Two Views on Unification: Terms as Strategies

Furio Honsell
Department of Mathematics, Computer Science and Physics, University of Udine, Italy

Marina Lenisa
Department of Mathematics, Computer Science and Physics, University of Udine, Italy

Ivan Scagnetto
Department of Mathematics, Computer Science and Physics, University of Udine, Italy

Abstract

In [19], the authors have shown that *linear application* in Geometry of Interaction (GoI) models of λ-calculus amounts to *resolution* between *principal types* of linear λ-terms. This analogy also works in the reverse direction. Indeed, an alternative definition of *unification* between algebraic terms can be given by viewing the terms to be unified as *strategies*, *i.e.* sets of pairs of occurrences of the same variable, and verifying the termination of the GoI interaction obtained by playing the two strategies. In this paper we prove that such a criterion of unification is equivalent to the standard one. It can be viewed as a local, bottom-up, definition of unification. Dually, it can be understood as the GoI interpretation of unification.

The proof requires generalizing earlier work to arbitrary algebraic constructors and allowing for multiple occurrences of the same variable in terms. In particular, we show that two terms σ and τ unify if and only if $\mathcal{R}(\sigma) \mathbin{\widehat{\subseteq}} \mathcal{R}(\tau)\mathbin{\widehat{;}}(\mathcal{R}(\sigma)\mathbin{\widehat{;}}\mathcal{R}(\tau))^*$ and $\mathcal{R}(\tau) \mathbin{\widehat{\subseteq}} \mathcal{R}(\sigma)\mathbin{\widehat{;}}(\mathcal{R}(\tau)\mathbin{\widehat{;}}\mathcal{R}(\sigma))^*$, where $\mathcal{R}(\sigma)$ denotes the set of pairs of paths leading to the same variable in the term σ, $\mathbin{\widehat{\subseteq}}$ denotes "inclusion up to substitution" and $\mathbin{\widehat{;}}$ denotes "composition up to substitution".

2012 ACM Subject Classification Theory of computation → Program semantics; Theory of computation → Linear logic

Keywords and phrases unification, geometry of interaction, games

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2024.26

Funding Work supported by project PNRR SERICS PE00000014 M4C2I1.3 – SECCO – "SecCo-OC – Secure Containers Open Call" – CUP D33C22001300002, and by project PNRR SERICS M4C2I1.3 – COVERT OpenCall In searCh Of eVidence of stEalth cybeR Threats – CUP G23C24000790006.

Acknowledgements The authors express their gratitude to the referees for their useful comments.

1 Introduction

Geometry of Interaction (GoI) is a general programme originated by J.-Y. Girard in the late 80's, which aims at giving language-independent mathematical models of algorithms. An extraordinary amount of results by dozens of logicians and computer scientists has been triggered by Girard's seminal papers [13, 14, 15] on the GoI-dynamics of cut-elimination in Linear Logic. Among these arose Game Semantics, as initiated by Abramsky and Hyland, which led to far reaching connections through notions such as traced monoidal categories. Inspired by [2, 3] and in particular by [1], in [19] the authors gave a simple explanation of *GoI-linear application* in GoI linear models of λ-calculus in terms of *unification*, more specifically *resolution*, between *principal types* of linear λ-terms (see also [10, 9, 18, 11, 17]). The notion of GoI-application was hitherto justified in terms of more complex mathematical notions, such as proof nets, C*-algebras, and categorical trace operators.

In this paper we show that the correspondence between GoI linear application and unification is intrisic. Indeed, we introduce an alternative criterion for *unification* between algebraic terms, by viewing the terms to be unified as *strategies*, *i.e.* sets of pairs of occurrences

of the same variable, taken as moves, and checking at the termination of the GoI interaction between the two strategies. We prove that such a criterion of unification is equivalent to the standard one. It can be viewed as a local, bottom-up definition of unification w.r.t. the more top-down approach used in ordinary unification algorithms. But more importantly, this can be viewed, dually, as a step in Girard's GoI programme, namely as the GoI-interpretation of the unification algorithm.

The proof requires generalizing earlier work to arbitrary algebraic constructors and allowing for multiple occurrences of the same variable. In particular, we show that two terms σ and τ unify if and only if

$$\mathcal{R}(\sigma) \;\widehat{\subseteq}\; \mathcal{R}(\tau)\,\widehat{;}\,(\mathcal{R}(\sigma)\,\widehat{;}\,\mathcal{R}(\tau))^* \quad \text{and} \quad \mathcal{R}(\tau) \;\widehat{\subseteq}\; \mathcal{R}(\sigma)\,\widehat{;}\,(\mathcal{R}(\tau)\,\widehat{;}\,\mathcal{R}(\sigma))^*$$

where $\mathcal{R}(\sigma)$ denotes the set of pairs of paths leading to the same variable, *i.e.* occurrences of the same variable, in the term σ, $\widehat{\subseteq}$ denotes "inclusion up to substitution", and $\widehat{;}$ denotes "composition up to substitution".

Since we view terms as strategies, in the form of irreflexive binary relations on variable occurrences, it is easier for expository purposes to discuss first the case of terms with no constants and no *hapax variables*, *i.e.* variables which occur only once. Moreover, since variables merely play the role of placeholders within a term, it is easier to discuss first the case where the terms to be unified do not share variables. Later we remove all these restrictions, by showing that the case of general terms can be reduced to the initial restricted one.

Summing up, unification can be carried out either in a traditional *top down* fashion, taking the view of terms, as in *e.g.* [20], or *bottom up* taking the view of variable occurrences. We will call the former simply *unification*, while we will call the latter *GoI-unification*.

Related work. The use of unification and resolution to build or explain GoI models has a long history. J.-Y. Girard, in [14], first pointed out the close connection between GoI and resolution. A number of authors pursued this investigation, *e.g.* [8], and most notably M.Bagnol in his thesis [7], and subsequent papers on the more recent line of investigation of "Transcendental syntax", which has been introduced by Girard (see [16]) and further investigated in B.Eng's thesis [12]. While in these works unification and resolution are used to build GoI models, in the present paper the point of view is different. Namely, here the goal is to do the converse, *i.e.* to provide a new perspective on unification through a GoI-like mechanism.

Finally, we mention the line of research in Geometry of Interaction which has been probably the most active and fruitful in the last one or two decades, *i.e.* that of GoI based abstract machines (see *e.g.* [4, 6, 5]), which have been studied in particular in connection to time and space efficiency of the computational model of λ-calculus.

Summary. The paper is organized as follows. In Section 2 we introduce preliminary definitions. In Section 3 we discuss standard unification. In Section 4 we give the main definition of when two terms *GoI-unify*, and state the main results for terms with no constants and no hapax variables, and under the assumption that the terms to unify do not share variables. In Section 5 we give examples. In Section 6 we outline the proofs of the main results. In Section 7 we discuss the extension of GoI-unification to general terms. Final remarks and possible directions for future work appear in Section 8.

2 Preliminaries

Throughout the paper we assume the following definitions.

▶ **Definition 1** (Terms and Variable Occurrences).

(i) Terms T_Σ *are given by a* signature $\Sigma = \bigcup_{i=0}^{n} \Sigma_i$, *where* $n > 0$, *and* Σ_i *are sets of* i*-ary constructors* f_i^j, *for* $j = 1, \ldots, |\Sigma_i|$. *Terms are trees whose leaves are variables* $\alpha, \beta, \ldots \in TVar$, *and nodes are operations in* Σ, *i.e.*

$$(T_\Sigma \ni)\ \sigma, \tau ::= \ \alpha \mid \beta \mid \ \ldots \ \mid f_i^j(\sigma_1, \ldots, \sigma_i).$$

(ii) *Let* T_Σ^- *be the subset of terms in* T_Σ *with no 0-ary constructors and no* hapax variables, *i.e. variables which occur only once.*

(iii) Occurrences *of variables in terms, or* occurrence terms, *are given by the following grammar:*

$$(O_\Sigma \ni)\ u[\alpha] ::= \alpha \mid (f_i^j, k)u[\alpha] \qquad \text{for } k \leq i,$$

where

- α *denotes the occurrence of the variable* α *in the term* α,
- *if* $u[\alpha]$ *denotes an occurrence of* α *in* σ, *then* $(f_i^j, k)u[\alpha]$ *denotes the corresponding occurrence of* α *in* $f_i^j(\underbrace{\sigma_1, \ldots, \sigma_{k-1}}_{k-1}, \sigma, \underbrace{\sigma_{k+1}, \ldots, \sigma_i}_{i-k})$.

(iv) *The (possibly empty)* path *of an occurrence* $u[\alpha]$ *is* u.

(v) *Let* $\mathcal{R}(O_\Sigma)$ *denote the set of binary relations on* O_Σ *containing only pairs of the shape* $\langle u[\alpha], v[\alpha] \rangle$, *for* $\alpha \in TVar$, *and* u, v *occurrence paths.*

(vi) *Given a relation* $\mathcal{R}$, *we denote by* $\mathcal{R}^+$ *the maximal irreflexive subrelation included in the symmetric and transitive closure of* $\mathcal{R}$.

▶ **Notation 2.**

- *Syntactical identity of terms is denoted by* $\equiv$.
- *We consider terms up to injective renaming of variables and denote term equality by* $=$.

▶ **Example 3.** Let f_1^1 and f_2^1 be a unary and a binary constructor, respectively, and let $\sigma = f_2^1(\alpha, f_1^1(\alpha))$. Then σ has two occurrences of the variable α, *i.e.* $(f_2^1, 1)\,\alpha$ and $(f_2^1, 2)(f_1^1, 1)\,\alpha$, with corresponding paths $u = (f_2^1, 1)$ and $v = (f_2^1, 2)(f_1^1, 1)$. ⌟

Terms give rise to *non-deterministic strategies* for a game where *moves* are variable occurrences, in the following sense:

▶ **Definition 4** (From Terms to Strategies).

(i) *A term* τ *gives rise to a set of* variable occurrences (moves)*:*

$$\mathcal{O}(\tau) = \{u[\alpha] \mid u[\alpha] \text{ is an occurrence of } \alpha \text{ in } \tau\}.$$

(ii) *A term* τ *gives rise to a symmetric and transitive irreflexive relation on* O_Σ (strategy)*:*

$$\mathcal{R}(\tau) = \{\langle u[\alpha], v[\alpha] \rangle \mid u[\alpha], v[\alpha] \text{ are different occurrences of } \alpha \text{ in } \tau\}.$$

(iii) *Two different variable occurrences* $u[\alpha], v[\beta]$ *are* compatible *if* $u = w(f_i^j, k)u'$ *and* $v = w(f_i^j, h)v'$, *for some paths* w, u', v', *and indexes* i, j, $k \neq h$.

▶ **Example 5.** Let f_1^1 and f_2^1 be a unary and a binary constructor, respectively. Then the variable occurrences $(f_2^1, 1)(f_2^1, 1)(f_1^1, 1)\,\alpha$ and $(f_2^1, 1)(f_2^1, 2)\,\beta$ are compatible, while the occurrences $(f_2^1, 1)(f_2^1, 1)(f_1^1, 1)\,\alpha$ and $(f_2^1, 1)(f_2^1, 1)\,\beta$ are not. ⌟

For a given term τ, it is immediate that all variable occurrences in $\mathcal{O}(\tau)$ are pairwise compatible. Vice versa, the compatibility condition permits us to build a term from a set of pairwise compatible variable occurrences, by tagging some nodes of the term tree with fresh variables, if needed.

▶ **Proposition 6.**

(i) *Given a set $\mathcal{S}$ of pairwise compatible variable occurrences, we can build the tree of a term, by tagging possible missing leaves with fresh variables in $Z = \{\zeta_1, \dots, \zeta_i, \dots\}$, i.e.*

$$\mathcal{T}_Z(\mathcal{S}) = \begin{cases} \zeta & \text{if } \mathcal{S} = \emptyset, \zeta \in Z \text{ fresh} \\ \alpha & \text{if } \mathcal{S} = \{\alpha\} \\ f_i^j(\mathcal{T}_Z(\{u[\alpha] \mid (f_i^j, 1)u[\alpha] \in \mathcal{S}\}), \dots, \mathcal{T}_Z(\{u[\alpha] \mid (f_i^j, i)u[\alpha] \in \mathcal{S}\})) & \text{if } \forall v[\alpha] \in \mathcal{S}\ \exists k, v' \text{ s.t. } v = (f_i^j, k)v' \end{cases}$$

(ii) *Let σ be a term with no 0-ary constructors. Then we have $\mathcal{T}_Z(\mathcal{O}(\sigma)) = \sigma$.*

Proof. By induction on the structure of terms. The compatibility condition ensures that at any step all occurrence paths start with the same constructor f_i^j. ◀

Notice that, in Proposition 6(ii) above we consider terms without constants. In principle, one could extend this result to the whole set of terms, by suitably generalizing Definition 4 and introducing a notion of *constant occurrence*. However, we prefer to follow a different approach, which allows for a simpler theory: we first introduce and study the notion of GoI-unification for terms without constants, and then, in Section 7, we show how the general case can be dealt with via a simple encoding into the restricted case.

3 The Top-down Perspective on Unification

First we fix notations for standard notions.

▶ **Definition 7** (Terms Unifiers). *Let σ and τ be terms.*

(i) *A* substitution *is a function $U : T_\Sigma \to T_\Sigma$ defined inductively on the structure of terms from a* variable substitution *$U_{Var} : TVar \to T_\Sigma$, i.e.*
- $U(\alpha) = U_{Var}(\alpha)$
- $U(f_i^j(\sigma_1, \dots, \sigma_i)) = f_i^j(U(\sigma_1), \dots, U(\sigma_i))$.

(ii) *A* unifier *for σ and τ is a substitution U which differs from the identity on a finite number of variables and such that $U(\sigma) = U(\tau)$. We call* domain *of U, $dom(U)$, the finite set of variables on which U is not the identity.*

(iii) *Given two substitutions, U and V, the composition $U;V$ is defined as usual, i.e. $U;V = V \circ U$.*

(iv) *Given two substitutions, U and V, we define $U \leq V$ if there exists a substitution U' such that $U;U' = V$, i.e. U is more general than V.*

(v) *Given terms σ, τ, a* most general unifier (m.g.u.) *of σ, τ is a unifier $\overline{U}$ of σ and τ such that, for any unifier U of σ and τ, $\overline{U} \leq U$.*

We introduce now a unification algorithm *à la* Martelli Montanari, [20], which unifies simultaneously sets of pairs of terms.

▶ **Definition 8** (Martelli-Montanari's Algorithm). *Let E be a set of pairs of the form $\langle \sigma, \tau \rangle$, for $\sigma, \tau \in T_\Sigma$. The unification algorithm is defined by the following rules for deriving judgements of the form $\mathcal{U}(E)$:*

(i) $\mathcal{U}(\{\langle f_i^j(\sigma_1,\ldots,\sigma_i), f_i^j(\tau_1,\ldots,\tau_i)\rangle\} \cup E) \to \mathcal{U}(\{\langle\sigma_1,\tau_1\rangle,\ldots,\langle\sigma_i,\tau_i\rangle\} \cup E)$
(ii) $\mathcal{U}(\{\langle\alpha,\alpha\rangle\} \cup E) \to \mathcal{U}(E)$
(iii) $\mathcal{U}(\{\langle f_i^j(\sigma_1,\ldots,\sigma_i),\alpha\rangle\} \cup E) \to \mathcal{U}(\{\langle\alpha, f_i^j(\sigma_1,\ldots,\sigma_i)\rangle\} \cup E)$
(iv) $\mathcal{U}(\{\langle\alpha,\sigma\rangle\} \cup E) \to \mathcal{U}(\{\langle\alpha,\sigma\rangle\} \cup E[\sigma/\alpha]),$ *if* $\alpha \notin \sigma \wedge \alpha \in Var(E)$
(v) $\mathcal{U}(\{\langle\alpha,\sigma\rangle\} \cup E) \to$ *fail, if* $\alpha \in \sigma \ \wedge \ \alpha \neq \sigma$
(vi) $\mathcal{U}(\{\langle f_i^j(\sigma_1,\ldots,\sigma_i), f_k^l(\tau_1,\ldots,\tau_k)\rangle\} \cup E \to$ *fail, if* $i \neq k$ *or* $j \neq l$

The algorithm in Definition 8 is non-deterministic. Moreover, notice that rule (iii) is not symmetric and that rule (iv) produces a set of pairs of terms to be unified, where there is only one occurrence of the variable α left.

In particular, termination of the algorithm is guaranteed by the fact that, by applying any of the rules, the complexity of the set of pairs E decreases, according to the following measure:

▶ **Definition 9.** *Let E be a set of pairs of the form $\langle\sigma,\tau\rangle$. We define a complexity measure of E by $m(E) = (m_t, m_v)$, where*

- *m_t is the sum of the complexities of terms in the lefthand parts of the pairs,*
- *m_v is the number of variables appearing in E, except variables α's for which there is only one pair $\langle\alpha,\sigma\rangle$ such that $\alpha \notin \sigma$ and $\alpha \notin E \setminus \{\langle\alpha,\sigma\rangle\}$.*

Then, the following lemma is straightforward.

▶ **Lemma 10.** *Given terms $\sigma,\tau \in T_\Sigma$, Martelli-Montanari's Algorithm terminates either with failure or with a set of pairs yielding a substitution.*

The following proposition is well-known.

▶ **Proposition 11** (Most General Unifier)**.** *Let $\sigma,\tau \in T_\Sigma$. The terms σ and τ unify if and only if Martelli-Montanari's algorithm on $\{\langle\sigma,\tau\rangle\}$ terminates with a substitution E. Then $\{\alpha \mapsto \rho_\alpha \mid \langle\alpha,\rho_\alpha\rangle \in E\}$ is a m.g.u. of σ and τ, which is unique up to appropriate injective renaming of variables.*

4 GoI-unification: the Bottom-up Perspective of Paths

In this section we provide a GoI account of unification. Namely, as done in [19], we utilize the machinery underlying process interaction in *game semantics* to explain the dynamics of unification. In this perspective, terms are viewed as non-deterministic strategies over the language of moves consisting of variable occurrences (see Definition 4). The proofs of the results in this section appear in Section 6. More specifically, we provide a GoI account of unification for a restricted class of terms, *i.e.* terms with no constants and no *hapax variables*. Moreover, we assume that the two terms to unify do not share variables. In Section 7 all these restrictions are removed, namely we show that the general case can be always reduced to the special case discussed in the previous sections.

Now we need to introduce the notions of *occ-substitution* and *occ-unifier*:

▶ **Definition 12** (Occ-unifier)**.**

(i) *An* occ-substitution *is a function $M : O_\Sigma \to O_\Sigma$ defined inductively on the structure of variable occurrences, starting from a substitution for variables $M_{Var} : TVar \to O_\Sigma$, i.e.*

- *$M(\alpha) = M_{Var}(\alpha)$*
- *$M((f_i^j,k)u[\alpha]) = (f_i^j,k)M(u[\alpha])$.*

(ii) *An* occ-unifier *of variable occurrences* $u[\alpha], v[\beta]$ *is an occ-substitution* M *such that* $M(u[\alpha]) = M(v[\beta])$.

Notice that occ-unifiers are actually *matchings*, namely two variable occurrences $u[\alpha], v[\beta]$ unify if and only if u is a prefix of v or v is a prefix of u:

▶ **Lemma 13.** *Let* $u[\alpha], v[\beta] \in O_\Sigma$. *The following are equivalent:*
(i) *there exists an occ-unifier of* $u[\alpha]$ *and* $v[\beta]$*;*
(ii) *there exists an occurrence path* w *such that* $u = vw$ *or* $v = uw$.

The following definition introduces a special operation on relations on variable occurrences, which will be used in the definition of GoI-unification, *i.e.* the operation of *composition up to occ-substitution*, together with the notion of *inclusion up to substitution*:

▶ **Definition 14.** *Let* $\mathcal{R}, \mathcal{R}', \mathcal{R}'' \in \mathcal{R}(O_\Sigma)$ *be binary relations on variable occurrences.*
(i) *We define* $\mathcal{R}(\sigma)\widehat{;}\mathcal{R}(\tau)$ *as* composition up to occ-substitution, *namely*

$$\langle u''[\gamma], v''[\gamma]\rangle \in \mathcal{R}(\sigma)\widehat{;}\mathcal{R}(\tau) \quad \textit{if and only if}$$
$$\exists\ \langle u[\alpha], v[\alpha]\rangle \in \mathcal{R}(\sigma),\ \langle u'[\beta], v'[\beta]\rangle \in \mathcal{R}(\tau)\ \textit{and}$$
$$\exists M\ \textit{occ-unifier of}\ v[\alpha], u'[\beta]\ \textit{s.t.}\ (M(u''[\gamma]) = M(u[\alpha])\ \wedge\ M(v''[\gamma]) = M(v'[\beta]))\ .$$

(ii) *We define the predicate* inclusion up to substitution, $\mathcal{R} \mathrel{\widehat{\subseteq}} \mathcal{R}'$, *which holds if and only if for each* $\langle u[\alpha], v[\alpha]\rangle \in \mathcal{R}$ *there exists a pair* $\langle u'[\beta], v'[\beta]\rangle \in \mathcal{R}'$ *and an occ-substitution* M *such that* $M(u[\alpha]) = M(u'[\beta]) \wedge\ M(v[\alpha]) = M(v'[\beta])$.

One can easily check (by case analysis on the occ-substitutions) that Definition 14(i) above determines an associative operation on $\mathcal{R}(O_\Sigma)$:

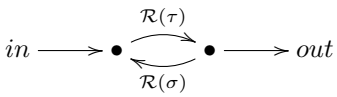
▶ **Lemma 15.** *Composition up to occ-substitution of relations is an associative operation over* $\mathcal{R}(O_\Sigma)$.

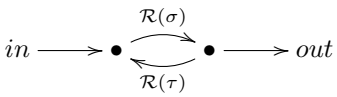
The following is the crucial definition of the paper:

▶ **Definition 16** (GoI-unification). *Let* $\sigma, \tau \in T_\Sigma^-$ *be such that* $Var(\sigma) \cap Var(\tau) = \emptyset$. *Then* σ *and* τ GoI-unify, *i.e.* $\mathcal{U}_{GoI}(\sigma, \tau)$, *if and only if*

$$\mathcal{R}(\sigma) \mathrel{\widehat{\subseteq}} \mathcal{R}(\tau)\,\widehat{;}\,(\mathcal{R}(\sigma)\,\widehat{;}\,\mathcal{R}(\tau))^* \quad \textit{and} \quad \mathcal{R}(\tau) \mathrel{\widehat{\subseteq}} \mathcal{R}(\sigma)\,\widehat{;}\,(\mathcal{R}(\tau)\,\widehat{;}\,\mathcal{R}(\sigma))^*\ ,$$

where, for a relation $\mathcal{R}$, $\mathcal{R}^*$ *denotes the relation* $Id_{O_\Sigma} \cup \bigcup_{n\geq 1} \mathcal{R}^n$, *where* Id_{O_Σ} *denotes the identity relation on* O_Σ *and* $\mathcal{R}^n$ *the n-ary composition up to substitution of* $\mathcal{R}$.
The "flow of control" in checking the two predicates is outlined in the following diagrams





Notice the similarity of the above definition with Girard's Execution Formula, and hence the connection with game semantics.

The following is the main result of the paper, whose proof will be given in Section 6:

▶ **Theorem 17** (Equivalence). *Let* $\sigma, \tau \in T_\Sigma^-$ *be such that* $Var(\sigma) \cap Var(\tau) = \emptyset$. *Then* σ *and* τ unify *if and only if they* GoI-unify.

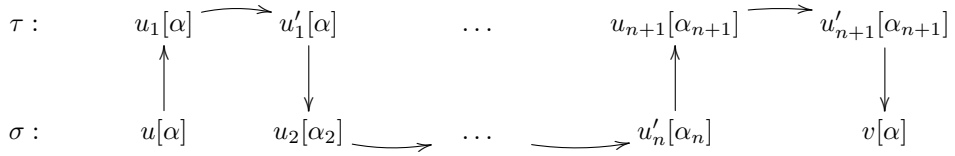


Figure 1 GoI-execution sequence.

The following lemma is very useful in verifying the criterion of Definition 16. It essentially amounts to spelling out that definition, once we observe that the order in which occ-unifications are performed in a composition chain is irrelevant. This in turn follows from Lemma 15.

▶ **Lemma 18.** *Let $\sigma, \tau \in T^{-}_{\Sigma}$ be such that $Var(\sigma) \cap Var(\tau) = \emptyset$. Then σ and τ* GoI-unify *($\mathcal{U}_{GoI}(\sigma, \tau)$) if and only if, for any $\langle u[\alpha], v[\alpha]\rangle \in \mathcal{R}(\sigma)$,*

- **(i)** *there exist a sequence* of odd length ≥ 1 $\langle u_1[\alpha_1], u'_1[\alpha_1]\rangle, \ldots, \langle u_{n+1}[\alpha_{n+1}], u'_{n+1}[\alpha_{n+1}]\rangle$, $n \geq 0$, *and*
- **(ii)** *occ-substitutions $M_1, \ldots, M_n, M_{\langle u[\alpha], v[\alpha]\rangle}$ such that*
  - *$\langle u_i[\alpha_i], u'_i[\alpha_i]\rangle \in \mathcal{R}(\tau)$, for i odd;*
  - *$\langle u_i[\alpha_i], u'_i[\alpha_i]\rangle \in \mathcal{R}(\sigma)$, for i even;*
  - *for all $1 \leq i \leq n$, $(M_1; \ldots; M_i)(u'_i[\alpha_i]) = M_i(u_{i+1}[\alpha_{i+1}])$;*
  - *$M_{\langle u[\alpha], v[\alpha]\rangle}(u[\alpha]) = (M_1; \ldots; M_n)(u_1[\alpha_1])$;*
  - *$M_{\langle u[\alpha], v[\alpha]\rangle}(v[\alpha]) = M_n(u'_{n+1}[\alpha_{n+1}])$.*

And vice versa, for any $\langle u[\alpha], v[\alpha]\rangle \in \mathcal{R}(\tau)$ there exists a sequence of pairs of variable occurrences and occ-substitutions satisfying the properties above.
We call GoI-execution sequence *the above sequence of pairs of variable occurrences together with the corresponding occ-substitutions.*

The GoI-execution sequence mimicking the pair $\langle u[\alpha], v[\alpha]\rangle \in \mathcal{R}(\sigma)$ can be visualized with a diagram as in Figure 1, where vertical arrows denote occ-unifications (which are left implicit), while horizontal ones connect pairs in $\mathcal{R}(\sigma)$ or in $\mathcal{R}(\tau)$.

The result in Theorem 17 is far from straightforward, since the two perspectives on unification, namely the one implemented in the algorithm $\mathcal{U}$, introduced in Definition 8, and the one given by the flow of control in dealing with pairs in $\mathcal{R}(\sigma)$ and $\mathcal{R}(\tau)$ in Definition 16 are conceptually different.

In Section 8, we shall briefly outline how term unifiers can be derived from GoI-execution sequences, thereby suggesting the analogue of Proposition 11.

5 GoI-unification at Work

In this section we illustrate positive examples and failures of the unification criterion given by Definition 16. But before doing this we need to introduce some notations.

▶ **Notation 19.** *To ease intuition, when dealing with a single binary constructor f^1_2, terms will be called linear types, moreover we shall use the infix operator $\multimap$ for f^1_2, and we shall denote the paths $(f^1_2, 1)$ and $(f^1_2, 2)$ simply by l and r, respectively.*

▶ **Example 20.** Consider the two linear types $\sigma \equiv \alpha \multimap \alpha$ and $\tau \equiv (\beta \multimap \beta) \multimap (\beta \multimap \beta)$. These two terms clearly unify by taking the variable substitution $\{\alpha \mapsto \beta \multimap \beta\}$. We have $\mathcal{R}(\sigma) = \{\langle l\alpha, r\alpha\rangle\}^+$ and $\mathcal{R}(\tau) = \{\langle ll\beta, lr\beta\rangle, \langle rl\beta, rr\beta\rangle \langle lr\beta, rl\beta\rangle\}^+$. For each of the four pairs we provide appropriate GoI-execution sequences, where the occ-substitutions have been fully applied:

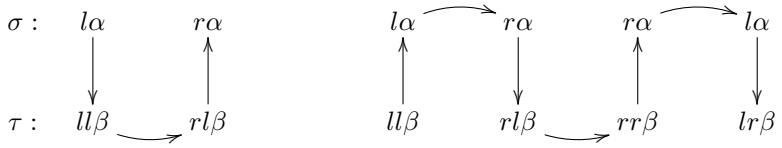


Figure 2 Example 20.

(i) $\langle l\alpha, r\alpha\rangle \rightsquigarrow ll\beta \overset{\tau}{\Leftrightarrow} rl\beta$
(ii) $\langle ll\beta, lr\beta\rangle \rightsquigarrow ll\beta \overset{\sigma}{\Leftrightarrow} rl\beta \overset{\tau}{\Leftrightarrow} rr\beta \overset{\sigma}{\Leftrightarrow} lr\beta$
(iii) $\langle rl\beta, rr\beta\rangle \rightsquigarrow rl\beta \overset{\sigma}{\Leftrightarrow} ll\beta \overset{\tau}{\Leftrightarrow} lr\beta \overset{\sigma}{\Leftrightarrow} rr\beta$
(iv) $\langle lr\beta, rl\beta\rangle \rightsquigarrow lr\beta \overset{\sigma}{\Leftrightarrow} rr\beta \overset{\tau}{\Leftrightarrow} ll\beta \overset{\sigma}{\Leftrightarrow} rl\beta$

We compute explicitly all the substitutions M's arising according Lemma 18 in the case (ii), *i.e.* for the pair $\langle ll\beta, lr\beta\rangle$. The sequence of pairs of variable occurrences are $\langle l\alpha, r\alpha\rangle, \langle rl\beta, ll\beta\rangle, \langle l\alpha, r\alpha\rangle$. The substitutions are:

- $M_1 : \alpha \mapsto r\beta$, which is used to match $r\alpha$ in the second component of the first pair and $rl\beta$ in the first component of the second pair;
- $M_2 : \alpha \mapsto l\beta$, which matches the second component of the second pair (after the substitution with M_1), and the first component of the third pair;
- finally we have $M_{\langle ll\beta, lr\beta\rangle} = Id$.

In Figure 2, the left-hand diagram represents the GoI sequence for the pair $\langle l\alpha, r\alpha\rangle$, while the right-hand diagram represents the GoI sequence for the pair $\langle ll\beta, lr\beta\rangle$. In the diagrams of Figure 2, vertical arrows denote occ-unifications (which are left implicit), while horizontal ones connect pairs in $\mathcal{R}(\sigma)$ or in $\mathcal{R}(\tau)$. In the present cases, the GoI-execution sequences can be directly read in the bottom lines of the two diagrams. ⌟

▶ **Example 21.** Consider the terms $\sigma \equiv f(\alpha, \alpha, \alpha)$ and $\tau \equiv f(\beta, g(\gamma, \gamma), \beta)$, where f is a ternary constructor and g is binary. We have that $\mathcal{R}(\sigma) = \{\langle (f,1)\alpha, (f,2)\alpha\rangle, \langle (f,1)\alpha, (f,3)\alpha\rangle, \langle (f,2)\alpha, (f,3)\alpha\rangle\}^+$, while $\mathcal{R}(\tau) = \{\langle (f,1)\beta, (f,3)\beta\rangle, \langle (f,2)(g,1)\gamma, (f,2)(g,2)\gamma\rangle\}^+$. These two terms clearly unify by taking the variable substitution $\{\alpha \mapsto g(\gamma,\gamma), \beta \mapsto g(\gamma,\gamma)\}$. We provide appropriate GoI-execution sequences for some of the pairs in the $\mathcal{R}(\ )$'s, the other pairs are dealt with symmetrically:

$\langle (f,1)\alpha, (f,3)\alpha\rangle \rightsquigarrow (f,1)\beta \overset{\tau}{\Leftrightarrow} (f,3)\beta$ $\qquad$ $\langle (f,1)\beta, (f,3)\beta\rangle \rightsquigarrow (f,1)\alpha \overset{\sigma}{\Leftrightarrow} (f,3)\alpha$

$\langle (f,1)\alpha, (f,2)\alpha\rangle \rightsquigarrow (f,1)(g,1)\gamma \overset{\tau}{\Leftrightarrow} (f,3)(g,1)\gamma \overset{\sigma}{\Leftrightarrow} (f,2)(g,1)\gamma \overset{\tau}{\Leftrightarrow} (f,2)(g,2)\gamma \overset{\sigma}{\Leftrightarrow} (f,1)(g,2)\gamma \overset{\tau}{\Leftrightarrow} (f,3)(g,2)\gamma \overset{\sigma}{\Leftrightarrow} (f,2)(g,2)\gamma \overset{\tau}{\Leftrightarrow} (f,2)(g,1)\gamma$

$\langle (f,2)(g,1)\gamma, (f,2)(g,2)\gamma\rangle \rightsquigarrow (f,2)(g,1)\gamma \overset{\sigma}{\Leftrightarrow} (f,3)(g,1)\gamma \overset{\tau}{\Leftrightarrow} (f,1)(g,1)\gamma \overset{\sigma}{\Leftrightarrow} (f,2)(g,1)\gamma \overset{\tau}{\Leftrightarrow} (f,2)(g,2)\gamma \overset{\sigma}{\Leftrightarrow} (f,3)(g,2)\gamma \overset{\tau}{\Leftrightarrow} (f,1)(g,2)\gamma \overset{\sigma}{\Leftrightarrow} (f,2)(g,2)\gamma.$ ⌟

▶ **Example 22.** Consider the two linear types $\sigma \equiv \alpha \multimap \alpha$ and $\tau \equiv \gamma \multimap ((\beta \multimap \beta) \multimap \gamma)$. Apparently the two terms do not unify. We have $\mathcal{R}(\sigma) = \{\langle l\alpha, r\alpha\rangle\}^+$ and $\mathcal{R}(\tau) = \{\langle l\gamma, rrr\gamma\rangle, \langle rll\beta, rlr\beta\rangle\}^+$. In order to achieve an instance of the pair $\langle rll\beta, rlr\beta\rangle$ from a GoI-execution sequence both the first and last term must start with an r and be the first and second component respectively of pairs in $\mathcal{R}(\sigma)$. We show that there cannot be any GoI-sequence where both the first and the last terms start with an r. We proceed by contradiction. Assume that such a sequence exists and consider the shortest one. The other components of such pairs must then be occurrence terms starting with an l. These in turn unify with occurrence terms coming from pairs in $\mathcal{R}(\tau)$, so their other component must start with an r. Hence the GoI-execution sequence we started from was not the shortest. Contradiction. ⌟

▶ **Example 23.** Consider the two linear types $\sigma \equiv (\alpha \multimap \alpha) \to \alpha$ and $\tau \equiv (\alpha \multimap \alpha) \multimap \alpha$, where we two binary constructors $\multimap$ and $\to$ appear, and whose occurrence components are denoted respectively by l, r and L, R. We have that $\mathcal{R}(\sigma) = \{\langle Ll\alpha, Lr\alpha\rangle, \langle Ll\alpha, R\alpha\rangle, \langle lr\alpha, r\alpha\rangle\}^+$ while $\mathcal{R}(\tau) = \{\langle ll\alpha, rl\alpha\rangle, \langle ll\alpha, r\alpha\rangle, \langle lr\alpha, r\alpha\rangle\}^+$. One can readily check that no GoI execution sequence can simulate $\langle Ll\alpha, Lr\alpha\rangle$. ⌟

6 Proof of the Main Theorem

For simplicity we shall sketch the proof only for a single binary constructor $(\_ \to \_)$, which we write in infix form. We need some new notation and a number of *invariance* lemmata, whose proofs are essentially straightforward from the definitions.

▶ **Notation 24.** *Let σ be a term where all the variables occur exactly twice. When we want to highlight one or more pairs of occurrences of some of the variables, we write $\sigma[(\alpha_i, \alpha_i)]$ or $\sigma[(\alpha_1, \alpha_1), \ldots, (\alpha_n, \alpha_n)]$. Moreover we write $\sigma[(\tau_1, \rho_1)]$ when we denote the term obtained from σ by replacing the first occurrence of the variable α, which should be clear form the context, in σ with τ_1 and the second occurrence of α in σ with ρ_1, and we write $\sigma[(\tau_1, \rho_1), \ldots (\tau_n, \rho_n))]$ to denote the term obtained from σ by substituting the variables $\alpha_1, \ldots, \alpha_n$. Notice that some pairs in $(\tau_1, \rho_1), \ldots (\tau_n, \rho_n)$ could be equal.*

▶ **Lemma 25.** *Let σ, τ be terms which do not share variables, and let α be a fresh variable. Then*

$$\mathcal{U}_{GoI}(\sigma,\ \tau) \Longleftrightarrow \mathcal{U}_{GoI}(\sigma \to \tau,\ \alpha \to \alpha)\ .$$

The Lemma above holds trivially, if σ and τ do not share variables. Otherwise, the righthand side can be taken as the definition of $\mathcal{U}_{GoI}(\sigma,\ \tau)$ when the two terms share variables (see Section 7.2 below).

▶ **Lemma 26.** *Let σ be a term where all variables occur exactly twice, and let $\sigma[(\tau_1, \rho_1), \ldots, (\tau_n, \rho_n)]$ be such that $Var(\sigma[(\tau_1, \rho_1), \ldots, (\tau_n, \rho_n)]) \cap Var(\sigma) = \emptyset$. If $\tau_i \equiv \tau_{i1} \to \tau_{i2}$ and $\rho_i \equiv \rho_{i1} \to \rho_{i2}$, for some $i \in \{1, \ldots, n\}$, then*

$$\mathcal{U}_{GoI}(\sigma[(\tau_1, \rho_1), \ldots, (\tau_i, \rho_i), \ldots, (\tau_n, \rho_n)],\ \sigma) \Leftrightarrow \mathcal{U}_{GoI}(\sigma'[(\tau_1, \rho_1), \ldots, (\tau_{i1}, \rho_{i1}), (\tau_{i2}, \rho_{i2}), \ldots, (\tau_n, \rho_n)],\ \sigma')$$

where $\sigma'[(\alpha_{i1}, \alpha_{i1}), (\alpha_{i2}, \alpha_{i2})] \equiv \sigma[(\alpha_{i1} \to \alpha_{i2}, \alpha_{i1} \to \alpha_{i2})]$, for α_{i1}, α_{i2} fresh variables.

Notice that the application of Lemma 26 only modifies the second term of the pair, *i.e.* σ, but in a way to preserve the property that its variables occur exactly twice. The proof is again straightforward, since execution sequences on the left hand side are immediately reflected on the right hand side, and vice versa.

▶ **Example 27.** Let $\sigma \equiv \alpha \to \alpha$, $\tau \equiv \delta \to (\beta \to \beta) \to \delta$, and $\rho \equiv \gamma \to \gamma \to \gamma$, where $\to$ associates to the right, as usual. Then $\sigma[(\tau, \rho)] \equiv (\delta \to (\beta \to \beta) \to \delta) \to (\gamma \to \gamma \to \gamma)$. We have:

$$\tau \equiv \tau_1 \to \tau_2,\ \text{where } \tau_1 \equiv \delta \text{ and } \tau_2 \equiv (\beta \to \beta) \to \delta\ ,$$

$$\rho \equiv \rho_1 \to \rho_2,\ \text{where } \rho_1 \equiv \gamma \text{ and } \rho_2 \equiv (\gamma \to \gamma)\ .$$

Applying Lemma 26, we get

$$\mathcal{U}_{GoI}(\sigma[(\tau,\rho)],\ \sigma) \iff \mathcal{U}_{GoI}(\sigma'[(\tau_1,\rho_1),(\tau_2,\rho_2)],\sigma')\ ,$$

where $\sigma' \equiv (\alpha_1 \to \alpha_2) \to (\alpha_1 \to \alpha_2)$.
Now both τ_2 and ρ_2 are arrow types, and so we can apply Lemma 26 again, getting:

$$\mathcal{U}_{GoI}(\sigma'[(\tau_1,\rho_1),(\tau_2,\rho_2)],\ \sigma') \iff \mathcal{U}_{GoI}(\sigma''[(\tau_1,\rho_1),(\tau_{21},\rho_{21}),(\tau_{22},\rho_{22})],\ \sigma'')\ ,$$

where

- $\sigma'' \equiv (\alpha_1 \to (\alpha_{21} \to \alpha_{22})) \to (\alpha_1 \to (\alpha_{21} \to \alpha_{22}))$
- $\tau_{21} \equiv \beta \to \beta$ and $\rho_{21} \equiv \gamma$
- $\tau_{22} \equiv \delta$ and $\rho_{22} \equiv \gamma$.

Notice that the pairs (τ_1,ρ_1) and (τ_{22},ρ_{22}) coincide.
At this point Lemma 26 is not applicable anymore. ⌟

▶ **Lemma 28.** *Let σ be a term where all variables occur exactly twice, and let $\sigma[(\tau_1,\rho_1),\dots,(\xi,\rho_i),\dots,$
$(\tau_n,\rho_n)]$ be such that ξ is a variable, and $Var(\sigma[(\tau_1,\rho_1),\dots,(\xi,\rho_i),\dots,(\tau_n,\rho_n)]) \cap Var(\sigma) = \emptyset$. Then,*

- *if $\xi \in Var(\rho_i)$ and $\xi \not\equiv \rho_i$, then $\sigma[(\tau_1,\rho_1),\dots,(\xi,\rho_i),\dots,(\tau_n,\rho_n)]$ and σ are not GoI-unifiable;*
- *if $\xi \notin Var(\rho_i)$,*

$$\mathcal{U}_{GoI}(\sigma[(\tau_1,\rho_1),\dots,(\xi,\rho_i),\dots,(\tau_n,\rho_n)],\ \sigma) \iff \mathcal{U}_{GoI}(\sigma[(\tau_1,\rho_1),\dots,(\xi,\rho_i),\dots,(\tau_n,\rho_n)][\rho_i/\xi_i],\ \sigma)\ .$$

And vice versa, i.e. the statement holds for $\sigma[(\tau_1,\rho_1),\dots,(\tau_i,\xi),\dots,(\tau_n,\rho_n)]$, where the variable ξ appears as second element in a pair.

The only not immediate part of the proof of the above lemma concerns the case where $\xi \in Var(\rho_i)$ and $\xi \not\equiv \rho_i$. For simplicity, we just consider the case where $\sigma \equiv \alpha \to \alpha$. The argument generalizes quite easily. If $\mathcal{U}_{GoI}(\xi \to \rho_i, \alpha \to \alpha)$, there exists a GoI-execution sequence which simulates the occurrence pair $\langle l[\xi], ru[\xi]\rangle$. But whatever substitution which is mediated by $\langle l[\alpha], r[\alpha]\rangle$, it will always produce occ-subsitutions of the same length in both occurrences, thereby preventing any GoI-execution sequence from reproducing the asymmetry in the original pair through an occ-substitution.

▶ **Example 29.** Continuing from Example 27, we can now apply Lemma 28 above to the last two terms $\sigma''[(\tau_1,\rho_1),(\tau_{21},\rho_{21}),(\tau_{22},\rho_{22})]$ and σ'', by considering the first pair (τ_1,ρ_1), and taking, *e.g.*, $\xi \equiv \delta$. Then we get:

$$\mathcal{U}_{GoI}(\sigma''[(\tau_1,\rho_1),(\tau_{21},\rho_{21}),(\tau_{22},\rho_{22})],\ \sigma'') \iff \mathcal{U}_{GoI}(\sigma''[(\tau_1,\rho_1),(\tau_{21},\rho_{21}),(\tau_{22},\rho_{22})][\gamma/\delta],\ \sigma'')$$

where the latter becomes

$$\mathcal{U}_{GoI}((\gamma \to (\beta \to \beta) \to \gamma) \to (\gamma \to \gamma \to \gamma),\ (\alpha_1 \to (\alpha_{21} \to \alpha_{22})) \to (\alpha_1 \to (\alpha_{21} \to \alpha_{22}))\ .$$

Then, applying again Lemma 28 to the pair (τ_{21},ρ_{21}), *i.e.* $(\beta \to \beta, \gamma)$, and performing the corresponding substitution, we finally get

$$\mathcal{U}_{GoI}(((\beta \to \beta) \to (\beta \to \beta) \to (\beta \to \beta)) \to ((\beta \to \beta) \to (\beta \to \beta) \to (\beta \to \beta)), (\alpha_1 \to (\alpha_{21} \to \alpha_{22})) \to (\alpha_1 \to (\alpha_{21} \to \alpha_{22}))\ .$$

Notice that at this point Lemma 26 above becomes again applicable, since the above coincides with

$$\mathcal{U}_{GoI}(\sigma''[((\beta\to\beta),(\beta\to\beta)),((\beta\to\beta),(\beta\to\beta)),((\beta\to\beta),(\beta\to\beta))],\ \sigma'')\ ,$$

where $\sigma'' \equiv (\alpha_1 \to (\alpha_{21} \to \alpha_{22})) \to (\alpha_1 \to (\alpha_{21} \to \alpha_{22}))$. ⌟

Clearly the above Lemmata 25, 26, 28 hold also for $\mathcal{U}$. Hence we can state:

▶ **Lemma 30.** *Lemmata 25, 26, and 28 hold by replacing $\mathcal{U}_{GoI}$ by $\mathcal{U}$.*

Finally, we have:

▶ **Lemma 31.** *Let σ be a term where all its n variables occur exactly twice. Then*

$$\mathcal{U}_{GoI}(\sigma[(\beta_1,\beta_1),\dots,(\beta_n,\beta_n)],\ \sigma) \Longleftrightarrow \mathcal{U}(\sigma[(\beta_1,\beta_1),\dots,(\beta_n,\beta_n)],\ \sigma)$$

where all the β_i's are fresh, but some of them can be equal.

Now we are ready to prove the main Theorem 17. Namely, we can proceed as follows.

6.1 GoI-unification $\Longrightarrow$ Unification

Assume σ, τ do not share common variables and $\mathcal{U}_{GoI}(\sigma, \tau)$. Then, by Lemma 25, $\mathcal{U}_{GoI}(\sigma \to \tau, \alpha \to \alpha)$, for α fresh variable. Now we repeatedly apply Lemmata 26 or 28 in the $(\Rightarrow)$-direction, until neither is applicable anymore. This procedure is guaranteed to terminate, because the complexity of the pair of terms to GoI-unify decreases at each step, according to the following definition of complexity measure:

▶ **Definition 32.** *Let σ be a term where all its n variables occur exactly twice, and let $\sigma[(\tau_1,\rho_1),\dots,(\tau_n,\rho_n)]$ be such that $Var(\sigma[(\tau_1,\rho_1),\dots,(\tau_n,\rho_n)]) \cap Var(\sigma) = \emptyset$. We define the complexity of the pair of terms $(\sigma[(\tau_1,\rho_1),\dots,(\tau_n,\rho_n)],\sigma)$ as the pair (m_t, m_v), where*

- *m_t is the sum of the complexities of the terms $\tau_1,\dots,\tau_n,\rho_1,\dots,\rho_n$,*
- *m_v is the number of variables appearing in $(\tau_1,\rho_1),\dots,(\tau_n,\rho_n)$.*

Now notice that the application of Lemmata 26 and 28 in the $(\Rightarrow)$-direction decreases the complexity of the pair of terms.
Hence, by repeatedly applying these lemmata, we reach a pair of the shape $(\overline{\sigma}[(\beta_1,\beta_1),\dots,(\beta_n,\beta_n)],\overline{\sigma})$, where $\overline{\sigma}$ has n variables and all the β_i's are fresh (some of them can possibly coincide). Therefore Lemma 31 applies, and we finally get $\mathcal{U}(\overline{\sigma}[(\beta_1,\beta_1),\dots,(\beta_n,\beta_n)],\overline{\sigma})$. At this point Lemmata 26 and 28 can be applied to $\mathcal{U}$ in the reverse direction, until we reach $\mathcal{U}(\sigma \to \tau, \alpha \to \alpha)$ (with α fresh), which in turn is equivalent to $\mathcal{U}(\sigma, \tau)$. This completes the proof.

6.2 Unification $\Rightarrow$ GoI-unification

In order to show the converse, just observe that Lemmata 25, 26, 28 hold also for Martelli-Montanari's unification algorithm $\mathcal{U}$, and apply the steps above, exchanging the role of $\mathcal{U}_{GoI}$ with $\mathcal{U}$, starting from $\mathcal{U}(\sigma, \tau)$.

7 Generalizations

Here we show how to remove all the restrictions on terms that we have considered up to now. This is achieved by providing suitable embeddings of terms including constants or/and hapax variables into the set of terms T_Σ^-, and by showing how the case of terms with common variables can be reduced to the case of terms which do not share variables.

7.1 Constants and Hapax Variables

Terms with constants can be easily embedded in the set of terms without constants and hapax variables, in such a way that the unification problem does not change. Namely, for any 0-ary constructor f_0^i, we introduce a fresh binary constructor $f_2^{i_k}$ and a fresh variable α_i, and we substitute f_0^i by $f_2^{i_k}(\alpha_i, \alpha_i)$.

Once constants have been eliminated via the simple encoding above, we can address the issue of variables which occur only once, *i.e.* hapax variables. In order to extend Definition 16 to include terms with hapax variables, we can simply give the following definition.

▶ **Definition 33.** *Let σ, τ be terms, possibly including hapax variables. Then σ and τ GoI-unify if there exists a substitution U such that $U(\sigma)$ and $U(\tau)$ do not contain hapax variables and are GoI-unifiable.*

The above definition, however, is highly non-effective. To achieve effectiveness we can go in the opposite direction and embed the two terms in larger terms with no hapax variables as follows:

▶ **Definition 34** (GoI-unification with hapax variables). *Let $\sigma[\alpha_1, \dots, \alpha_n], \tau[\beta_1, \dots, \beta_m] \in T_\Sigma$ be terms where $\alpha_1, \dots, \alpha_n$ and $\beta_1, \dots, \beta_m$ are the hapax variables in σ and τ respectively. Pick a fresh $1 + 2n + 2m$ constructor, say $f^k_{1+2n+2m}$, then σ and τ GoI-unify if the terms*

$$f^k_{1+2n+2m}(\sigma[\alpha_1, \dots, \alpha_n], \alpha_1, \dots, \alpha_n, \alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_m, \beta_1, \dots, \beta_m)$$

and

$$f^k_{1+2n+2m}(\tau[\beta_1, \dots, \beta_m], \alpha_1, \dots, \alpha_n, \alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_m, \beta_1, \dots, \beta_m)$$

GoI-unify.

The intuition behind the above definition is immediate. We have to duplicate variables in both terms because the set of hapax variables are disjoint and possibly of different cardinality. Of course we can give more involved definitions which reduce the number of extra occurrence pairs to consider, but we do not pursue this issue further. Using Definition 34 we can give the appropriate extensions of Theorem 17, and its consequences, also for terms with hapax variables.

7.2 Unifying Terms with Common Variables

Up to here we have assumed that terms to unify do not share common variables. The reason for this is that, given a term σ, the binary relation $\mathcal{R}(\sigma)$ does not keep track of the name of the variable, which basically plays the role of the place holder for the tail-end of the path. For instance, applying directly Definition 16 to $\mathcal{R}((\alpha \multimap \alpha) \multimap \alpha \multimap \alpha)$ and $\mathcal{R}(\alpha \multimap \alpha)$ would yield success. But clearly the two terms $(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha$ and $\alpha \multimap \alpha$ do not unify, if the name of the variable in the two terms has to be taken seriously. Now, we show how we can remove the assumption that the terms to unify do not share variables. We have essentially already solved the problem in Section 6. Namely, if we want to GoI-unify σ and τ, which have variables in common, then we can GoI-unify the new terms $f_2^k(\sigma, \tau)$ and $f_2^k(\alpha, \alpha)$, where α is fresh, and f_2^k is a new binary constructor. Hence we give the following definition:

▶ **Definition 35.** *Let $\sigma, \tau \in T_\Sigma$, let α be a fresh variable, and let f_2^k be a new binary constructor. Then σ and τ GoI-unify if and only if $f_2^k(\sigma, \tau)$ and $f_2^k(\alpha, \alpha)$ GoI-unify.*

8 Conclusion and Final Remarks

In this paper we provide an alternative criterion for checking that two terms unify in terms of Girard's GoI. It can indeed be seen as the GoI model of unification. The new definition amounts to carrying out successful GoI-execution sequences over the terms under consideration. It is inspired by [19], where an explanation, in terms of resolution of principal types, is given of the notion of linear application between purely linear λ-terms, in the line of the game model in [1]. In a sense, this paper elaborates further on the close relation between the two main logic-derived paradigms of computation, namely *cut-elimination* and *resolution*.

The following remarks are in order.

Recovering substitutions from occ-substitutions. Considering all possible GoI-execution sequences, we can derive *unifiers* thereby providing an analogue of Proposition 11. Namely we conjecture the following:

▶ **Conjecture 36.** *Let $\sigma, \tau \in T_\Sigma$ be terms, and suppose $\mathcal{U}_{GoI}(\sigma, \tau)$. Let $\Gamma^{\langle u[\alpha], v[\alpha]\rangle}$ denote a successful GoI-execution sequence for a pair of variable occurrences $\langle u[\alpha], v[\alpha]\rangle$, and let $M^{\Gamma^{\langle u[\alpha],v[\alpha]\rangle}}_{\langle u[\alpha],v[\alpha]\rangle}$ be the overall occ-substitution generated along $\Gamma^{\langle u[\alpha],v[\alpha]\rangle}$, simply denoted by $M^{\Gamma^{\langle u[\alpha],v[\alpha]\rangle}}$ in the sequel. Define the substitution*

$$U_{GoI}(\alpha) = \begin{cases} \mathcal{T}(\{M^{\Gamma^{\langle u[\alpha],v[\alpha]\rangle}}(\alpha) \mid \langle u[\alpha], v[\alpha]\rangle \in \mathcal{R}(\sigma)\}) & \text{if } \alpha \in \sigma \\ \mathcal{T}(\{M^{\Gamma^{\langle u[\alpha],v[\alpha]\rangle}}(\alpha) \mid \langle u[\alpha], v[\alpha]\rangle \in \mathcal{R}(\tau)\}) & \text{if } \alpha \in \tau \ , \end{cases}$$

then $U_{GoI}(\sigma) = U_{GoI}(\tau)$.

Moreover, we further conjecture that, if throughout the GoI-execution sequences of Lemma 18 the most general occ-substitutions are always picked, then U_{GoI} is the m.g.u. of σ and τ.

We only provide the intuition behind the above statement. If σ and τ GoI-unify, then for all occurrence pairs $\langle u[\alpha], v[\alpha]\rangle \in \mathcal{R}(\sigma)$ and GoI sequences $\Gamma^{\langle u[\alpha],v[\alpha]\rangle}$ we have, by Lemma 18, that there are occurrences in a substitution instance of τ which match the occurrence $M^{\Gamma^{\langle u[\alpha],v[\alpha]\rangle}}(u[\alpha])$. For the definition in Proposition 36 to be well-defined we need to know that the set $\{M^{\Gamma^{\langle u[\alpha],v[\alpha]\rangle}}(\alpha) \mid \langle u[\alpha], v[\alpha]\rangle \in \mathcal{R}(\sigma)\}$ consists of compatible occurrences, so that Proposition 6 can apply. This is not immediate, but is should ultimately follow from the fact that the substitutions arise from pairs in $\mathcal{R}(\tau)$ which in turn are compatible, using a maximal execution path which passes through all the occurrence pairs of the variables involved. Finally, the fact that one has chosen all possible execution sequences $\Gamma^{\langle u[\alpha],v[\alpha]\rangle}$ for each occurrence pair ensures that, in applying Proposition 6, no extra new variable in Z is necessary.

Comparison between *unification* and *GoI-unification*. *GoI-unification* is more a criterion rather than an algorithm, although it permits to synthesize the unifier, when it exists, as shown above. Usual unification requires a repeated deep substitution of terms for variables, while *GoI unification* only requires pattern matching along GoI execution sequences. Moreover not all possible GoI execution sequences need to be computed to check that unification achieves. If we were to use Definition 16 for checking unification rather than ordinary unification, termination would be more difficult to deal with. Termination is easily proved for standard *unification* by induction on syntax, while termination of *GoI-unification* would require to inspect all the possible finitely many paths along the occurrence terms, or a maximising path which uses all possible occurrence pairs for the variables involved.

Resolution. In [18], only GoI linear application, between purely linear binary types, was discussed in terms of resolution on terms formed using a single binary constructor, namely $\multimap$. Clearly the definitions in this paper can be easily extended to encompass resolution in full generality.

Duplication. We can extend this paper by dealing also with other meta-operations on terms besides substitution, namely *replication* of subterms as arise in Linear Logic. This would amount to generalizing the language of occurrences in the line of [9, 18, 11]. Giving a top-down account of replication is rather complex, in that a unification-duplication algorithm needs to be defined (see [11]). On the other hand, a bottom-up approach comes naturally following the literature on GoI, as Girard has shown, see *e.g.* [13, 1].

Invariants. In [1] only resolutions on copy-cat strategies, *i.e.* partial involutions, were considered, which in our setting amount to terms where each variable occurs exactly twice. On the other hand, in this paper we open up the possibility of using arbitrary terms, *i.e.* non-deterministic strategies. It would be worthwhile to study the λ-models which arise using resolution on such strategies to model β-reduction or, equivalently, the *invariants* of λ-terms that such strategies can express.

References

1 Samson Abramsky. A structural approach to reversible computation. *Theoretical Computer Science*, 347(3):441–464, 2005. doi:10.1016/j.tcs.2005.07.002.

2 Samson Abramsky, Esfandiar Haghverdi, and Philip Scott. Geometry of Interaction and linear combinatory algebras. *Mathematical Structures in Computer Science*, 12(5):625–665, 2002. doi:10.1017/S0960129502003730.

3 Samson Abramsky and Marina Lenisa. Linear realizability and full completeness for typed lambda-calculi. *Annals of Pure and Applied Logic*, 134(2):122–168, 2005. doi:10.1016/j.apal.2004.08.003.

4 Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. The Machinery of Interaction. In *Proceedings of the 22nd International Symposium on Principles and Practice of Declarative Programming*, PPDP '20, New York, NY, USA, 2020. Association for Computing Machinery. doi:10.1145/3414080.3414108.

5 Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. Multi types and reasonable space. *Proc. ACM Program. Lang.*, 6(ICFP), August 2022. doi:10.1145/3547650.

6 Beniamino Accattoli, Ugo Dal Lago, and Gabriele Vanoni. Reasonable Space for the λ-Calculus, Logarithmically. In *Proceedings of the 37th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '22, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3531130.3533362.

7 Marc Bagnol. *On the Resolution Semiring.* Theses, Aix-Marseille Universite, December 2014. URL: https://theses.hal.science/tel-01215334.

8 Patrick Baillot and Marco Pedicini. Elementary complexity and geometry of interaction. *Fundamenta Informaticae*, 45(1-2):1–31, 2001. URL: http://content.iospress.com/articles/fundamenta-informaticae/fi45-1-2-02.

9 Alberto Ciaffaglione, Pietro Di Gianantonio, Furio Honsell, Marina Lenisa, Ivan Scagnetto, et al. $\lambda!$-calculus, Intersection Types, and Involutions. In *Leibniz International Proceedings in Informatics, LIPIcs*, volume 131. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, 2019.

10 Alberto Ciaffaglione, Furio Honsell, Marina Lenisa, Ivan Scagnetto, et al. The involutions-as-principal types/application-as-unification analogy. *EPiC Series in Computing*, 57:254–270, 2018. doi:10.29007/NTWG.

11 Pietro Di Gianantonio and Marina Lenisa. Principal Types as Lambda Nets. In Henning Basold, Jesper Cockx, and Silvia Ghilezan, editors, *27th International Conference on Types for Proofs and Programs (TYPES 2021)*, volume 239 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 5:1–5:23, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.TYPES.2021.5.

12 Boris Eng. *An exegesis of transcendental syntax*. Theses, Université Sorbonne Paris Nord, June 2023. URL: https://hal.science/tel-04179276.

13 Jean-Yves Girard. Geometry of Interaction 1: Interpretation of System F. In R. Ferro, C. Bonotto, S. Valentini, and A. Zanardo, editors, *Logic Colloquium '88*, volume 127 of *Studies in Logic and the Foundations of Mathematics*, pages 221–260. Elsevier, 1989. doi:10.1016/S0049-237X(08)70271-4.

14 Jean-Yves Girard. Geometry of interaction 2: Deadlock-free algorithms. In Per Martin-Löf and Grigori Mints, editors, *COLOG-88*, pages 76–93, Berlin, Heidelberg, 1990. Springer Berlin Heidelberg.

15 Jean-Yves Girard. Geometry of Interaction III: accommodating the additives. *London Mathematical Society Lecture Note Series*, pages 329–389, 1995.

16 Jean-Yves Girard. Transcendental syntax I: deterministic case. *Mathematical Structures in Computer Science*, 27(5):827–849, 2017. doi:10.1017/S0960129515000407.

17 Furio Honsell. Talk delivered at IFIP W.G. 2.2 Bologna Meeting, 2023.

18 Furio Honsell, Marina Lenisa, and Ivan Scagnetto. Λ-Symsym: An Interactive Tool for Playing with Involutions and Types. In Ugo de'Liguoro, Stefano Berardi, and Thorsten Altenkirch, editors, *26th International Conference on Types for Proofs and Programs (TYPES 2020)*, volume 188 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 7:1–7:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.TYPES.2020.7.

19 Furio Honsell, Marina Lenisa, and Ivan Scagnetto. Principal Types as Partial Involutions, 2024. doi:10.48550/arXiv.2402.07230.

20 Alberto Martelli and Ugo Montanari. An Efficient Unification Algorithm. *ACM Trans. Program. Lang. Syst.*, 4(2):258–282, April 1982. doi:10.1145/357162.357169.