[ \Delta' \vdash (s_n \eta_n, t_n \eta_n) \approx_n t_n' = (r, r) \, and \, \Delta' \vdash \Gamma_n \eta_n \, . \] Thus, \( (\Delta', \eta_n) \) is a solution of \( \Gamma_n, s_n \approx_n t_n \).

Since \( (\Gamma, \rho) \) is the least unifier, it follows that \( (\Gamma, \mu) \preceq (\Delta', \eta_n) \) and:
\[ \exists t : \forall X, \Delta' \vdash X \eta_n \vdash \Delta' \vdash \Gamma_n X \eta \, and \, \Gamma_n \mu \preceq \Delta' \, \vdash (\Gamma, \mu) \vdash X \eta | \rho \approx \alpha \eta \eta | \rho \approx \alpha \eta \] and \( \Delta' \vdash \eta | \sigma = \rho | \sigma \) that is, \( (\Gamma, \theta_n \mu) \preceq_E (\Delta', \rho) \).

Now we can describe how to build a complete set of \( \mathcal{E} \)-unifiers for two terms-in-context.

**Theorem 16.** Let \( \mathcal{E} \) be a closed nominal equational theory and \( \mathcal{R} \) be an equivalent convergent nominal rewrite theory. Let \( \Delta \vdash s \) and \( \nabla \vdash t \) be two terms-in-context, and \( V \) be a finite set of variables containing \( V(\Delta, s, \nabla, t) \). Let \( S \) be the set of pairs \( (\Gamma, \sigma) \) such that there exists an \( \leadsto \)-derivation: \( \Gamma_0 \vdash u = (s,t) = u_0 \leadsto \cdots \leadsto \Gamma_n \vdash t_n = (s_n, t_n) \), where \( (\Gamma_0 \equiv \Delta, \nabla) \), \( \Gamma_n, s_n \approx_n t_n \) has a least solution \( (\Gamma, \mu), \sigma \equiv \theta_n \mu \), and \( \theta_n \) is the normalised composition of the narrowing substitutions. Then \( S \) is a complete set of \( \mathcal{E} \)-unifiers of \( \Delta \vdash s \) and \( \nabla \vdash t \) away from \( V \).

**Proof.** Consequence of Lemmas 14 and 15.
\[ y \# F \vdash \text{diff}(\lambda y [y] F, X) \to 0 \]
\[ y \# F \vdash \text{diff}(\lambda y [y] y, X) \to 1 \]
\[ y \# F \vdash \text{diff}(\lambda y [y] \sin(F), X) \to \text{mult}(\text{sub}([y] F, X), \text{diff}([y] F, X)) \]
\[ y \# F \vdash \text{diff}(\lambda y [y] \cos(F), X) \to \text{mult}(\text{diff}([y] F, X), \text{sub}([y] G, X)) \]
\[ y \# F \vdash \text{diff}(\lambda y \text{mult}(F, G), X) \to \text{plus}(\text{diff}([y] F, X), \text{diff}([y] G, X)) \]
\[ y \# F \vdash \text{diff}(\lambda y \text{sub}(F, G), X) \to \text{plus}(\text{diff}([y] F, X), \text{sub}([y] G, X)) \]
\[ y \# F \vdash \text{diff}(\lambda y \text{mult}(F, G), X) \to \text{plus}(\text{mult}(\text{diff}([y] F, X), \text{sub}([y] G, X)), \text{mult}(\text{diff}([y] G, X), \text{sub}([y] F, X))) \]

Figure 4 Rewrite rules for symbolic differentiation.

An \( E \)-unification procedure follows from the construction of Theorem 16: enumerate all elements of \( S \). The set \( S \) may be infinite, one can organise the enumeration in such a way that if two nominal terms \( \Delta \vdash s \) and \( \nabla \vdash t \) are \( E \)-unifiable, then an \( E \)-solution will be produced in a finite number of steps. Thus, assuming \( E \) is presented by a convergent rewrite theory \( R \), we have a semi-decision procedure for nominal \( E \)-unification.

3.1 An example: symbolic differentiation

The rewrite rules in Figure 2 define a \( \lambda \)-calculus with names and explicit substitutions [13]; the extension with numbers and operations (\( \text{plus}, \text{mult}, \text{sin}, \text{cos} \)) is straightforward.

Consider now symbolic differentiation [26]: \( \text{diff}(F, X) \) computes the differential of a function \( F \) (meta-level unknown that can be instantiated by a \( \lambda \)-term) at a point \( X \), using the rewrite rules given in Figure 4.

Example 17. Let \( E \) be the theory defined by rewrite rules in Figures 2 and 4 together with standard rules for arithmetic operations. This system is closed but not convergent (we can simulate the untyped \( \lambda \)-calculus, which is non-terminating) so narrowing is not necessarily complete; however, we can still obtain the \( E \)-solution (\( \emptyset \{ F \mapsto y \} \)) for the nominal \( E \)-unification problem \( \lambda x ([z] \text{diff}([y] \text{sin}(F)), z) \vdash [z] \text{cos}(z) \)

The first closed-narrowing step uses a freshened rule
\[ \vdash \text{diff}(\lambda y [y] \sin(F'), X') \to \text{mult}(\text{cos}(\text{sub}([y] F', X'), \text{diff}([y] F'), X')) \]
with the assumption \( y \# F \) (below the narrowed subterm is in bold, the substitution used is \( \{ F' \mapsto (y y') \cdot F, X' \mapsto z \} \):

\[ \lambda x ([z] \text{diff}([y] \text{sin}(F)), z) \leadsto [z] \text{cos}(z) \]
\[ \leadsto [z] \text{mult}(\text{cos}(\text{sub}([y] y', [y y'] \cdot F, z)), \text{diff}([y] (y y') \cdot F, z)) \] \[ \leadsto [z] \text{cos}(z) \]

We now use the freshened rule \( \vdash \text{diff}(\lambda w [w] W, W) \to 1 \) with substitution \( \{ F \mapsto y, W \mapsto z \} \) and assumption \( w \# F \) to narrow the second argument of \( \text{mult} \):

\[ \leadsto [z] \text{mult}(\text{cos}(\text{sub}([y] y', z)), 1) \] \[ \leadsto [z] \text{cos}(z) \]

Using now the rules for \( \text{sub} \), we can rewrite (hence also narrow) to

\[ [z] \text{mult}(\text{cos}(z), 1) \] \[ \leadsto [z] \text{cos}(z) \]

and by rewriting with the usual rules for multiplication, we obtain two equal terms.

---

7 Here we do not rely on \( \text{Beta} \), \( \text{diff} \) uses just the substitution rules, which are terminating.
Nominal Narrowing

\begin{align*}
(1) & \emptyset \vdash \pi_i([X_1, X_2]) \rightarrow X_i & (i \in \{1, 2\}) \\
(2) & \emptyset \vdash d([X])_Y, Y^{-1} \rightarrow X \\
(3) & \emptyset \vdash d([X])_{Y^{-1}, Y} \rightarrow X \\
(4) & \emptyset \vdash (X^{-1})^{-1} \rightarrow X \\
(5) & \emptyset \vdash subst^a_\mu([z]z_k, \overline{X}) \rightarrow X_k & (1 \leq k \leq j) \\
(6) & z \# Y \vdash subst^a_\mu([z]Y, X) \rightarrow Y \\
(7) & z_k \# Y \vdash subst^a_\mu([z]Y, X) \rightarrow subst^a_{\mu-1}([z]y, X^\prime) & (1 \leq k \leq j, j > 1) \\
(8) & \emptyset \vdash subst^a_\mu([z]f(W), X) \rightarrow f(subst^a_\mu([z]W, X))
\end{align*}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure5}
\caption{Rewrite theory DYT.}
\end{figure}

\subsection{Application: Intruder Deduction Problem}

In this section we present an application of nominal E-matching.

\begin{definition}[Nominal Intruder Deduction Problem] Given a finite set of ground messages in normal form \( \Gamma = \{t_1, \ldots, t_n\} \), a ground message in normal form \( m \) (the secret), and private names \( a_1, \ldots, a_k \), we model the Intruder Deduction Problem (IDP) as a nominal E-matching problem with one unknown: \( \Delta \vdash subst([z]X, \overline{7}) \) if and only if \( m \). Here \( m \) is short for \( \emptyset \vdash m \) and \( \Delta = \{a_1 \# X, \ldots, a_k \# X\} \) is a freshness context specifying that the names \( a_1, \ldots, a_n \) are fresh in the unknown term \( X \). subst is a term-former denoting the substitution of \( z_1, \ldots, z_n \) (denoted by \( \overline{z} \)) by \( t_1, \ldots, t_n \) (denoted by \( \overline{7} \)), \( \overline{z} \) are abstracted in \( X \), and \( \overline{7} \) represent the messages in \( \Gamma \).

To illustrate the results we consider a simple equational theory, namely the Axiomatised Dolev-Yao Theory (DYT). It is essentially the classical Dolev-Yao model with explicit destructors such as decryption and projections. It is well-known that IDP for this theory is decidable in polynomial time\(^8\), the purpose here is to show how nominal narrowing could be used to solve this security problem.

The signature for DYT, \( \Sigma_{DYT} \), includes function symbols \( \langle \_ , \_ \rangle, \pi_1(\_), \pi_2(\_), d(\_ , \_) \), \( \langle \_ , \_ \rangle^{-1} \) for pairing, projections, decryption, encryption and inverse, respectively, as well as a family of symbols \( subst^a_\mu(n \geq 1, j \in \{1, \ldots, n\}) \) to perform substitution. Intuitively, projections are inverses of pairing and decrypting with \( k^{-1} \) a message encrypted with \( k \) gives back the plaintext.

The rewrite rules are given, in a schematic way, in Figure 5. The index \( j \) in \( subst^a_\mu \) denotes the number of abstracted atoms in \( [\overline{z}] \), for \( j \in \{1, \ldots, n\} \). In rule schemes (5) and (7), \( z_k \) is a term in \( \{z_1, \ldots, z_j\} \) and there is a rule for each \( k \) s.t. \( 1 \leq k \leq j \). In rule scheme (7), \( j > 1 \); in case \( j = 1 \) we use rule (6). In rules (7) and (8) we use the following abbreviations:

1. \( [z] = [z_1, \ldots, z_j] \) and \( [\overline{z}] = [z_1, \ldots, z_k-1, z_k+1, \ldots, z_j] \);
2. \( X = (X_1, \ldots, X_j) \) and \( X^\prime = (X_1, \ldots, X_k-1, X_k+1, \ldots, X_{j-1}) \);
3. \( f \in \Sigma_{DYT} \) is an \( r \)-ary function symbol (there is a version of rule (8) for each \( f \neq subst \), and \( f(subst^a_\mu([z]W, X)) = f(subst^a_\mu([z]W_1, X), \ldots, subst^a_\mu([z]W_r, X)) \)).

\begin{definition} DYT is a closed and convergent nominal rewrite system.\end{definition}

\begin{proof}
The termination is obtained by a simplification ordering. It is convergent because the critical pairs obtained are joinable [11].
\end{proof}

\begin{footnote}
\footnotetext[8]{This result was obtained using another approach [7].}
\end{footnote}
\[
X \{ z \mapsto t \} \xrightarrow{\text{DYT}} m
\]

Figure 6 First level of the narrowing tree.

**Remark.** Below, the notation \( t \{ z \mapsto t' \} \) is syntactic sugar for \( \text{subst}(\{z\mapsto t\}, t') \).

**Example 20.** Consider \( \Gamma = \{ \{ m \}_h, \{ b^{-1} \}_h, \{ c^{-1} \}_h, k^{-1}, r^{-1} \} \) and a secret \( m \) (a constant). Taking into account the theory DY T, this IDP can be stated as \( X \{ z \mapsto t \} \xrightarrow{\text{DYT}} m \), where \( \{ z \mapsto t \} \) denotes the substitution of \( t_i \) for \( z_i \), \( i = 1, \ldots, 5 \). Figure 6 shows part of the first level of the narrowing tree for this problem.

The substitutions \( \theta_i \) are \( \{ X \mapsto z_i \}, i = 1, \ldots, 5 \) and the corresponding narrowing steps use rule (5). The result \( t_i \xrightarrow{\gamma} m \) is a ground problem, which can be decided by checking syntactic equality since each \( t_i \) and \( m \) are in normal form. The branch labelled with the substitution \( \theta_f \) is an abbreviation for six branches, namely, one for each \( f \in \Sigma_{\text{DYT}} \) (except \( \text{subst} \)).

To illustrate, consider the case in which \( f \) is a constructor, for instance, \( f = \langle \, , \rangle \):

\[
X \{ z \mapsto t \} \xrightarrow{\text{DYT}} m
\]

This branch is obtained via

\[
\text{subst}_f^l([w|W_1, W_2], \tilde{z}) \rightarrow \langle \text{subst}_f^l([w|W_1, \tilde{z}], \text{subst}_f^l([w|W_2, \tilde{z}])\rangle
\]

and substitution \( \theta_f = \{ X \mapsto \langle X_1, X_2 \rangle, W_1 \mapsto ( w \cdot X_1, W_2 \mapsto ( w \cdot X_2, \tilde{Z} \mapsto \tilde{T} \rangle \) with the assumption \( w \# X \).

Consider the case in which \( f \) is a destructor, for instance, \( f = d \). There is a narrowing step:

\[
X \{ z \mapsto t \} \xrightarrow{\text{DYT}} m \xrightarrow{\theta_d} d(X_1 \{ z \mapsto t \}, X_2 \{ z \mapsto t \}) \xrightarrow{\text{DYT}} m
\]

obtained via substitution \( \theta_d = \{ X \mapsto d(X_1, X_2) \} \) and rule (8). From this node we can narrow with \( \theta_d^1 = \{ X_1 \mapsto z_i \} \), or \( \theta_d^2 = \{ X_2 \mapsto z_i \} \) (i = 1, \ldots, 5), or \( \theta_d^{f_1} = \{ X_1 \mapsto f(X'_1) \} \) or \( \theta_d^{f_2} = \{ X_2 \mapsto f(X'_2) \} \) (\( f \in \Sigma_{\text{DYT}} \)).
The left branch represents 5 narrowing branches, one for each $i$. After applying rule (5) one has $c_{d,1} := d(t_i, X_2) \overset{\text{DYT}}{\approx} m$. Similarly, $c_{d,2}$ represents 6 other possible branches, one for each function symbol from $\Sigma_{\text{DYT}}$. Iterating this reasoning, we obtain the narrowing branch shown in Figure 7, which leads to a ground problem whose solution is positive.

The previous example illustrates the fact that a series of narrowing steps might be necessary in order to obtain a solution. Variables might need to be instantiated with constructors for two reasons:

- either the term $m$ contains a sequence of constructors in its structure, therefore, the variables in the term being matched have to be instantiated with the same sequence of constructors, and rule (8) applies;
- or a sequence of constructors matches a sequence of the corresponding destructors in a term in $\Gamma$, enabling a rewriting rule to be applied.

As a consequence, the number of applications of DYT rules is bounded by $|\Gamma| + |m|$.

**Theorem 21.** If a narrowing derivation $(\Delta_0 \vdash \text{subst}([z \mapsto \overline{t}]X, \overline{t}), m) \leadsto \sigma_0 \ldots \leadsto \sigma_{k-1} (\Delta_k \vdash u_k)$ has more than $|\Gamma| + |m|$ narrowing steps then $\text{height} (\text{subst}([z \mapsto \overline{t}]X, \overline{t})\sigma_0\sigma_1\ldots\sigma_{k-1}) > \text{height}(m)$. Therefore, it does not lead to a solution.

**Proof.** Each application of a Dolev-Yao rule eliminates one symbol from the term. In the worst case, in all terms from $\Gamma$ all the function symbols can be eliminated by a rule, before several steps of composition (with a constructor that has not just been eliminated) can be applied until one reaches the size of $m$.

Notice that for infinite branches of the form $\Pi, \Pi, \Pi, \ldots$ or $ddd\ldots$ either the term $m$ would have to be headed with the same sequence of functions or rewrite rules would be applied. By the Lifting Theorem, we can assume the compositions of substitutions are normalised, therefore, the only way to apply rewrite rules is when the terms in $\Gamma$ contain, in the first case, a sequence of pairings $\langle\langle\ldots\rangle\rangle$ or, in the second case, a sequence of encryptions $\{\{\ldots\}\}$. We cannot introduce a destructor followed by its corresponding constructor with a substitution, e.g, $\Pi, \Pi, \Pi, \langle\langle\ldots\rangle\rangle$, otherwise the substitution would not be normalised. Since
all the terms in \( \Gamma \) are finite, only a finite number of destructive rewrite rules could be applied and the number of constructive rewrite rules that could be applied is bounded by the size of \( m \). The same reasoning applies when we have interleaving of destructors \( d \Pi_1 d \Pi_2 d \Pi_3 \ldots \) or even constructors and destructors of the form \( d[\{]d[\{]d[\{] \), when the encryption/decryption keys do not correspond.

As a consequence, we obtain the decidability of the nominal IDP for DYT.

\section{Basic Nominal Narrowing}

Hullot \cite{Hullot} introduced basic narrowing to eliminate redundant narrowing derivations in order to give sufficient conditions for the termination of the narrowing process. Following \cite{Hullot}, with the corrections made in \cite{Avanzini09,Avanzini12}, we define basic (closed) nominal narrowing. In the rest of this section, \( R = \{ R_k \equiv \nabla \vdash t_k \rightarrow r_k \} \) is a closed nominal rewrite theory.

\begin{definition}
Consider a nominal term \( s \) and a set \( U \) of positions that are proper prefixes of \( s \), that is, \( U = \text{Pos}(r) \), for some subterm \( r \) of \( s \). We define by induction what it means for a nominal rewriting derivation \( \Delta \vdash s = s_0 \rightarrow [C_0, R_0] s_1 \rightarrow [C_1, R_1] \rightarrow \cdots \rightarrow [C_{n-1}, R_{n-1}] s_n \) to be based on \( U \) and construct sets of positions \( U_i \subseteq \text{Pos}(s_i) \), \( 0 \leq i \leq n \), inductively: the empty derivation is based on \( U \), and \( U_0 = U \); if a derivation up to \( s_i \) is based on \( U \), then the derivation obtained from it by adding one step \( s_i \rightarrow [C_i, R_i] s_{i+1} \) is based on \( U \) iff \( C_i \in U_i \), and in this case we take: \( U_{i+1} = (U_i - \{ C \in U_i | C \leq C_i \}) \cup \{ C_i | C \in \text{Pos}(r_i) \} \), where \( r_i \) denotes the right-hand side of the rule \( R_i \) in \( R \).
\end{definition}

A nominal rewrite step \( \Delta \vdash C[s] \rightarrow C[s'] \) at position \( C \) is innermost if for any \( C_i \) such that \( C < C_i \) and \( C[s] = C_i[s_i] \), there is no rewrite step \( \Delta \vdash C_i[s_i] \rightarrow C_i[t_i] \) at position \( C_i \). In other words, there is no rewrite step inside \( s \). An innermost nominal rewrite derivation contains only innermost rewrite steps.

\begin{lemma}
Let \( \Delta \vdash s \sim_\alpha s_0 \eta \), with \( \eta \) normalised in \( \Delta \). Every innermost nominal rewrite derivation from \( \Delta \vdash s \) is based on \( \text{Pos}(s_0) \).
\end{lemma}

\begin{definition}
A nominal narrowing derivation \( \tilde{\Delta}_0 \vdash s_0 \sim \{ C_0, R_0, \sigma_0 \} \ldots \sim \{ C_{n-1}, R_{n-1}, \sigma_{n-1} \} (\Delta_n \vdash s_n) \) (\( \Delta_i \vdash s_i \)), is basic if it is based on \( \text{Pos}(s_0) \) (in the same sense as in the previous definition for nominal rewriting derivation).
\end{definition}

\begin{theorem}
The narrowing derivations constructed in Theorem 12 are all basic.
\end{theorem}

\begin{proof}
Let \( \tilde{\Delta}_0 \vdash s_0 \sim \{ C_0, R_0, \sigma_0 \} \ldots \sim \{ C_{n-1}, R_{n-1}, \sigma_{n-1} \} (\Delta_n \vdash s_n) \) be the nominal narrowing derivation associated by Theorem 12 with \( \Delta \vdash s_0 \eta = t_0 \rightarrow [C_0, R_0] \ldots \rightarrow [C_{n-1}, R_{n-1}] t_n \), such that \( \eta \) is normalised. Since \( R \) is confluent we may assume that the nominal rewriting sequence from \( \Delta \vdash s_0 \eta \) is innermost. By Lemma 23, this nominal rewriting derivation is based on \( \text{Pos}(s_0) \), and since the sets \( U_i \) in the two derivations are equivalent, it follows that the considered nominal narrowing derivation is basic.
\end{proof}

\begin{remark}
Definition 22, Lemma 23 can also be stated for closed narrowing. Theorem 16 holds also for closed basic narrowing.
\end{remark}

The main interest of closed basic narrowing is that we can give a sufficient condition for the termination of the narrowing process when we consider only basic \( \sim \)-derivations and therefore for the termination of the corresponding nominal E-unification procedure.

\footnote{Given \( C_i = (s_i, \_ ) \) and \( C = (s, \_ ) \), it follows \( C_i C = (s_i \{ \_ \rightarrow s \}, \_ ) \) and \( C_i \leq C \) if \( \exists t : s_i \{ \_ \rightarrow t \} = s \).}
Proposition 26. Let $R = \{ \forall k \vdash l_k \rightarrow r_k \}$ be a convergent nominal rewriting system such that any basic $\sim$-derivation issuing from any of the right-hand sides $r_k$ terminates. Then any basic $\sim$-derivation issuing from any nominal term terminates.

The previous proposition also holds for basic closed narrowing.

Theorem 27. Basic closed nominal narrowing is complete for convergent closed nominal rewriting systems.

Moreover, if $R$ satisfies the hypothesis of Proposition 26, nominal basic narrowing leads to a complete and finite $E$-unification algorithm.

5 Conclusion and Future Work

We have introduced the nominal narrowing relation and designed a general nominal $E$-unification procedure, which is complete for a wide class of theories, namely, the theories defined by convergent closed nominal rewriting systems.

There is a lot of work to be done regarding nominal $E$-unification. A first step would be to study the relationship between nominal narrowing and pattern narrowing [26]. For the analysis of protocols, it would be interesting to study nominal unification modulo equational theories including associativity and commutativity axioms. From a practical point of view, narrowing strategies should be studied, such as lazy narrowing for nominal terms, and also more general versions of nominal narrowing such as conditional [26] and variant [10] narrowing, which have interesting applications [21, 23]. We would like to define conditions for termination of nominal narrowing similar to the finite variant and boundedness properties [6], to obtain an alternative way to study the security of protocols, via nominal narrowing.

Acknowledgements. We thank Santiago Escobar, Jesus Dominguez Alvarez and the FSCD reviewers for their valuable comments, which helped us improve the paper.

References