Mechanized Undecidability of Higher-order beta-Matching

Andrej Dudenhefner
TU Dortmund University, Germany

Abstract

Higher-order β-matching is the following decision problem: given two simply typed λ-terms, can the first term be instantiated to be β-equivalent to the second term? This problem was formulated by Huet in the 1970s and shown undecidable by Loader in 2003 by reduction from λ-definability.

The present work presents a novel undecidability proof for higher-order β-matching, in an effort to verify this result by means of a proof assistant in full detail. Rather than starting from λ-definability, the presented proof encodes a restricted form of string rewriting as higher-order β-matching. The particular approach is similar to Urzyczyn's undecidability result for intersection type inhabitation.

The presented approach has several advantages. First, the proof is simpler to verify in full detail due to the simple form of rewriting systems, which serve as a starting point. Second, undecidability of the considered problem in string rewriting is already certified using the Coq proof assistant. As a consequence, we obtain a certified many-one reduction from the Halting Problem to higher-order β-matching. Third, the presented approach identifies a uniform construction which shows undecidability of higher-order β-matching, λ-definability, and intersection type inhabitation.

The presented undecidability proof is mechanized in the Coq proof assistant and contributed to the existing Coq Library of Undecidability Proofs.

2012 ACM Subject Classification Theory of computation $\rightarrow$ Lambda calculus

Keywords and phrases lambda-calculus, simple types, undecidability, higher-order matching, mechanization, Coq

Digital Object Identifier 10.4230/LIPIcs.CVIT.2016.23

Supplementary Material *Software (Source Code)*: `github/uds-psl/coq-library-undecidability`

1 Introduction

Higher-order β-unification in the simply typed λ-calculus is the following decision problem: given two simply typed λ-terms M, N, is there a substitution S such that the instance $S(M)$ is β-equivalent to the instance $S(N)$? Undecidability of higher-order β-unification was established by Huet [9] in the 1970s, raising the question whether the β-matching problem [10] (the right-hand side term N does not contain free variables) is decidable[1]. An equivalent presentation of higher-order β-matching (cf. Statman's range question [17]) is: given a term F which can be assigned the simple type $\sigma \to \tau$ and a term N which can be assigned the simple type τ, is there a term M which can be assigned the simple type σ such that $F\,M$ is β-equivalent to N?

Decidability of higher-order β-matching was answered negatively[2] by Loader [13] by reduction from a variant of λ-definability. Loader's proof introduces intricate machinery to formulate β-matching constraints which specify arbitrary finite functions. A later approach by Joly [11] refines Loader's result, shifting technical challenges to undecidability of the underlying λ-definability problem. Both approaches render verification of the negative result in full detail (for example, by means of a mechanization in a proof assistant) quite challenging.

[1] Dowek [5] gives a comprehensive overview over unification and matching problems for the λ-calculus.
[2] Not to be confused with the positive answer by Stirling [18] for higher-order $\beta\eta$-matching.

42nd Conference on Very Important Topics (CVIT 2016).
Editors: John Q. Open and Joan R. Access; Article No. 23; pp. 23:1–23:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The present work presents a novel proof of the undecidability of higher-order β-matching, which is mechanized using the Coq proof assistant [19]. The mechanization leaves no room for ambiguities and potential errors, complementing existing work on mechanized undecidability of higher-order β-unification [16].

The presented proof is not based on λ-definability; rather, we consider a known rewriting problem in a restricted class of semi-Thue systems [20, Lemma 2] as a starting point. The specific rewriting problem, referred to as $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$, is: given a collection of rewrite rules of shape $ab \Rightarrow cd$, where a, b, c, d are alphabet symbols, is there a non-empty sequence of $\mathbf{0}$s which can be transformed into a sequence of $\mathbf{1}$s? As a consequence of the different starting point, the presented proof is simpler to verify in full detail and yields a concise mechanization. The mechanization is incorporated into the existing Coq Library of Undecidability Proofs [8], alongside the existing mechanization[3] of undecidability of the problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$.

The main inspiration for the novel approach in the present work is Urzyczyn's undecidability proof for intersection type inhabitation [20]. Considering the relationship between the intersection type discipline and finite model theory [15], the approach in the present work has an additional benefit: it is uniformly applicable to prove undecidability of higher-order β-matching, intersection type inhabitation, and λ-definability.

Paper organization The present work is structured as follows:

- **Section 2:** Preliminaries for the simply typed λ-calculus, higher-order β-matching, and simple semi-Thue systems (including the undecidable Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$).
- **Section 3:** Reduction from Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ to higher-order β-matching.
- **Section 4:** Overview over the mechanization in the Coq proof assistant.
- **Section 5:** Applicability to intersection type inhabitation and λ-definability.
- **Section 6:** Concluding remarks.

Statements and proofs in the digital version of the present work are linked to the corresponding mechanization, which is marked by the symbol [🐓].

2 Preliminaries

In the present section we fix preliminaries and basic notation, following standard literature [2].

Higher-order β-Matching in the Simply Typed λ-Calculus

The syntax of untyped λ-terms is given in the following Definition 1.

▶ **Definition 1** (λ-Terms [🐓]).

$$M, N ::= x \mid M\, N \mid \lambda x.M \quad \textit{where } a, \ldots, z \textit{ range over term variables}$$

Substitution of the term variable x in the term M by the term N is denoted $M[x := N]$. As usual, term application associates to the left, and we may group consecutive λ-abstractions. We commonly refer to the term $\lambda x.x$ as I.

▶ **Definition 2** (β-Reduction [🐓]). *The relation $\rightarrow_\beta$ on terms is the contextual closure of $(\lambda x.M)\, N \rightarrow_\beta M[x := N]$.*

The β-equivalence relation $=_\beta$ is the reflexive, transitive, symmetric closure of $\rightarrow_\beta$.

[3] The (Turing machine) Halting Problem is easily presented as Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ [6, Lemma 3.3].

In the simply typed λ-calculus we may assign to a term M a simple type τ in type environment Γ, written $\Gamma \vdash M : \tau$. Similarly to prior work [13], one ground atom ι in the construction of simple types suffices for the negative result in the present work. Definition 5 contains the rules (Var), ($\to$I), and ($\to$E) of the simple type system.

▶ **Definition 3** (Simple Types with Ground Atom ι []).

$$\sigma, \tau ::= \iota \mid \sigma \to \tau$$

The arrow type constructor $\to$ associates to the right.

▶ **Definition 4** (Type Environments).

$$\Gamma ::= \{x_1 : \sigma_1, \dots, x_n : \sigma_n\}$$

▶ **Definition 5** (Simple Type System []).

$$\frac{(x : \sigma) \in \Gamma}{\Gamma \vdash x : \sigma}\,(\text{Var}) \qquad \frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x.M : \sigma \to \tau}\,(\to\text{I}) \qquad \frac{\Gamma \vdash M : \sigma \to \tau \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\,N : \tau}\,(\to\text{E})$$

The following Example 6 illustrates a type derivation in the simple type system.

▶ **Example 6.**

$$\cfrac{\cfrac{\cfrac{\cfrac{(f : \iota \to \iota) \in \{u : \iota, f : \iota \to \iota\}}{\{u : \iota, f : \iota \to \iota\} \vdash f : \iota \to \iota}\,(\text{Var}) \qquad \cfrac{(u : \iota) \in \{u : \iota, f : \iota \to \iota\}}{\{u : \iota, f : \iota \to \iota\} \vdash u : \iota}\,(\text{Var})}{\{u : \iota, f : \iota \to \iota\} \vdash f\,u : \iota}\,(\to\text{E})}{\{u : \iota\} \vdash \lambda f.f\,u : (\iota \to \iota) \to \iota}\,(\to\text{I})}{\emptyset \vdash \lambda u.\lambda f.f\,u : \iota \to (\iota \to \iota) \to \iota}\,(\to\text{I})$$

Higher-order β-matching is the following typed unification problem, for which only one side is subject to instantiation.

▶ **Problem 7** (Higher-order β-Matching ($F\,\mathsf{X} = N$) []). Given terms F, N and simple types σ, τ such that $\emptyset \vdash F : \sigma \to \tau$ and $\emptyset \vdash N : \tau$, is there a term M such that $\emptyset \vdash M : \sigma$ and $F\,M =_\beta N$?

Undecidability of higher-order β-matching is shown by Loader [13] using a reduction from a variant of λ-definability.

▶ **Theorem 8** ([13, Theorem 5.5]). *Higher-order β-matching is undecidable.*

For the remainder of the present work we use the term *matching* in order to refer to higher-order β-matching. Since simply typed terms are strongly normalizing and β-reduction is confluent [2], it suffices to consider terms F, M, N in normal form.

Let us get familiar with matching by means of several illustrating examples. The following Example 9 illustrates a positive matching instance.

▶ **Example 9.** Consider the terms $F := \lambda x.\lambda y.x\,y\,I$ and $N := I$, for which we have $\emptyset \vdash F : (\iota \to (\iota \to \iota) \to \iota) \to (\iota \to \iota)$ and $\emptyset \vdash N : \iota \to \iota$.

The matching instance $F\,\mathsf{X} = N$ is solvable, including the solution $M := \lambda u.\lambda f.f\,u$. To be precise, we have the following two properties:

- $\emptyset \vdash M : \iota \to (\iota \to \iota) \to \iota$ (Example 6)
- $F\,M =_\beta \lambda y.(\lambda u.\lambda f.f\,u)\,y\,I =_\beta \lambda y.I\,y =_\beta N$

We want to encode certain functional behavior as a matching instance. The following Example 10 shows a naive approach to such an encoding and its limitations.

▶ **Example 10.** Let us associate elements of the set $\{1, 2, 3\}$ with projections $\pi_1 := \lambda xyz.x$, $\pi_2 := \lambda xyz.y$, and $\pi_3 := \lambda xyz.z$ respectively. For the term $G := \lambda h.\lambda xyz.h\,y\,z\,x$ we have $G\,\pi_1 =_\beta \pi_2$, $G\,\pi_2 =_\beta \pi_3$, and $G\,\pi_3 =_\beta \pi_1$. Therefore, G realizes a finite function $f_G : \{1, 2, 3\} \to \{1, 2, 3\}$ such that $f_G(1) = 2$, $f_G(2) = 3$, and $f_G(3) = 1$.

Let $\kappa := \iota \to \iota \to \iota \to \iota$ be a simple type for which we have $\emptyset \vdash \pi_i : \kappa$ for $i \in \{1, 2, 3\}$. Consider the matching instance $F\,\mathsf{X} = \pi_2$, where $F := \lambda t.t\,G\,\pi_1$, and for which we have $\emptyset \vdash F : ((\kappa \to \kappa) \to \kappa \to \kappa) \to \kappa$. The intended "meaning" of this matching instance is: starting with the element 1, repeatedly apply the function f_G in order to construct the element 3.

One solution for this instance is the term $\lambda f.\lambda s.f\,(f\,s)$ for which we have:

$$F\,(\lambda f.\lambda s.f\,(f\,s)) =_\beta f\,(f\,s)[f := G, s := \pi_1] =_\beta G\,\pi_2 =_\beta N$$

This solution follows the intended meaning of the underlying representation, constructing the element $f_G(f_G(1)) = 3$. Another solution is $\lambda f.\lambda s.f\,(f\,(f\,(f\,(f\,s))))$, which utilizes $f_G(f_G(f_G(f_G(f_G(1))))) = 3$.

Unfortunately, there are solutions to the above matching instance which behave differently. One such solution is $\lambda f.\lambda s.\pi_3$, for which we also have $F\,(\lambda f.\lambda s.\pi_3) =_\beta \pi_3$. In this case, the element 3 is constructed "ad-hoc", with no reference to the provided arguments. Another solution is the term $\lambda f.\lambda s.\lambda xyz.f\,s\,z\,z\,z$. This solution exploits the exact representation of elements via projections, disregarding any intended meaning of the underlying representation.

In the above Example 10 we would like to exclude certain ad-hoc solutions, in order to faithfully encode intended functional behavior. Towards this aim, a known technique how to restrict the shape of solutions is illustrated in the following Example 11.

▶ **Example 11** ([5, Proposition 3.4])**.** Consider a term M in normal form such that $M\,I\,u =_\beta u$ where u is a term variable, and $\emptyset \vdash M : (\kappa \to \kappa) \to \kappa \to \kappa$. By case analysis on M we have that $M = \lambda f.\lambda s.N$ for some term N in normal form. Furthermore, $\{f : \kappa \to \kappa, s : \kappa\} \vdash N : \kappa$ and $N[f := \lambda x.x, s := u] =_\beta u$. Therefore, the term N is not an abstraction. By induction on the size of N and case analysis of the normal form we have that $N = s$ or $N = f\,(\ldots(f\,s)\ldots)$. Since the term M contains exactly two λ-abstractions, it cannot be an ad-hoc solution from Example 10.

Combining Example 10 with Example 11, we formulate in the following Example 12 a matching instance which faithfully encodes the desired functional behavior.

▶ **Example 12.** Let $F := \lambda t.\lambda r.r\,(t\,G\,\pi_1)\,(\lambda u.t\,I\,u)$ and $N := \lambda r.r\,\pi_3\,(\lambda u.u)$ where the term $G = \lambda h.\lambda xyz.h\,y\,z\,x$ is from Example 10. We have

- $\emptyset \vdash F : ((\kappa \to \kappa) \to \kappa \to \kappa) \to (\kappa \to (\kappa \to \kappa) \to \iota) \to \iota$
- $\emptyset \vdash N : (\kappa \to (\kappa \to \kappa) \to \iota) \to \iota$

The matching instance $F\,\mathsf{X} = N$ combines the matching instance from Example 10 with the additional restriction from Example 11. Therefore, solutions such as $\lambda f.\lambda s.f\,(f\,s)$ and $\lambda f.\lambda s.f\,(f\,(f\,(f\,(f\,s))))$ from Example 10 which follow the intended "meaning" of the underlying representation still solve $F\,\mathsf{X} = N$. However, ad-hoc solutions such as $\lambda f.\lambda s.\pi_3$ or $\lambda f.\lambda s.\lambda xyz.f\,s\,z\,z\,z$ from Example 10 do not solve $F\,\mathsf{X} = N$ because such solutions contain too many λ-abstractions.

▶ Remark 13. Without the restriction $\emptyset \vdash M : (\kappa \to \kappa) \to \kappa \to \kappa$ in Example 11 the term $M := \lambda f.\lambda s.f\,f\,s$ satisfies $M\,I\,u =_\beta u$. However, Example 12 does not admit M as a solution. The present work relies on well-typedness, but might be adapted to an untyped scenario.

The following Remark 14 illustrates how the addition of η-reduction would make the technique from Example 12 (as well as Loader's approach) unsuitable.

▶ Remark 14. Example 11 demonstrates how to restrict the number of abstractions in solutions. However, in the presence of η-reduction (contextual closure of $\lambda x.f\,x \to_\eta f$) this does not work, as shown below. Consider the terms $G := \lambda h.\lambda xyz.h\,y\,z\,x$, $\pi_1 = \lambda xyz.x$, $\pi_2 = \lambda xyz.y$, and $\pi_3 = \lambda xyz.z$ from Example 11. The term $M := \lambda g.\lambda h.\lambda xyz.h\,(g\,\pi_1 x\,z\,y)\,y\,z$ solves the matching instance in Example 12 because:

$$M\,G\,\pi_1 =_\beta \big(\lambda g.\lambda h.\lambda xyz.h\,(g\,\pi_1 x\,z\,y)\,y\,z\big)\,G\,\pi_1 =_\beta \lambda xyz.\pi_1\,(G\,\pi_1 x\,z\,y)\,y\,z$$
$$=_\beta \lambda xyz.G\,\pi_1 x\,z\,y \qquad =_\beta \lambda xyz.\pi_1\,z\,y\,x =_\beta \lambda xyz.z = \pi_3$$
$$M\,I\,u =_\beta \big(\lambda g.\lambda h.\lambda xyz.h\,(g\,\pi_1 x\,z\,y)\,y\,z\big)\,I\,u =_\beta \lambda xyz.u\,(I\,\pi_1 x\,z\,y)\,y\,z$$
$$=_\beta \lambda xyz.u\,(\pi_1 x\,z\,y)\,y\,z \qquad =_\beta \lambda xyz.u\,x\,y\,z \to_\eta^* u$$

In particular, η-reduction allows for additional λ-abstractions in the solution, making the technique from Example 11 unsuitable.

The observation from Example 12 is generalized by Loader to encode arbitrary families of finite functions. This results in undecidability of higher-order β-matching by reduction from a variant of λ-definability. Loader's generalization is quite sophisticated, as it requires construction principles to restrict shapes of realizers of arbitrary higher-order finite functions. In the present work, we focus on a fragment which can be identified by inspection of intersection types occurring in the undecidability proof of intersection type inhabitation [20, 6] and their relationship to finite model theory [15]. This leads to a simpler undecidability proof and reveals a connection between matching, intersection type inhabitation, and λ-definability. The presented approach has similarities with Joly's [11] refinement of Loader's proof. Joly shifts the technical burden to a particular λ-definability problem, which then is simpler to handle. Instead, we avoid λ-definability altogether and use a rewriting problem in a restricted class of *simple* semi-Thue systems as a starting point.

Simple Semi-Thue Systems

A simple semi-Thue system (Definition 15) is a rewriting system of restricted shape, introduced by Urzyczyn [20] in order to show undecidability of intersection type inhabitation.

▶ **Definition 15** (Simple Semi-Thue System []). *A semi-Thue system $\mathfrak{R}$ over an alphabet $\mathcal{A}$ is* simple*, if each rule has the shape $ab \Rightarrow cd$ for $a, b, c, d \in \mathcal{A}$.*

The reflexive, transitive closure of the rewriting relation for a given simple semi-Thue system $\mathfrak{R}$ is denoted $\Rightarrow_{\mathfrak{R}}^*$. For arbitrary simple semi-Thue systems it is undecidable whether some non-empty sequence of $\mathbf{0}$s can be transformed into a sequence of $\mathbf{1}$s.

▶ **Problem 16** ($\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ []). Given a simple semi-Thue system $\mathfrak{R}$, does $\mathbf{0}^n \Rightarrow_{\mathfrak{R}}^* \mathbf{1}^n$ hold for some $n > 0$?

▶ **Theorem 17** ([6, Lemma 3.3 []]). *Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ is undecidable.*

The following Example 18 illustrates a positive instance of Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$.

▶ **Example 18.** Let $\mathfrak{R} := \{\mathbf{00} \Rightarrow \mathbf{22}, \mathbf{02} \Rightarrow \mathbf{11}, \mathbf{20} \Rightarrow \mathbf{11}\}$ be a simple semi-Thue system over the alphabet $\{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$. We have $\mathbf{0000} \Rightarrow_{\mathfrak{R}} \mathbf{0220} \Rightarrow_{\mathfrak{R}} \mathbf{1120} \Rightarrow_{\mathfrak{R}} \mathbf{1111}$. As a side note, we have $\mathbf{0}^n \not\Rightarrow_{\mathfrak{R}}^* \mathbf{1}^n$ for $n \in \{1, 2, 3\}$.

▶ Remark 19. Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ is used as a starting point in a refinement [6, Lemma 4.4] of Urzyczyn's undecidability result for intersection type inhabitation [20]. Undecidability of Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ is mechanized as part of Coq Library of Undecidability Proofs [8], making it a good starting point for further mechanized undecidability results.

3 Undecidability of Higher-order β-Matching

In this section we develop our main result (Theorem 39): a reduction from the rewriting problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ to higher-order β-matching.

For the remainder of the section we fix the simple semi-Thue system $\mathfrak{R} := \{R_1, \dots, R_L\}$ with $L > 0$ rules over the finite alphabet $\{\mathbf{0}, \mathbf{1}, \mathbf{2}, \dots, \mathbf{K}\}$. Our approach is to construct simply typed terms which capture the two main aspects of the rewriting problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$: the search for a sufficiently long starting sequence of $\mathbf{0}$s, and the individual rewriting steps to the desired sequence of $\mathbf{1}$s.

The remainder of the present section is structured as follows. First, we fix basic notation, encoding, and properties of the rewriting in the system $\mathfrak{R}$. Second, we restrict the shape of potential solutions for the constructed matching instance, similarly to Example 11. Third, for solutions of restricted shape we capture the functional properties of the rewriting problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$.

Notation

We introduce four additional symbols in extended alphabet $\mathcal{A} := \{\mathbf{0}, \mathbf{1}, \dots, \mathbf{K}\} \cup \{\$, \bullet, \top, \bot\}$. We represent an alphabet symbol $i \in \mathcal{A}$ as the projection $\pi_i := \lambda s_{\mathbf{0}} s_{\mathbf{1}} \dots s_{\mathbf{K}} s_{\$} s_{\bullet} s_{\top} s_{\bot}.s_i$ typed by the simple type $\kappa := \underbrace{\iota \to \dots \to \iota}_{|\mathcal{A}| \text{ times}} \to \iota$. For readability, we use the following **case** notation to match individual symbols.

▶ **Definition 20 (case).** *For $k \in \mathbb{N}$, distinct $i_1, \dots, i_k \in \mathcal{A}$, and terms $M_1, \dots, M_k$:*

$$\textbf{case } x \textbf{ of } \langle M \mid i_1 \mapsto M_1 \mid \dots \mid i_k \mapsto M_k \rangle := x\, N_{\mathbf{0}}\, N_{\mathbf{1}} \dots N_{\mathbf{K}}\, N_{\$}\, N_{\bullet}\, N_{\top}\, N_{\bot}$$

$$\textit{where } N_i = \begin{cases} M_j & \textit{if } i = i_j \\ M & \textit{otherwise} \end{cases}$$

A particular term δ_i for $i \in \mathcal{A}$, which we use commonly is:

$$\delta_i := \lambda x.\lambda s_{\mathbf{0}} s_{\mathbf{1}} \dots s_{\mathbf{K}} s_{\$} s_{\bullet} s_{\top} s_{\bot}.\textbf{case } x \textbf{ of } \langle s_\bot \mid \top \mapsto s_i \rangle$$

We have $\emptyset \vdash \delta_i : \kappa \to \kappa$, and the following Lemma 21 specifies the behavior of δ_i.

▶ **Lemma 21.** *For $i, j \in \mathcal{A}$ such that $i \neq j$ we have $\delta_i\, \pi_\top =_\beta \pi_i$ and $\delta_i\, \pi_j =_\beta \pi_\bot$.*

Syntactic Constrains

We identify the shape of "well-formed" terms, suitable to represent rewriting. In the following Definition 22 terms in the set $\mathcal{Q}_m$ capture consecutive rule application for a word of length $m + 1$, ending in the word $\mathbf{1}^{m+1}$ (represented by $z_{\mathbf{1}} \in \mathcal{Q}_m$). The subterm $r_i\, p_j$ (and $r_i\, (\lambda w.p_j\, w)$) for $i \in \{1, \dots, L\}$ and $j \in \{1, \dots, m\}$ indicates an application of the rule R_i at position j. Additionally, terms in the set $\mathcal{R}_m$ capture consecutive increase of word length starting with $m+1$, and initialization with $\mathbf{0}$s before rewriting (represented by $z_{\mathbf{0}}\, N\, M \in \mathcal{R}_m$ for $M \in \mathcal{Q}_m$). Specifically, the subterm $z_\star\, N\, (\lambda p_{m+1}.M)$ introduces an additional bound variable p_{m+1} in order the argue about rule application at position $m + 1$ in the longer word. Consequently, terms in $\mathcal{R}_1$ represent witnesses for an arbitrary expansion of a word starting with length 2, followed by initialization with $\mathbf{0}$s, and consecutive rule application (potentially ending in $\mathbf{1}$s).

▶ **Definition 22** (Sets $\mathcal{Q}_m$ [], $\mathcal{R}_m$ [] of Terms)**.** *For $m > 0$, let $\mathcal{Q}_m$ and $\mathcal{R}_m$ be the smallest sets of terms satisfying the following rules:*

- $z_{\mathbf{1}} \in \mathcal{Q}_m$
- *if* $M \in \mathcal{Q}_m$ *then* $(r_i\, p_j\, M) \in \mathcal{Q}_m$ *for* $i \in \{1, \ldots, L\}$ *and* $j \in \{1, \ldots, m\}$
- *if* $M \in \mathcal{Q}_m$ *then* $(r_i\, (\lambda w.p_j\, w)\, M) \in \mathcal{Q}_m$ *for* $i \in \{1, \ldots, L\}$ *and* $j \in \{1, \ldots, m\}$
- *if* $M \in \mathcal{Q}_m$ *then* $(z_{\mathbf{0}}\, N\, M) \in \mathcal{R}_m$
- *if* $M \in \mathcal{R}_{m+1}$ *then* $(z_\star\, N\, (\lambda p_{m+1}.M)) \in \mathcal{R}_m$

▶ Remark 23. Terms in $\mathcal{R}_1$ are inspired by inhabitants in a refinement [6, Lemma 4.4] of Urzyczyn's undecidability result for intersection type inhabitation [20].

Free variables occurring in terms in $\mathcal{Q}_m$ and $\mathcal{R}_m$ are assigned simple types according to the following type environment Γ_m.

▶ **Definition 24** (Type Environment Γ_m [])**.** *For $m > 0$ let*

$$\begin{aligned}\Gamma_m := \{&z_{\mathbf{1}} : \kappa, z_{\mathbf{0}} : (\kappa \to \kappa) \to \kappa \to \kappa, z_\star : (\kappa \to \kappa) \to ((\kappa \to \kappa) \to \kappa) \to \kappa,\\ &p_1 : \kappa \to \kappa, \ldots, p_m : \kappa \to \kappa,\\ &r_1 : (\kappa \to \kappa) \to \kappa \to \kappa, \ldots, r_L : (\kappa \to \kappa) \to \kappa \to \kappa\}\end{aligned}$$

Similarly to Example 11, we formulate typed terms (Definition 25) and β-equivalence constraints characterizing members of $\mathcal{Q}_m$ (Lemma 27) and $\mathcal{R}_m$ (Lemma 28).

▶ **Definition 25** (Typed Terms $H_\star$ [], $H_{\mathbf{0}}$ [], H_R [])**.**

$$H_\star := \lambda h.\lambda g.\lambda s_{\mathbf{0}} s_{\mathbf{1}} \ldots s_{\mathbf{K}} s_\$ s_\bullet s_\top s_\bot.\textbf{case}\ g\, \delta_\bullet\ \textbf{of}\ \langle s_\bot \mid \$ \mapsto s_\$ \rangle$$

$$H_{\mathbf{0}} := \lambda h.\lambda x.\lambda s_{\mathbf{0}} s_{\mathbf{1}} \ldots s_{\mathbf{K}} s_\$ s_\bullet s_\top s_\bot.\textbf{case}\ x\ \textbf{of}\ \langle s_\bot \mid \mathbf{1} \mapsto s_\$ \rangle$$

$$H_R := \lambda h.\lambda x.\lambda s_{\mathbf{0}} s_{\mathbf{1}} \ldots s_{\mathbf{K}} s_\$ s_\bullet s_\top s_\bot.\textbf{case}\ h\, \pi_\top\ \textbf{of}\ \langle s_\bot \mid \bullet \mapsto \textbf{case}\ x\ \textbf{of}\ \langle s_\bot \mid \mathbf{1} \mapsto s_{\mathbf{1}} \rangle\rangle$$

$$\emptyset \vdash H_\star : \Gamma_m(z_\star) \qquad \emptyset \vdash H_{\mathbf{0}} : \Gamma_m(z_{\mathbf{0}}) \qquad \emptyset \vdash H_R : \Gamma_m(r_i)\ \textit{for}\ i \in \{1, \ldots, L\}$$

We introduce substitutions S_F and S_H acting on the term variables $z_\star, z_{\mathbf{1}}, z_{\mathbf{0}}, r_1, \ldots, r_L$, which occur in terms in $\mathcal{Q}_m$ and $\mathcal{R}_m$.

▶ **Definition 26** (Substitutions S_F [], S_H [])**.**

$$\begin{array}{llll} S_F(z_\star) := \lambda h.\lambda g.g\, I & S_F(z_{\mathbf{1}}) := u & S_F(z_{\mathbf{0}}) := \lambda h.I & S_F(r_j) := I\ \textit{for}\ j \in 1, \ldots L \\ S_H(z_\star) := H_\star & S_H(z_{\mathbf{1}}) := \pi_{\mathbf{1}} & S_H(z_{\mathbf{0}}) := H_{\mathbf{0}} & S_H(r_j) := H_R\ \textit{for}\ j \in 1, \ldots L \end{array}$$

▶ **Lemma 27.** *For $m > 0$, if a term M is in normal form such that $\Gamma_m \vdash M : \kappa$, $S_F(M)[p_1 := I, \ldots, p_m := I] =_\beta u$, and $S_H(M)[p_1 := \delta_\bullet, \ldots, p_m := \delta_\bullet] =_\beta \pi_{\mathbf{1}}$, then $M \in \mathcal{Q}_m$.*

Proof []. Induction on the size of M and case analysis of the normal form. ◀

▶ **Lemma 28.** *For $m > 0$, if a term M is in normal form such that $\Gamma_m \vdash M : \kappa$, $S_F(M)[p_1 := I, \ldots, p_m := I] =_\beta u$, and $S_H(M)[p_1 := \delta_\bullet, \ldots, p_m := \delta_\bullet] =_\beta \pi_\$$, then $M \in \mathcal{R}_m$.*

Proof []. Induction on the size of M, case analysis of the normal form, and Lemma 27. ◀

As a consequence of the above Lemma 28, the following Theorem 29 presents β-equivalence constraints which suffice to restrict the shape of terms under consideration.

▶ **Theorem 29.** *If a term M is in normal form such that $\emptyset \vdash M : \Gamma_1(r_1) \to \cdots \to \Gamma_1(r_L) \to \Gamma_1(z_{\mathbf{0}}) \to \Gamma_1(z_{\mathbf{1}}) \to \Gamma_1(z_\star) \to \Gamma_1(p_1) \to \kappa$, $M\, I \ldots I\, (\lambda h.I)\, u\, (\lambda h.\lambda g.g\, I)\, I =_\beta u$, and $M\, H_R \ldots H_R\, H_{\mathbf{0}}\, \pi_{\mathbf{1}}\, H_\star\, \delta_\bullet =_\beta \pi_\$$, then $M = \lambda r_1 \ldots r_L.\lambda z_{\mathbf{0}} z_{\mathbf{1}} z_\star p_1.N$ for some term N such that $N \in \mathcal{R}_1$.*

Proof []. Induction on the size of M, case analysis of the normal form, and Lemma 28. ◀

▶ **Example 30.** Assume $\mathfrak{R} = \{\mathbf{00} \Rightarrow \mathbf{22}, \mathbf{02} \Rightarrow \mathbf{11}, \mathbf{20} \Rightarrow \mathbf{11}\}$ over the alphabet $\{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$ from Example 18. Let $N := r_1\, p_2\, (r_2\, p_1\, (r_3\, p_3\, z_{\mathbf{1}}))$ and $M := z_\star\, p_1\, (\lambda p_2.z_\star\, p_2\, (\lambda p_3.z_{\mathbf{0}}\, p_3\, N))$. We have $N \in \mathcal{Q}_3$ and $M \in \mathcal{R}_1$. In congruence with Theorem 29 we have:

$$\begin{aligned}(\lambda r_1 r_2 r_3.\lambda z_{\mathbf{0}} z_{\mathbf{1}} z_\star p_1.M)\, I\, I\, I\, (\lambda h.I)\, u\, (\lambda h.\lambda g.g\, I)\, I \quad &=_\beta u\\ (\lambda r_1 r_2 r_3.\lambda z_{\mathbf{0}} z_{\mathbf{1}} z_\star p_1.M)\, H_R\, H_R\, H_R\, H_{\mathbf{0}}\, \pi_{\mathbf{1}}\, H_\star\, \delta_\bullet \quad &=_\beta \pi_\$\end{aligned}$$

While Theorem 29 only establishes "well-formedness", the term M has an intended meaning: An initial word of length 2 is expanded twice (using $z_\star$) to a word of length 4 and initialized to $\mathbf{0}$s (using $z_{\mathbf{0}}$). The introduced variables p_2 and p_3 are used to address positions in the longer word. The intended meaning of N is that the first rule (using r_1) is applied at position 2 (using p_2), followed by the second rule at position 1, and third rule at position 3 accordingly. The resulting word contains only $\mathbf{1}$s (indicated by $z_{\mathbf{1}}$). Overall, this corresponds to $\mathbf{0000} \Rightarrow_{\mathfrak{R}} \mathbf{0220} \Rightarrow_{\mathfrak{R}} \mathbf{1120} \Rightarrow_{\mathfrak{R}} \mathbf{1111}$.

Having only "well-formed" terms to consider (cf. Example 10 and Example 11), we can focus on the functional properties of rewriting.

Semantic Constraints

We formulate typed terms (Definition 31) and β-equivalence constraints characterizing word expansion (Lemma 36) and rewriting (Lemma 34). The presented terms are programs which realize the intended meaning (Example 30) of "well-formed" terms in $\mathcal{Q}_m$ and $\mathcal{R}_m$. Specifically, $G_\star$ realizes word expansion, $G_{\mathbf{0}}$ realizes initialization with $\mathbf{0}$s, $G_{ab\Rightarrow cd}$ realizes rule application, and G^i_j addresses position i for rule application at position j.

▶ **Definition 31** (Typed Terms $G_\star$ [], $G_{\mathbf{0}}$ [], $G_{ab\Rightarrow cd}$ [], G^j_i []).

$$\begin{aligned}
G_\star :=\,& \lambda h.\lambda g.\lambda s_{\mathbf{0}} s_{\mathbf{1}} \ldots s_{\mathbf{K}} s_\$ s_\bullet s_\top s_\bot.\mathbf{case}\ h\,\pi_\top\ \mathbf{of}\ \langle s_\bot\\
& \mid \bullet \mapsto \mathbf{case}\ g\,\delta_\bullet\ \mathbf{of}\ \langle s_\bot \mid \mathbf{0} \mapsto s_{\mathbf{0}} \mid \$ \mapsto \mathbf{case}\ g\,\delta_{\mathbf{0}}\ \mathbf{of}\ \langle s_\bot \mid \mathbf{1} \mapsto s_\$\rangle\rangle\\
& \mid \mathbf{0} \mapsto \mathbf{case}\ g\,\delta_{\mathbf{1}}\ \mathbf{of}\ \langle s_\bot \mid \mathbf{0} \mapsto s_{\mathbf{1}}\rangle\\
& \mid \mathbf{1} \mapsto \mathbf{case}\ g\,\delta_\bullet\ \mathbf{of}\ \langle s_\bot \mid \mathbf{0} \mapsto s_{\mathbf{0}}\rangle\rangle\\
G_{\mathbf{0}} :=\,& \lambda h.\lambda x.\lambda s_{\mathbf{0}} s_{\mathbf{1}} \ldots s_{\mathbf{K}} s_\$ s_\bullet s_\top s_\bot.\mathbf{case}\ h\,\pi_\top\ \mathbf{of}\ \langle s_\bot\\
& \mid \bullet \mapsto \mathbf{case}\ x\ \mathbf{of}\ \langle s_\bot \mid \mathbf{0} \mapsto s_{\mathbf{0}} \mid \mathbf{1} \mapsto s_\$\rangle\\
& \mid \mathbf{0} \mapsto \mathbf{case}\ x\ \mathbf{of}\ \langle s_\bot \mid \mathbf{0} \mapsto s_{\mathbf{1}}\rangle\\
& \mid \mathbf{1} \mapsto \mathbf{case}\ x\ \mathbf{of}\ \langle s_\bot \mid \mathbf{0} \mapsto s_{\mathbf{0}}\rangle\rangle\\
G_{ab\Rightarrow cd} :=\,& \lambda h.\lambda x.\lambda s_{\mathbf{0}} s_{\mathbf{1}} \ldots s_{\mathbf{K}} s_\$ s_\bullet s_\top s_\bot.\mathbf{case}\ h\,\pi_\top\ \mathbf{of}\ \langle s_\bot\\
& \mid \bullet \mapsto x\, s_{\mathbf{0}}\, s_{\mathbf{1}}\ \ldots s_{\mathbf{K}}\, s_\$\, s_\bullet\, s_\top\, s_\bot\\
& \mid \mathbf{0} \mapsto \mathbf{case}\ x\ \mathbf{of}\ \langle s_\bot \mid d \mapsto s_b\rangle\\
& \mid \mathbf{1} \mapsto \mathbf{case}\ x\ \mathbf{of}\ \langle s_\bot \mid c \mapsto s_a\rangle\rangle\\
G^i_j :=\,& \begin{cases}\delta_{\mathbf{1}} & \textit{if } i = j\\ \delta_{\mathbf{0}} & \textit{if } i = j+1\\ \delta_\bullet & \textit{else}\end{cases}
\end{aligned}$$

$$\emptyset \vdash G_\star : \Gamma_m(z_\star) \qquad \emptyset \vdash G_{\mathbf{0}} : \Gamma_m(z_{\mathbf{0}}) \qquad \emptyset \vdash G_{ab\Rightarrow cd} : \Gamma_m(r_i)\ \textit{for } i \in \{1, \ldots, L\}$$

Similarly to substitutions S_F and S_H, we introduce the following substitution S_G.

▶ **Definition 32** (Substitution S_G []).

$$S_G(z_\star) := G_\star \qquad S_G(z_{\mathbf{1}}) := \pi_{\mathbf{1}} \qquad S_G(z_{\mathbf{0}}) := G_{\mathbf{0}} \qquad S_G(r_j) := G_{R_j}\ \textit{for } j \in 1, \ldots L$$

The following Example 33 illustrates how the term $G_{ab\Rightarrow cd}$ represents rule application.

▶ **Example 33.** Consider for symbols $\mathbf{0}, \mathbf{1}, \dots, \mathbf{5}$ an application of the rule $\mathbf{12} \Rightarrow \mathbf{45}$ at position 2 in order to rewrite the word $\mathbf{0123}$ to $\mathbf{0453}$. Accordingly, we have:

Position 1: $G_{\mathbf{12}\Rightarrow\mathbf{45}}\, G_2^1\, \pi_{\mathbf{0}} =_\beta \pi_{\mathbf{0}}$
Position 2: $G_{\mathbf{12}\Rightarrow\mathbf{45}}\, G_2^2\, \pi_{\mathbf{4}} =_\beta \pi_{\mathbf{1}}$
Position 3: $G_{\mathbf{12}\Rightarrow\mathbf{45}}\, G_2^3\, \pi_{\mathbf{5}} =_\beta \pi_{\mathbf{2}}$
Position 4: $G_{\mathbf{12}\Rightarrow\mathbf{45}}\, G_2^4\, \pi_{\mathbf{3}} =_\beta \pi_{\mathbf{3}}$

The above observation is generalized for terms in $\mathcal{Q}_m$ in the following Lemma 34.

▶ **Lemma 34.** *For $m > 0$, let $a_1, \dots, a_{m+1} \in \{\mathbf{0}, \mathbf{1}, \dots, \mathbf{K}\}$ and $M \in \mathcal{Q}_m$. If $\Gamma_m \vdash M : \kappa$, $S_G(M)[p_1 := G_1^0, \dots, p_m := G_m^0] =_\beta \pi_{\mathbf{1}}$, and $S_G(M)[p_1 := G_1^i, \dots, p_m := G_m^i] =_\beta \pi_{a_i}$ for $i \in \{1, \dots, m+1\}$, then $a_1 \dots a_{m+1} \Rightarrow_{\mathfrak{R}}^* \mathbf{1}^{m+1}$.*

Proof [ ]. Induction on the size of M and case analysis using Definition 22. ◀

The following Example 35 builds upon the previous Example 30 and illustrates the intended meaning (rewriting $\mathbf{0}$s to $\mathbf{1}$s) of a "well-formed" example term in $\mathcal{Q}_3$.

▶ **Example 35.** Assume $\mathfrak{R} = \{\mathbf{00} \Rightarrow \mathbf{22}, \mathbf{02} \Rightarrow \mathbf{11}, \mathbf{20} \Rightarrow \mathbf{11}\}$ over the alphabet $\{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$, and consider the term $N = r_1\, p_2\, (r_2\, p_1\, (r_3\, p_3\, z_{\mathbf{1}}))$ from Example 30. Replacing G_j^i accordingly for $i \in \{0, \dots, 5\}$ and $j \in \{1, 2, 3\}$, we have the following β-equivalences $(0) - (4)$.

$$\begin{array}{ll}
(0) & S_G(N)[p_1 := \delta_\bullet, p_2 := \delta_\bullet, p_3 := \delta_\bullet] =_\beta \pi_{\mathbf{1}} \\
(1) & S_G(N)[p_1 := \delta_{\mathbf{1}}, p_2 := \delta_\bullet, p_3 := \delta_\bullet] =_\beta \pi_{\mathbf{0}} \\
(2) & S_G(N)[p_1 := \delta_{\mathbf{0}}, p_2 := \delta_{\mathbf{1}}, p_3 := \delta_\bullet] =_\beta \pi_{\mathbf{0}} \\
(3) & S_G(N)[p_1 := \delta_\bullet, p_2 := \delta_{\mathbf{0}}, p_3 := \delta_{\mathbf{1}}] =_\beta \pi_{\mathbf{0}} \\
(4) & S_G(N)[p_1 := \delta_\bullet, p_2 := \delta_\bullet, p_3 := \delta_{\mathbf{0}}] =_\beta \pi_{\mathbf{0}}
\end{array}$$

In accordance with Lemma 34, we have that $\mathbf{0}^4 \Rightarrow_{\mathfrak{R}}^* \mathbf{1}^4$. Equivalences $(1) - (4)$ witness the particular rewriting steps at positions $1 - 4$ (cf. Example 30 and Example 33).

Complementarily to word rewriting, the following Lemma 36 characterizes word expansion and initialization with $\mathbf{0}$s.

▶ **Lemma 36.** *If $M \in \mathcal{R}_1$ such that $\Gamma_1 \vdash M : \kappa$,
$S_G(M)[p_1 := \delta_\bullet] =_\beta \pi_\$$, $S_G(M)[p_1 := \delta_{\mathbf{1}}] =_\beta \pi_{\mathbf{0}}$, and $S_G(M)[p_1 := \delta_{\mathbf{0}}] =_\beta \pi_{\mathbf{1}}$,
then there exists an $m > 0$ and an $N \in \mathcal{Q}_m$ such that $\Gamma_m \vdash N : \kappa$,
$S_G(N)[p_1 := G_1^0, \dots, p_m := G_m^0] =_\beta \pi_{\mathbf{1}}$, and $S_G(N)[p_1 := G_1^i, \dots, p_m := G_m^i] =_\beta \pi_{\mathbf{0}}$ for $i \in \{1, \dots, m+1\}$.*

Proof [ ]. Considering the general case $M \in \mathcal{R}_{m'}$ for $m' > 0$, induction on the size of M and case analysis using Definition 22. ◀

The following Example 37 complements the previous Example 35 and illustrates the intended meaning (word expansion and initialization with $\mathbf{0}$s) of a "well-formed" term in $\mathcal{R}_1$.

▶ **Example 37.** Assume $\mathfrak{R} = \{\mathbf{00} \Rightarrow \mathbf{22}, \mathbf{02} \Rightarrow \mathbf{11}, \mathbf{20} \Rightarrow \mathbf{11}\}$ over the alphabet $\{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$, and consider the terms $M_1 := z_\star\, p_1\, (\lambda p_2.M_2)$, $M_2 := z_\star\, p_2\, (\lambda p_3.M_3)$, $M_3 := z_{\mathbf{0}}\, p_3\, N$, and $N := r_1\, p_2\, (r_2\, p_1\, (r_3\, p_3\, z_{\mathbf{1}}))$ from Example 35. Proceeding bottom up, we have $M_3 \in \mathcal{R}_3$ and the following β-equivalences hold:

$$\begin{array}{l}
S_G(M_3)[p_1 := \delta_\bullet, p_2 := \delta_\bullet, p_3 := \delta_\bullet] =_\beta \pi_\$ \\
S_G(M_3)[p_1 := \delta_{\mathbf{1}}, p_2 := \delta_\bullet, p_3 := \delta_\bullet] =_\beta \pi_{\mathbf{0}} \\
S_G(M_3)[p_1 := \delta_{\mathbf{0}}, p_2 := \delta_{\mathbf{1}}, p_3 := \delta_\bullet] =_\beta \pi_{\mathbf{0}} \\
S_G(M_3)[p_1 := \delta_\bullet, p_2 := \delta_{\mathbf{0}}, p_3 := \delta_{\mathbf{1}}] =_\beta \pi_{\mathbf{0}} \\
S_G(M_3)[p_1 := \delta_\bullet, p_2 := \delta_\bullet, p_3 := \delta_{\mathbf{0}}] =_\beta \pi_{\mathbf{1}}
\end{array}$$

Additionally, $M_2 \in \mathcal{R}_2$, $M_1 \in \mathcal{R}_1$, and the following β-equivalences hold:

$$\begin{array}{ll} S_G(M_2)[p_1 := \delta_\bullet, p_2 := \delta_\bullet] =_\beta \pi_\$ & \\ S_G(M_2)[p_1 := \delta_\mathbf{1}, p_2 := \delta_\bullet] =_\beta \pi_\mathbf{0} & S_G(M_1)[p_1 := \delta_\bullet] =_\beta \pi_\$ \\ S_G(M_2)[p_1 := \delta_\mathbf{0}, p_2 := \delta_\mathbf{1}] =_\beta \pi_\mathbf{0} & S_G(M_1)[p_1 := \delta_\mathbf{1}] =_\beta \pi_\mathbf{0} \\ S_G(M_2)[p_1 := \delta_\bullet, p_2 := \delta_\mathbf{0}] =_\beta \pi_\mathbf{1} & S_G(M_1)[p_1 := \delta_\mathbf{0}] =_\beta \pi_\mathbf{1} \end{array}$$

In combination with the previous Example 35, the term $M_1 \in \mathcal{R}_1$ represents word expansion up to length 4, followed by initialization with $\mathbf{0}$s, and rewriting to $\mathbf{1}$s.

Next, we combine syntactic and semantic constraints in the following key Lemma 38.

▶ **Lemma 38.** *There exists an $n \in \mathbb{N}$ such that $\mathbf{0}^{n+1} \Rightarrow^*_\mathfrak{R} \mathbf{1}^{n+1}$ iff there exists a term M in normal form, such that the following conditions hold:*

$\emptyset \vdash M : \Gamma_1(r_1) \to \cdots \to \Gamma_1(r_L) \to \Gamma_1(z_\mathbf{0}) \to \Gamma_1(z_\mathbf{1}) \to \Gamma_1(z_\star) \to \Gamma_1(p_1) \to \kappa,$

$$\begin{array}{ll} M\, I \ldots I\, (\lambda h.I)\, u\, (\lambda h.\lambda g.g\, I)\, I & =_\beta u, \\ M\, H_R \ldots H_R\, H_\mathbf{0}\, \pi_\mathbf{1}\, H_\star\, \delta_\bullet & =_\beta \pi_\$, \\ M\, G_{R_1} \ldots G_{R_L}\, G_\mathbf{0}\, \pi_\mathbf{1}\, G_\star\, \delta_\bullet & =_\beta \pi_\$, \\ M\, G_{R_1} \ldots G_{R_L}\, G_\mathbf{0}\, \pi_\mathbf{1}\, G_\star\, \delta_\mathbf{1} & =_\beta \pi_\mathbf{0}, \\ M\, G_{R_1} \ldots G_{R_L}\, G_\mathbf{0}\, \pi_\mathbf{1}\, G_\star\, \delta_\mathbf{0} & =_\beta \pi_\mathbf{1}. \end{array}$$

Proof [ ]. The direction from left to right proceeds in two steps. First, by induction on the number of rewriting steps we construct a term $N \in \mathcal{Q}_n$ (easy converse of Lemma 34). Second, by induction on n we construct a term $M' \in \mathcal{R}_1$ containing $N \in \mathcal{Q}_n$ as a subterm (easy converse of Lemma 36). Then, the solution is $M := \lambda r_1 \ldots r_L.\lambda z_\mathbf{0} z_\mathbf{1} z_\star p_1.M'$.

The direction from right to left is proceeds in two steps. First, by Theorem 29 we have $\lambda r_1 \ldots r_L.\lambda z_\mathbf{0} z_\mathbf{1} z_\star p_1.M'$ for some $M' \in \mathcal{R}_1$. Second, by Lemma 36 and Lemma 34 we have $\mathbf{0}^{n+1} \Rightarrow^*_\mathfrak{R} \mathbf{1}^{n+1}$ for some $n \in \mathbb{N}$. ◀

Finally, we present the combination of constraints from the above Lemma 38 as a matching instance $F_\mathfrak{R}\, \mathsf{X} =_\beta N_\mathfrak{R}$. This constitutes the main result of the present work.

▶ **Theorem 39.** *Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ many-one reduces to higher-order β-matching.*

Proof [ ]. Given a simple semi-Thue system $\mathfrak{R} = \{R_1, \ldots, R_L\}$ due to Lemma 38 there exists an $n \in \mathbb{N}$ such that $\mathbf{0}^{n+1} \Rightarrow^*_\mathfrak{R} \mathbf{1}^{n+1}$ iff the instance $F_\mathfrak{R}\, \mathsf{X} =_\beta N_\mathfrak{R}$ of higher-order β-matching is solvable, where

- $F_\mathfrak{R} := \lambda x.\lambda y.y\, (\lambda u.x\, \underbrace{I \ldots I}_{L \text{ times}}\, (\lambda h.I)\, u\, (\lambda h.\lambda g.g\, I)\, I)$
  $(x\, \underbrace{H_R \ldots H_R}_{L \text{ times}}\, H_\mathbf{0}\, \pi_\mathbf{1}\, H_\star\, \delta_\bullet)$
  $(x\, G_{R_1} \ldots G_{R_L}\, G_\mathbf{0}\, \pi_1\, G_\star\, \delta_\bullet)$
  $(x\, G_{R_1} \ldots G_{R_L}\, G_\mathbf{0}\, \pi_1\, G_\star\, \delta_\mathbf{1})$
  $(x\, G_{R_1} \ldots G_{R_L}\, G_\mathbf{0}\, \pi_1\, G_\star\, \delta_\mathbf{0})$
- $N_\mathfrak{R} := \lambda y.y\, (\lambda u.u)\, \pi_\$\, \pi_\$\, \pi_\mathbf{0}\, \pi_\mathbf{1}$
- $\sigma_\mathfrak{R} := \Gamma_1(r_1) \to \cdots \to \Gamma_1(r_L) \to \Gamma_1(z_\mathbf{0}) \to \Gamma_1(z_\mathbf{1}) \to \Gamma_1(z_\star) \to \Gamma_1(p_1) \to \kappa$
- $\tau_\mathfrak{R} := ((\kappa \to \kappa) \to \kappa \to \kappa \to \kappa \to \kappa \to \iota) \to \iota$
- $\emptyset \vdash F_\mathfrak{R} : \sigma_\mathfrak{R} \to \tau_\mathfrak{R}$
- $\emptyset \vdash N_\mathfrak{R} : \tau_\mathfrak{R}$ ◀

▶ **Theorem 40.** *Higher-order β-matching (Problem 7) is undecidable.*

Proof [ ]. By reduction from the undecidable Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ (Theorem 17 and Theorem 39). ◀

4 Mechanization

This section provides a brief overview over the mechanization of undecidability of higher-order β-matching (Theorem 40) using the Coq proof assistant [19]. The mechanization is axiom-free and spans approximately 4000 lines of code, consisting of the following parts:

- `HOMatching.v` contains definitions of the simply typed λ-calculus [] and higher-order β-matching [].
- `Util/stlc_facts.v` and `Util/term_facts.v` contain basic properties of the simply typed λ-calculus, such as confluence of β-reduction [], substitution lemmas [], and type preservation properties [].
- `Reductions/SSTS01_to_HOMbeta.v` contains the reduction from Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ to higher-order β-matching [].
- `HOMatching_undec.v` contains the undecidability result for higher-order β-matching [].

The simple type system `stlc` is mechanized in `HOMatching.v`, borrowing the existing term definitions from the library [], for which variable binding is addressed via the unscoped de Bruijn approach [4]. The proposition `stlc Gamma M t` mechanizes that the term `M` is assigned the simple type `t` in the simple type environment `Gamma`.

```
Inductive ty : Type :=
  | atom (* type variable *)
  | arr (s t : ty). (* function type *)

Inductive term : Type :=
  | var (n : nat) : term (* term variable *)
  | app (s : term) (t : term) : term (* application *)
  | lam (s : term). (* abstraction *)

Inductive stlc (Gamma : list ty) : term -> ty -> Prop :=
  | stlc_var x t : nth_error Gamma x = Some t ->
      stlc Gamma (var x) t (* variable rule *)
  | stlc_app M N s t : stlc Gamma M (arr s t) -> stlc Gamma N s ->
      stlc Gamma (app M N) t (* application rule *)
  | stlc_lam M s t : stlc (cons s Gamma) M t ->
      stlc Gamma (lam M) (arr s t). (* abstraction rule *)
```

Higher-order β-matching is mechanized as the predicate `HOMbeta`: given terms `F` of type `arr s t` and `N` of type `t`, is there a simply typed term `M` of type `s` such that `app F M` is β-equivalent (reflexive, symmetric, transitive closure of `step`) to `N`?

```
Definition HOMbeta : { '(s, t, F, N) : (ty * ty * term * term)
  | stlc nil F (arr s t) /\ stlc nil N t } -> Prop :=
    fun '(exist _ (s, t, F, N) _) =>
      exists (M : term), stlc nil M s /\
        clos_refl_sym_trans term step (app F M) N.
```

The proposition `undecidable HOMbeta` [] mechanizes the undecidability of the predicate `HOMbeta`, relying on the following library definition [7, Chapter 19]. A predicate `p` is undecidable, if existence of a computable decider for `p` implies recursive co-enumerability of the (Turing machine) Halting Problem.

```
Definition undecidable {X} (p : X -> Prop) :=
  decidable p -> enumerable (complement SBTM_HALT).
```

Since the Halting Problem is recursively enumerable, decidability of `p` would imply decidability of the Halting Problem.

5 On Intersection Type Inhabitation and λ-Definability

We conclude the technical presentation with the following observation: the presented approach reducing Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ to higher-order β-matching is easily transferred to intersection type inhabitation and λ-definability.

The following Remark 41 shows the structure of the corresponding finite model with respect to the present construction.

▶ Remark 41. Terms in Definition 31 realize certain finite functions as follows.

- δ_i for $i \in \mathcal{A}$ realizes a member of the finite function family specified by the partial function table $\big(\top \mapsto i\big)$.
- $G_{\mathbf{0}}$ realizes a member of the family specified by $\begin{pmatrix} (\top \mapsto \bullet) \mapsto (\mathbf{0} \mapsto \mathbf{0}) \\ (\top \mapsto \bullet) \mapsto (\mathbf{1} \mapsto \$) \\ (\top \mapsto \mathbf{0}) \mapsto (\mathbf{0} \mapsto \mathbf{1}) \\ (\top \mapsto \mathbf{1}) \mapsto (\mathbf{0} \mapsto \mathbf{0}) \end{pmatrix}$.
- $G_{ab \Rightarrow cd}$ realizes a member of the family specified by $\begin{pmatrix} (\top \mapsto \mathbf{1}) \mapsto (c \mapsto a) \\ (\top \mapsto \mathbf{0}) \mapsto (d \mapsto b) \end{pmatrix}$.
- $G_\star$ realizes a member of the family specified by $\begin{pmatrix} (\top \mapsto \bullet) \mapsto ((\top \mapsto \bullet) \mapsto \mathbf{0}) \mapsto \mathbf{0} \\ (\top \mapsto \bullet) \mapsto \left(\begin{pmatrix} (\top \mapsto \bullet) \mapsto \$ \\ (\top \mapsto \mathbf{0}) \mapsto \mathbf{1} \end{pmatrix} \mapsto \$\right) \\ (\top \mapsto \mathbf{0}) \mapsto ((\top \mapsto \mathbf{1}) \mapsto \mathbf{0}) \mapsto \mathbf{1} \\ (\top \mapsto \mathbf{1}) \mapsto ((\top \mapsto \bullet) \mapsto \mathbf{0}) \mapsto \mathbf{0} \end{pmatrix}$.

The above specifications follow the intended meaning (Example 30) of the corresponding programs, when used in "well-formed" terms in $\mathcal{Q}_m$ and $\mathcal{R}_m$. For example, we have $G_{\mathbf{0}}\, \delta_\bullet\, \pi_1 =_\beta \pi_\$$, in agreement with the above Remark 41.

Let us state the relationship between simple semi-Thue system rewriting, higher-order β-matching, λ-definability, and intersection type inhabitation.

▶ **Proposition 42.** *Given a simple semi-Thue system $\mathfrak{R}$, one can construct simply typed terms $F_\mathfrak{R}$ and $N_\mathfrak{R}$, an intersection type $T_\mathfrak{R}$, and a finite function $\mathcal{F}_\mathfrak{R}$ such that the following statements are equivalent:*

1. *There exists an $n \in \mathbb{N}$ such that $\mathbf{0}^{n+1} \Rightarrow^*_\mathfrak{R} \mathbf{1}^{n+1}$.*
2. *The instance $F_\mathfrak{R}\, \mathsf{X} =_\beta N_\mathfrak{R}$ of higher-order β-matching is solvable.*
3. *The intersection type $T_\mathfrak{R}$ is inhabited.*
4. *The finite function $\mathcal{F}_\mathfrak{R}$ is λ-definable.*

The presented approach shows (1) $\iff$ (2). Of course, (1) $\iff$ (3) can be concluded from undecidability of intersection type inhabitation [20] and (1) $\iff$ (4) from undecidability of λ-definability [12], along with corresponding constructions. However, we make the following two observations regarding an alternative, uniform argument. First, based on Remark 23, the approach is easily adapted to show (1) $\iff$ (3), such that the inhabitant is essentially a member of $\mathcal{R}_1$. This is already done in the existing mechanized reduction from Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ to intersection type inhabitation [ ]. Second, based on Remark 41, the approach can be adapted to show (1) $\iff$ (4), such that the realizer is essentially a member of $\mathcal{R}_1$. This is further supported by the known correspondence between intersection type inhabitation (in the fragment at hand) and λ-definability [15].

6 Conclusion

The present work presents a new, mechanized proof of the undecidability of higher-order β-matching. The mechanization is contributed to the existing Coq Library of Undecidability Proofs [8].

While the existing proofs by Loader [13] and by Joly [11] are each based on variants of λ-definability, the presented proof reduces a rewriting problem (Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$) to higher-order β-matching. As a result, the proof is simpler to verify in full detail and yields a concise mechanization. Additionally, undecidability of Problem $\mathbf{0}^+ \Rightarrow^* \mathbf{1}^+$ is already mechanized, and is part the Coq Library of Undecidability Proofs.

Besides the main technical result, we argue that the present approach is uniformly applicable to show undecidability of intersection type inhabitation and λ-definability. The former is already established and implemented as refinement [6] of Urzyczyn's undecidability result [20]. The latter is an application of the known correspondence between intersection type inhabitation and λ-definability [15].

The *order* of a type is the maximal nesting depth of the arrow type constructor to the left, starting by $\text{order}(\iota) = 1$. The present approach agrees with Loader's result that β-matching is undecidable at order 6. While Loader conjectures that order 5 may suffice, neither Loader's technique (as observed by Joly [11, Section 5]), nor the present approach are applicable at order 5. Constraining the shape of candidate solutions both in the present work as well as in Loader's proof seems to necessitate order 6. While β-matching at order 4 is decidable [14], decidability at order 5 remains an open question.

As pointed out in Remark 13, the presented approach might be adapted to scenarios beyond the simply typed λ-calculus. An interesting alternative to the simple type system is the Coppo-Dezani intersection type assignment system [3], which characterizes strong normalization [1]. Well-typedness in this system would allow for more solution candidates and require more effort with respect to syntactic constraints (cf. Section 3). It is reasonable to believe that higher-order β-matching is undecidable in any type system for the λ-calculus which includes the simple type system.

Interaction with a proof assistant supported the formation process of the present approach. The existing infrastructure for the λ-calculus provided by the undecidability library served as an excellent starting point for the development. While proofs of the individual lemmas (cf. Section 3) in the development are quite simplistic, they involve exhaustive case analyses and are sensitive to the exact details of the underlying construction. Bookkeeping capabilities of the Coq proof assistant, proof automation based on `auto` and `lia` tactics, and quick adaptability to an evolving construction were of great benefit. Additionally, once all cases are covered, there is no room for doubt that the construction is correct. As a result, the proof was developed via interaction with the proof assistant prior to being transcribed into a traditional written format.

References

1. Roberto M. Amadio and Pierre-Louis Curien. *Domains and lambda-calculi*. Number 46. Cambridge University Press, 1998.
2. Hendrik Pieter Barendregt, Wil Dekkers, and Richard Statman. *Lambda Calculus with Types*. Perspectives in logic. Cambridge University Press, 2013. URL: http://www.cambridge.org/de/academic/subjects/mathematics/logic-categories-and-sets/lambda-calculus-types.

3 Mario Coppo and Mariangiola Dezani-Ciancaglini. An extension of the basic functionality theory for the lambda-calculus. *Notre Dame Journal of Formal Logic*, 21(4):685 – 693, 1980. doi:10.1305/ndjfl/1093883253.

4 Nicolaas Govert De Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. In *Indagationes Mathematicae (Proceedings)*, volume 75, pages 381–392. North-Holland, 1972.

5 Gilles Dowek. Higher-order unification and matching. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 1009–1062. Elsevier and MIT Press, 2001. URL: https://doi.org/10.1016/b978-044450813-3/50018-7, doi:10.1016/B978-044450813-3/50018-7.

6 Andrej Dudenhefner and Jakob Rehof. Undecidability of intersection type inhabitation at rank 3 and its formalization. *Fundam. Informaticae*, 170(1-3):93–110, 2019. doi:10.3233/FI-2019-1856.

7 Yannick Forster. Computability in constructive type theory. 2021. doi:http://dx.doi.org/10.22028/D291-35758.

8 Yannick Forster, Dominique Larchey-Wendling, Andrej Dudenhefner, Edith Heiter, Dominik Kirst, Fabian Kunze, Gert Smolka, Simon Spies, Dominik Wehr, and Maximilian Wuttke. A Coq library of undecidable problems. In *The Sixth International Workshop on Coq for Programming Languages (CoqPL 2020)*, 2020. URL: https://github.com/uds-psl/coq-library-undecidability.

9 Gérard P. Huet. The undecidability of unification in third order logic. *Inf. Control.*, 22(3):257–267, 1973. doi:10.1016/S0019-9958(73)90301-X.

10 Gérard P. Huet. A unification algorithm for typed lambda-calculus. *Theor. Comput. Sci.*, 1(1):27–57, 1975. doi:10.1016/0304-3975(75)90011-0.

11 Thierry Joly. On lambda-definability I: the fixed model problem and generalizations of the matching problem. *Fundam. Informaticae*, 65(1-2):135–151, 2005. URL: http://content.iospress.com/articles/fundamenta-informaticae/fi65-1-2-07.

12 Ralph Loader. The undecidability of λ-definability. In *Logic, meaning and computation: Essays in memory of Alonzo church*, pages 331–342. Springer, 2001.

13 Ralph Loader. Higher order beta matching is undecidable. *Log. J. IGPL*, 11(1):51–68, 2003. URL: https://doi.org/10.1093/jigpal/11.1.51, doi:10.1093/JIGPAL/11.1.51.

14 Vincent Padovani. Decidability of fourth-order matching. *Math. Struct. Comput. Sci.*, 10(3):361–372, 2000. URL: http://journals.cambridge.org/action/displayAbstract?aid=44653.

15 Sylvain Salvati, Giulio Manzonetto, Mai Gehrke, and Henk Barendregt. Loader and Urzyczyn are logically related. In Artur Czumaj, Kurt Mehlhorn, Andrew M. Pitts, and Roger Wattenhofer, editors, *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, volume 7392 of *Lecture Notes in Computer Science*, pages 364–376. Springer, 2012. doi:10.1007/978-3-642-31585-5_34.

16 Simon Spies and Yannick Forster. Undecidability of higher-order unification formalised in Coq. In Jasmin Blanchette and Catalin Hritcu, editors, *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2020, New Orleans, LA, USA, January 20-21, 2020*, pages 143–157. ACM, 2020. doi:10.1145/3372885.3373832.

17 Richard Statman. Completeness, invariance and lambda-definability. *J. Symb. Log.*, 47(1):17–26, 1982. doi:10.2307/2273377.

18 Colin Stirling. Decidability of higher-order matching. *Log. Methods Comput. Sci.*, 5(3), 2009. URL: http://arxiv.org/abs/0907.3804.

19 The Coq Development Team. The Coq proof assistant, July 2023. doi:10.5281/zenodo.8161141.

20 Paweł Urzyczyn. Inhabitation of low-rank intersection types. In Pierre-Louis Curien, editor, *Typed Lambda Calculi and Applications, 9th International Conference, TLCA 2009, Brasilia, Brazil, July 1-3, 2009. Proceedings*, volume 5608 of *Lecture Notes in Computer Science*, pages 356–370. Springer, 2009. doi:10.1007/978-3-642-02273-9_26.