Lower End of the Linial-Post Spectrum

Andrej Dudenhefner
Technical University of Dortmund, Dortmund, Germany
andrej.dudenhefner@cs.tu-dortmund.de

Jakob Rehof
Technical University of Dortmund, Dortmund, Germany
jakob.rehof@cs.tu-dortmund.de

Abstract

We show that recognizing axiomatizations of the Hilbert-style calculus containing only the axiom $a \to (b \to a)$ is undecidable (a reduction from the Post correspondence problem is formalized in the Lean theorem prover). Interestingly, the problem remains undecidable considering only axioms which, when seen as simple types, are principal for some λ-terms in β-normal form. This problem is closely related to type-based composition synthesis, i.e. finding a composition of given building blocks (typed terms) satisfying a desired specification (goal type).

Contrary to the above result, axiomatizations of the Hilbert-style calculus containing only the axiom $a \to (b \to b)$ are recognizable in linear time.

2012 ACM Subject Classification Theory of computation → Proof theory

Keywords and phrases combinatory logic, lambda calculus, type theory, simple types, inhabitation, principal type

Digital Object Identifier 10.4230/LIPIcs.TYPES.2017.2

Acknowledgements We thank the anonymous reviewers for their comprehensive and particularly constructive comments.

1 Introduction

The problem of decidability of a given Hilbert-style propositional calculus was posed by Tarski in 1946 and has subsequently been studied by several authors in various forms, beginning with the abstract published in 1949 by Linial and Post [13] in which the existence of an undecidable propositional calculus and undecidability of the problem of recognizing axiomatizations of classical propositional logic was stated (Linial-Post theorems). Zolin [20] provides a good overview of the history of this problem, and let us mention here only that Singletary [16] proved in 1974 that there exists a purely implicational propositional calculus which can represent any r.e. degree. Such a calculus can be seen as a combinatory logic [1, 11] in simple types [10, 2] with an undecidable inhabitation problem. In honor of the pioneers, we refer to the set of all Hilbert-style propositional calculi as the Linial-Post spectrum. In the present work we shed some light on the 'lower end' (seemingly very weak calculi) of the Linial-Post spectrum from the point of view of functional program synthesis. Our main result is that recognizing axiomatizations of the Hilbert-style calculus containing only the axiom $a \to (b \to a)$ (the type of the combinator $\mathbf{K}$ in combinatory logic) is undecidable. Moreover, we show that the problem remains undecidable considering only *principal* axioms, i.e. axioms that, seen as simple types, are principal for some λ-terms in β-normal form. In general, to recognize whether given axioms $\Delta = \{\sigma_1, \ldots, \sigma_n\}$ axiomatize some calculus $\mathfrak{C}$ means to decide whether theorems in $\mathfrak{C}$ coincide with formulae derivable from Δ using the rules of substitution and modus ponens. In particular, if $\mathfrak{C}$ is the Hilbert-style calculus

23rd International Conference on Types for Proofs and Programs (TYPES 2017).
Editors: Andreas Abel, Fredrik Nordvall Forsberg, and Ambrus Kaposi; Article No. 2; pp. 2:1–2:15
LIPICS Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

containing only the axiom $a \to (b \to a)$, then Δ axiomatizes $\mathfrak{C}$ iff $a \to (b \to a)$ is derivable from Δ and σ is derivable from $\{a \to (b \to a)\}$ for each $\sigma \in \Delta$. We show that the former problem, even if only principal axioms are considered, is undecidable while the latter, in general, is decidable in linear time[1]. If one is not interested in principal axioms, then the former result follows directly from the recent work by Bokov [6] which shows that problems of recognizing axiomatizations and completeness are undecidable for propositional calculi containing the axiom $a \to (b \to a)$.

The result presented here encompasses two motivating aspects which distinguish it from existing work (see [20] for an overview). First, similar to Bokov [6], we explore the lower end of the Linial-Post spectrum, whereas existing work focuses on classical [13, 19, 4] or superintuitionistic [12, 20, 5] calculi, often having rich type syntax, e.g. containing negation. In this work, we consider only implicational formulae and stay below $a \to (b \to a)$ in terms of derivability. This is arguably 'as low as you can get' because, as will be shown, axiomatizations of $a \to a$ (or even $a \to (b \to b)$) are recognizable in linear time. Second, we are interested in synthesis of functional programs from given building blocks. Following the same motivation as [15, 3], we want to utilize proof search (inhabitation in combinatory logics) to synthesize code by composition from a given collection of combinators. Specifically, provided simply typed λ-terms $M_1, \dots, M_n$ in β-normal form, we search for an applicative composition of the given terms that has some designated simple type σ. This is equivalent to proof search in the Hilbert-style calculus having axioms $\sigma_1, \dots, \sigma_n$ where σ_i is the principal type of M_i for $i = 1 \dots n$. It is a typical synthesis scenario, in which $M_1, \dots, M_n$ are library components exposing functionality specified by their corresponding principal types $\sigma_1, \dots, \sigma_n$. The synthesized composition is a functional program that uses the library to realize behavior specified by the type σ.

Our second motivation forces us to deviate from standard constructions pervading existing work. For example, considering axioms $a \to (a \to a)$ (testing equality of two arguments) or $(a \to b) \to b$ (encoding disjunction), there are no λ-terms in β-normal form having such axioms as their principal types. Therefore, such logical formulae could be considered an artificial and purely logical peculiarity from the point of view of program synthesis. Moreover, necessarily deviating from existing techniques (also using the Post correspondence problem instead of Post production systems as in [6]) we provide a novel and formalized[2] proof.

A noteworthy side effect when considering axioms that are tied to corresponding λ-terms via principality is an additional twist to the Curry-Howard isomorphism. We observe that the constructed proof that a formula is derivable (therefore logically solving the underlying problem) corresponds to a λ-term that actually solves the underlying problem computationally.

The paper is organized as follows. Section 2 recapitulates preliminary definitions (simply typed λ-calculus, simply typed combinatory logic, Hilbert-style calculi and the Post correspondence problem). Our main result on undecidability of recognizing axiomatizations of $a \to (b \to a)$ in shown in Section 3, which also contains linear time derivability from $a \to (b \to a)$. The proof is formalized[2] in the Lean theorem prover. Formalizations of key statements are referred to by `namespace.lemma`. Complementary, linear time decidability of recognizing axiomatizations of $a \to a$ (resp. $a \to (b \to b)$) are shown in Section 4 (resp. Section 5). We conclude the paper in Section 6 which also contains remarks on future work.

---

[1] As is well known, adding the axiom $(a \to (b \to c)) \to ((a \to b) \to (a \to c))$, the type of the combinator **S**, results in intuitionistic implicational logic for which provability is PSPACE-complete [18].

[2] `http://www-seal.cs.tu-dortmund.de/seal/downloads/research/TYPES17.zip`

2 Preliminaries

In this section we briefly assemble necessary prerequisites in order to discuss principal axiomatizations of implicational propositional calculi. For a survey on simply typed calculi along with corresponding (under the Curry-Howard isomorphism) implicational propositional calculi see [17].

2.1 Simply Typed Lambda Calculus

We denote λ-*terms* (cf. Definition 1) by M, N, L where *term variables* are denoted by x, y, z. As is usual, application associates to the left and binds stronger than abstraction.

▶ **Definition 1** (λ-Terms). $M, N, L ::= x \mid (\lambda x.M) \mid (M\ N)$

Simple types (cf. Definition 2) are denoted by σ, τ where *type atoms* (also called *type variables* in literature) are denoted by a, b, c and drawn from the denumerable set $\mathbb{A}$. As is usual, $\to$ associates to the right, i.e. $\sigma \to \tau \to \sigma = \sigma \to (\tau \to \sigma)$.

▶ **Definition 2** (Simple Types). $\mathbb{T} \ni \sigma, \tau ::= a \mid \sigma \to \tau$

A *type environment* Γ is a finite set of *type assumptions* $\{x_1 : \sigma_1, \ldots, x_n : \sigma_n\}$ in which term variables occur at most once. We set

$$\mathrm{dom}(\Gamma) = \{x_1, \ldots, x_n\}, \qquad |\Gamma| = \{\sigma_1, \ldots, \sigma_n\}, \qquad \Gamma(x_i) = \sigma_i \text{ for } i = 1 \ldots n$$

We write $\Gamma \cup \{x : \sigma\}$ as $\Gamma, x : \sigma$ if $x \notin \mathrm{dom}(\Gamma)$.

The rules (Var), ($\to$I) and ($\to$E) of the *simple type system* ($\vdash$) are given in the following Definition 3.

▶ **Definition 3** (Simply Typed λ-Calculus (⊢)).

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

A *substitution* $\zeta : \mathbb{A} \to \mathbb{T}$ is a mapping such that its *substitution domain* $\mathrm{dom}(\zeta) := \{a \in \mathbb{A} \mid \zeta(a) \neq a\}$ is finite. We lift substitutions homomorphically to types. We say σ is *unifiable* with τ, if there exists a substitution ζ such that $\zeta(\sigma) = \zeta(\tau)$. A *principal type* (cf. Definition 4) of a term is the most general type assignable to that term and is unique up to atom renaming.

▶ **Definition 4** (Principal Type). We say that τ is a *principal type* of M, if $\vdash M : \tau$ and for all types σ such that $\vdash M : \sigma$ there exists a substitution ζ such that $\zeta(\tau) = \sigma$.

2.2 Simply Typed Combinatory Logic

We call λ-terms without abstractions *combinatory terms* (cf. Definition 5), denoted by F, G, H.

▶ **Definition 5** (Combinatory Terms). $F, G, H ::= x \mid (F\ G)$

The *size* of a combinatory term is the number of leaves in its syntax tree (cf. Definition 6).

▶ **Definition 6** (Size). $\mathrm{size}(x) = 1$ and $\mathrm{size}(F\ G) = \mathrm{size}(F) + \mathrm{size}(G)$.

The rules (Ax) and ($\to$E) of the *simply typed combinatory logic* ($\vdash_{\mathcal{C}}$) are given in the following Definition 7.

▶ **Definition 7** (Simply Typed Combinatory Logic ($\vdash_{\mathcal{C}}$)).

$$\frac{\zeta \text{ is a substitution}}{\Gamma, x : \sigma \vdash_{\mathcal{C}} x : \zeta(\sigma)} \text{(Ax)} \qquad \frac{\Gamma \vdash_{\mathcal{C}} F : \sigma \to \tau \qquad \Gamma \vdash_{\mathcal{C}} G : \sigma}{\Gamma \vdash_{\mathcal{C}} F\ G : \tau} (\to\text{E})$$

Observe that the above definition is *relativized* to arbitrary bases Γ whereas 'simply typed combinatory logic' often refers to a fixed base containing the combinators **S** and **K** with their corresponding types. We will use relativization to inspect arbitrary bases allowing to go below **S** and **K** in term of derivability.

Naturally, combinatory terms are of shape $x\ F_1 \dots F_n$ for some $n \in \mathbb{N}$ and we have the following generation lemma (cf. `derivation.long_typability`).

▶ **Lemma 8** (Generation Lemma). *If $\Gamma \vdash_{\mathcal{C}} x\ F_1 \dots F_n : \tau$, then there exists a substitution ζ such that $\zeta(\Gamma(x)) = \sigma_1 \to \dots \to \sigma_n \to \tau$ for some $n \in \mathbb{N}$, $\sigma_1, \dots, \sigma_n$ and $\Gamma \vdash_{\mathcal{C}} F_i : \sigma_i$ holds for $i = 1 \dots n$.*

To mitigate extensive use of parentheses in combinatory terms we use the left-associative *pipe* metaoperator $\triangleright$ defined as $F \triangleright G = (G\ F)$. For example, $G_3\ (G_2\ (G_1\ F)) = F \triangleright G_1 \triangleright G_2 \triangleright G_3$.

2.3 Hilbert-Style Calculus

We identify propositional implicational *axioms* (sometimes called *formulae*) with simple types and denote finite sets of axioms by Δ. The rules (Ax) and ($\to$E) of the *Hilbert-style calculus* ($\vdash_{\mathcal{C}}$) are given in the following Definition 9.

▶ **Definition 9** (Hilbert-Style Calculus ($\vdash_{\mathcal{H}}$)).

$$\frac{\zeta \text{ is a substitution}}{\Delta, \sigma \vdash_{\mathcal{H}} \zeta(\sigma)} \text{(Ax)} \qquad \frac{\Delta \vdash_{\mathcal{H}} \sigma \to \tau \qquad \Delta \vdash_{\mathcal{H}} \sigma}{\Delta \vdash_{\mathcal{H}} \tau} (\to\text{E})$$

Again, ($\vdash_{\mathcal{H}}$) is relativized to arbitrary sets of axioms Δ. Observe that ($\vdash_{\mathcal{H}}$) and ($\vdash_{\mathcal{C}}$) are in direct Curry-Howard correspondence. The set of *derivable* formulae is denoted by $[\Delta]_{\mathcal{H}} = \{\tau \in \mathbb{T} \mid \Delta \vdash_{\mathcal{H}} \tau\}$.

We say Δ_1 *axiomatizes* $[\Delta_2]_{\mathcal{H}}$ if $[\Delta_1]_{\mathcal{H}} = [\Delta_2]_{\mathcal{H}}$. Clearly, $[\Delta_1]_{\mathcal{H}} = [\Delta_2]_{\mathcal{H}}$ iff $\Delta_1 \vdash_{\mathcal{H}} \tau$ for all $\tau \in \Delta_2$ and $\Delta_2 \vdash_{\mathcal{H}} \sigma$ for all $\sigma \in \Delta_1$. For brevity, we say Δ axiomatizes σ if $[\Delta]_{\mathcal{H}} = [\{\sigma\}]_{\mathcal{H}}$.

▶ **Example 10.** For $\Delta = \{a \to b \to a, (a \to b \to c) \to (a \to b) \to a \to c\}$ the set of formulae $[\Delta]_{\mathcal{H}}$ contains exactly the intuitionistic propositional implicational tautologies.

For $\Delta' = \Delta \cup \{((p \to q) \to p) \to p\}$ the set of formulae $[\Delta']_{\mathcal{H}}$ contains exactly the classical propositional implicational tautologies [20].

▶ **Definition 11.** An axiom σ is *principal* if there exists a λ-term M in β-normal form such that σ is the principal type of M in the simply typed λ-calculus.

Intuitively, axioms that are not principal, e.g. $a \to a \to a$, could in some contexts (for example, in synthesis) be considered 'artificial' since they have no 'naturally' associated realization. Principality of axioms is decidable [7] and, in fact, PSPACE-complete [9].

2.4 Post Correspondence Problem

The Post correspondence problem (PCP) is well-known for its undecidability ([14]).

▶ **Problem 12** (Post Correspondence Problem). *Given pairs of words $(v_1, w_1), \dots, (v_k, w_k)$ over the alphabet $\{\mathtt{a}, \mathtt{b}\}$ such that $v_i \neq \epsilon \neq w_i$ for $i = 1 \dots k$ (where ϵ is the empty word) decide whether there exists an index sequence $i_1, \dots, i_n$ such that $v_{i_1} v_{i_2} \dots v_{i_n} = w_{i_1} w_{i_2} \dots w_{i_n}$.*

As an immediate consequence, the following slight restriction of the problem, where $v_i \neq w_i$, is also undecidable.

▶ **Corollary 13.** *Given pairs of words* $(v_1, w_1), \ldots, (v_k, w_k)$ *over the alphabet* $\{\mathtt{a}, \mathtt{b}\}$ *such that* $\epsilon \neq v_i \neq w_i \neq \epsilon$ *for* $i = 1 \ldots k$ *it is undecidable whether there exists an index sequence* $i_1, \ldots, i_n$ *such that* $v_1 v_{i_1} v_{i_2} \ldots v_{i_n} = w_1 w_{i_1} w_{i_2} \ldots w_{i_n}$*.*

We aim at showing undecidability of recognizing principal axiomatizations of the calculus $a \to b \to a$ by reduction from PCP. Usually, the Post correspondence problem is approached constructively, i.e. start with some given pair of words, then iteratively append corresponding suffixes, and finally test for equality. The approach taken in the present work is, in a sense, 'deconstructive'. In particular, we start from an arbitrary pair of equal words, then iteratively remove corresponding suffixes, and finally test whether the resulting pair is given. While the former approach requires an equality test for arbitrarily large structures as a final operation (the encoding of which appears problematic in terms of principal axioms), the final operation of the latter approach can be bounded. The following Definition 14 and Lemma 15 capture the outlined iterative deconstruction.

▶ **Definition 14.** Given a set $\mathsf{PCP} = \{(v_1, w_1), \ldots, (v_k, w_k)\}$ of pairs of words over the alphabet $\{\mathtt{a}, \mathtt{b}\}$ we define for $n \geq 0$ the set PCP_n of pairs of words as follows

$$\mathsf{PCP}_0 = \{(v, v) \mid v \in \{\mathtt{a}, \mathtt{b}\}^*\} \qquad \mathsf{PCP}_{n+1} = \{(v, w) \mid \exists i \in \{1, \ldots, k\}.(v v_i, w w_i) \in \mathsf{PCP}_n\}$$

▶ **Lemma 15.** *Let* $n \geq 0$ *and* $v, w \in \{\mathtt{a}, \mathtt{b}\}^*$*. We have* $v v_{i_1} v_{i_2} \ldots v_{i_n} = w w_{i_1} w_{i_2} \ldots w_{i_n}$ *for some index sequence* $i_1, \ldots, i_n$ *iff* $(v, w) \in \mathsf{PCP}_n$*.*

Proof. Routine induction on n (cf. `pcp.pcp_set_iff_sync_word_pair`). ◀

In sum, it is undecidable whether the prefix (v_1, w_1) is in PCP_n for some $n \geq 0$.

▶ **Lemma 16.** *Given a set* $\mathsf{PCP} = \{(v_1, w_1), \ldots, (v_k, w_k)\}$ *of pairs of words over the alphabet* $\{\mathtt{a}, \mathtt{b}\}$ *such that* $\epsilon \neq v_i \neq w_i \neq \epsilon$ *for* $i = 1 \ldots k$ *it is undecidable whether there exists an* $n \geq 0$ *such that* $(v_1, w_1) \in \mathsf{PCP}_n$*.*

Proof. Immediate consequence of Corollary 13 and Lemma 15. ◀

3 Recognizing Axiomatizations of $a \to b \to a$

In this section we show that recognizing principal axiomatizations of the Hilbert-style calculus containing only the axiom $a \to b \to a$ is undecidable (cf. Theorem 17), which is our main result.

▶ **Theorem 17.** *Given principal axioms* $\sigma_1, \ldots, \sigma_n$ *such that* $\{a \to b \to a\} \vdash_{\mathcal{H}} \sigma_i$ *for* $i = 1 \ldots n$*, it is undecidable whether* $\{\sigma_1, \ldots, \sigma_n\} \vdash_{\mathcal{H}} a \to b \to a$*.*

▶ **Corollary 18.** *Given* λ*-terms* $M_1, \ldots, M_n$ *in* β*-normal form with principal types* $\sigma_1, \ldots, \sigma_n$ *in the simply typed* λ*-calculus such that* $\{a \to b \to a\} \vdash_{\mathcal{H}} \sigma_i$ *for* $i = 1 \ldots n$*, it is undecidable whether there is an applicative composition of* $M_1, \ldots, M_n$ *having the simple type* $a \to b \to a$*.*

In the context of type-based composition synthesis, the types $\sigma_1, \ldots, \sigma_n$ are natural specifications of associated terms $M_1, \ldots, M_n$ and $a \to b \to a$ is a goal specification. Deriving $a \to b \to a$ from $\sigma_1, \ldots, \sigma_n$ naturally corresponds to finding a composition of given terms satisfying the goal specification.

We prove Theorem 17 by reduction from the Post correspondence problem, specifically, the construction in Lemma 16. For that reason, we fix pairs $(v_1, w_1), \ldots, (v_k, w_k)$ of words over the alphabet $\{\mathtt{a}, \mathtt{b}\}$ such that $\epsilon \neq v_i \neq w_i \neq \epsilon$ for $i = 1 \ldots k$. Our goal is to construct principal axioms $\sigma_1, \ldots, \sigma_l$ such that $\{a \to b \to a\} \vdash_{\mathcal{H}} \sigma_i$ for $i = 1 \ldots l$ and $\{\sigma_1, \ldots, \sigma_l\} \vdash_{\mathcal{H}} a \to b \to a$ is equivalent to $(v_1, w_1) \in \mathsf{PCP}_n$ for some $n \geq 0$.

PCP Reduction

In this subsection we construct axioms $\sigma_1, \ldots, \sigma_l$ associated with the outlined reduction. We do not address principality which will be dealt with in the next subsection.

We need to represent words, pairs and suffixing. Let us fix a unique type atom $\bullet$. For a word $v \in \{\mathtt{a}, \mathtt{b}\}^*$ we define its representation as $[v] = \bullet \cdot v$ where the operation $\cdot$ is defined as

$$\sigma \cdot \epsilon = \sigma \qquad \sigma \cdot w\mathtt{a} = (\bullet \to \bullet) \to (\sigma \cdot w) \qquad \sigma \cdot w\mathtt{b} = (\bullet \to \bullet \to \bullet) \to (\sigma \cdot w)$$

We represent a pair of types σ, τ as

$$\langle \sigma, \tau \rangle = (\bullet \to \bullet \to \bullet) \to (\sigma \to \tau \to \bullet) \to (\bullet \to \sigma) \to (\bullet \to \tau) \to \bullet \to \bullet \to \bullet$$

As a side note, $[v]$ contains only $\bullet$ as atom. More importantly, we have $[v] \cdot w = [vw]$, and representations of two distinct words are not unifiable (cf. Lemma 19).

▶ **Lemma 19.** *Let $v, w \in \{\mathtt{a}, \mathtt{b}\}^*$. If $[v]$ and $[w]$ are unifiable, then $v = w$.*

Proof. Assuming $v \neq w$ we show that $[v]$ and $[w]$ are not unifiable by induction on the length of v (cf. `word.append_unique_encoding`). Wlog. v is not longer than w.

Case $v = \epsilon$: Clearly, $[v] = \bullet$ is not unifiable with $[w] = \sigma_1 \to \ldots \to \sigma_n \to \bullet$ with $n \geq 1$.

Case $v = v'\mathtt{a}$, $w = w'\mathtt{b}$ (resp. $v = v'\mathtt{b}$, $w = w'\mathtt{a}$): Let $\sigma = \bullet \to \bullet$ (resp. $\sigma = \bullet \to \bullet \to \bullet$) and $\tau = \bullet \to \bullet \to \bullet$ (resp. $\tau = \bullet \to \bullet$). Since σ is not unifiable with τ, we have that $[v] = \sigma \to [v']$ is not unifiable with $[w] = \tau \to [w']$.

Case $v = v'\mathtt{a}$, $w = w'\mathtt{a}$ (resp. $v = v'\mathtt{b}$, $w = w'\mathtt{b}$): By induction hypothesis $[v']$ is not unifiable with $[w']$. Therefore, $[v] = \sigma \to [v']$ is not unifiable with $[w] = \sigma \to [w']$, where $\sigma = \bullet \to \bullet$ (resp. $\sigma = \bullet \to \bullet \to \bullet$). ◀

Additionally, for any types σ, τ we have that $\langle \sigma, \tau \rangle$ is derivable from $a \to b \to a$ (cf. Lemma 21).

▶ **Lemma 20.** *Let σ, τ be types. If $\{a \to b \to a\} \vdash_{\mathcal{H}} \tau$, then $\{a \to b \to a\} \vdash_{\mathcal{H}} \sigma \to \tau$.*

Proof. Use ($\to$E) with the premises $\{a \to b \to a\} \vdash_{\mathcal{H}} \tau \to \sigma \to \tau$ and $\{a \to b \to a\} \vdash_{\mathcal{H}} \tau$. ◀

▶ **Lemma 21.** *Let σ, τ be types. We have $\{a \to b \to a\} \vdash_{\mathcal{H}} \langle \sigma, \tau \rangle$.*

Proof. Iterative application of Lemma 20 starting with $\{a \to b \to a\} \vdash_{\mathcal{H}} \bullet \to \bullet \to \bullet$. ◀

Finally, we define a type environment Γ of $k + 2$ combinators typed by principal axioms

$$\Gamma = \{x : \langle a, a \rangle, z : \langle [v_1], [w_1] \rangle \to \bullet \to a \to \bullet\} \cup \{y_i : \langle a \cdot v_i, b \cdot w_i \rangle \to \langle a, b \rangle \mid 1 \leq i \leq k\}$$

Due to Lemma 21, each axiom in $|\Gamma|$ is derivable from $a \to b \to a$.

Having established all prerequisite definitions, we now proceed with our main reduction. The following Lemma 22 establishes a connection between elements $(v, w) \in \mathsf{PCP}_n$ and inhabitants of $\langle [v], [w] \rangle$.

▶ **Lemma 22.** *Let ζ be a substitution and let $v, w \in \{\mathtt{a}, \mathtt{b}\}^*$. If $\Gamma \vdash_{\mathcal{C}} x \triangleright y_{i_1} \triangleright y_{i_2} \triangleright \ldots \triangleright y_{i_n} : \zeta(\langle [v], [w] \rangle)$ for some index sequence $i_1 \ldots i_n$, then $(v, w) \in \mathsf{PCP}_n$.*

Proof. Induction on n (cf. `pcp_reduction.piped_term_to_pcp_set`).
Basis Step: $\Gamma \vdash_{\mathcal{C}} x : \zeta(\langle [v], [w] \rangle)$ implies $\zeta([v]) = \zeta([w])$. By Lemma 19 we obtain $v = w$.
Inductive Step: Assume $\Gamma \vdash_{\mathcal{C}} x \triangleright y_{i_1} \triangleright y_{i_2} \triangleright \ldots \triangleright y_{i_n} \triangleright y_l : \zeta(\langle [v], [w] \rangle)$ for some index sequence $i_1, \ldots, i_n, l$. We necessarily have $\Gamma \vdash_{\mathcal{C}} y_l : \sigma \to \zeta(\langle [v], [w] \rangle)$ and $\Gamma \vdash_{\mathcal{C}} x \triangleright y_{i_1} \triangleright y_{i_2} \triangleright \ldots \triangleright y_{i_n} : \sigma$ for some type σ. Additionally, $\sigma \to \zeta(\langle [v], [w] \rangle) = \xi(\langle a \cdot v_l, b \cdot w_l \rangle \to \langle a, b \rangle)$ for some substitution ξ, which implies $\zeta([v]) = \xi(a)$, $\zeta([w]) = \xi(b)$ and $\zeta(\bullet) = \xi(\bullet)$. Therefore, $\xi(a \cdot v_l) = \zeta([v v_l])$ and $\xi(b \cdot w_l) = \zeta([w w_l])$. As a result, we have $\sigma = \xi(\langle a \cdot v_l, b \cdot w_l \rangle) = \zeta(\langle [v v_l], [w w_l] \rangle)$.
By induction hypothesis $(v v_l, w w_l) \in \mathsf{PCP}_n$, which implies $(v, w) \in \mathsf{PCP}_{n+1}$. ◀

Let us define $\mathfrak{n} \in \mathbb{N} \cup \{\infty\}$ (cf. `pcp_reduction.min_special`) as either the minimal size of a combinatory term typable in Γ by $\sigma \to \sigma \to \sigma$ or as ∞ if no such term exists.

$$\mathfrak{n} = \min\{\text{size}(F) \mid \Gamma \vdash_{\mathcal{C}} F : \sigma \to \sigma \to \sigma \text{ for some type } \sigma\}$$

Intuitively, a ‘small’, i.e. of size less than $\mathfrak{n}$, derivation of an instance of $\langle [v_1], [w_1] \rangle$ contains no derivation of an instance of $\bullet \to \bullet \to \bullet$. Due to our pair encoding, which contains as its first argument the type $\bullet \to \bullet \to \bullet$, we are able to severely restrict the shape of the minimal derivation of $\langle [v_1], [w_1] \rangle$ (cf. Lemma 23).

▶ **Lemma 23.** *If $\Gamma \vdash_{\mathcal{C}} F : \zeta(\langle \sigma, \tau \rangle)$ for some substitution ζ such that $\text{size}(F) < \mathfrak{n}$, then $F = x \triangleright y_{i_1} \triangleright \ldots \triangleright y_{i_m}$ for some (possibly empty) index sequence $i_1, \ldots, i_m$.*

Proof. Induction on $\text{size}(F)$ (cf. `pcp_reduction.small_to_piped_term`).
Basis Step: We have $F \neq z$ and $F \neq y_i$ for any i because the type of the corresponding combinator is not unifiable with $\langle \sigma, \tau \rangle$. If $F = x$ the claim follows.
Inductive Step:
Case $F = z\, G_1 \ldots G_m$ for some $m \geq 1$: We have $\Gamma \vdash_{\mathcal{C}} G_1 : \zeta(\langle [v_1], [w_1] \rangle)$ for some substitution ζ which using $a \mapsto \zeta(\bullet)$ implies $\Gamma \vdash_{\mathcal{C}} z\, G_1 : \zeta(\bullet) \to \zeta(\bullet) \to \zeta(\bullet)$. Therefore, $\mathfrak{n} \leq \text{size}(z\, G_1) \leq \text{size}(F) < \mathfrak{n}$. ↯
Case $F = x\, G_1 \ldots G_m$: We have $\Gamma \vdash_{\mathcal{C}} G_1 : \sigma' \to \sigma' \to \sigma'$ for some σ'. However, $\mathfrak{n} \leq \text{size}(G_1) < \text{size}(F) < \mathfrak{n}$ is a contradiction. ↯
Case $F = y_i\, G$ for some i: We have $\Gamma \vdash_{\mathcal{C}} G : \zeta(\langle \sigma \cdot v_i, \tau \cdot w_i \rangle)$. By induction hypothesis we have $G = x \triangleright y_{i_1} \triangleright \ldots \triangleright y_{i_m}$ for some (potentially empty) index sequence $i_1, \ldots, i_m$. Therefore, $F = x \triangleright y_{i_1} \triangleright \ldots \triangleright y_{i_m} \triangleright y_i$.
Case $F = y_i\, G_1 \ldots G_m$ for some i and some $m \geq 2$: We have $\Gamma \vdash_{\mathcal{C}} G_2 : \sigma' \to \sigma' \to \sigma'$ for some σ'. However, $\mathfrak{n} \leq \text{size}(G_2) < \text{size}(F) < \mathfrak{n}$ is a contradiction. ↯ ◀

Next, we show that if $a \to b \to a$ is derivable, then there is a small derivation of an instance of $\langle [v_1], w_1 \rangle$ (cf. Lemma 24).

▶ **Lemma 24.** *If $|\Gamma| \vdash_{\mathcal{H}} a \to b \to a$, then $\Gamma \vdash_{\mathcal{C}} F : \zeta(\langle [v_1], [w_1] \rangle)$ for some substitution ζ and combinatory term F such that $\text{size}(F) < \mathfrak{n}$.*

Proof. Case analysis (cf. `pcp_reduction.aba_to_small_v1w1`). $|\Gamma| \vdash_{\mathcal{H}} a \to b \to a$ implies $\mathfrak{n} < \infty$, i.e. $\Gamma \vdash_{\mathcal{C}} H : \sigma \to \sigma \to \sigma$ for some type σ and combinatory term H such that $\text{size}(H) = \mathfrak{n}$.

Cases $H = x$ or $H = z$ or $H = y_i$ or $H = y_i\, G$: H cannot be typed by $\sigma \to \sigma \to \sigma$. ↯
Case $H = x\, G_1 \ldots G_m$ for some $m \geq 1$: We have $\Gamma \vdash_{\mathcal{C}} G_1 : \sigma' \to \sigma' \to \sigma'$ for some σ'. However, $\mathfrak{n} \leq \text{size}(G_1) < \text{size}(H) = n$ is a contradiction. ↯

Case $H = y_i\, G_1 \dots G_m$ for some $m \geq 2$: We have $\Gamma \vdash_{\mathcal{C}} G_2 : \sigma' \to \sigma' \to \sigma'$ for some σ'. Again, $\mathfrak{n} \leq \text{size}(G_2) < \text{size}(H) = n$ is a contradiction. ↯

Case $H = z\, G_1 \dots G_m$ for some $m \geq 1$: We have $\Gamma \vdash_{\mathcal{C}} G_1 : \zeta(\langle [v_1], [w_1] \rangle)$ for some substitution ζ which proves the claim since $\text{size}(G_1) < \text{size}(H) = \mathfrak{n}$. ◀

By construction, elements $(v, w) \in \mathsf{PCP}_n$ are associated with terms of type $\langle [v], [w] \rangle$ (cf. Lemma 25).

▶ **Lemma 25.** *Let $v, w \in \{\mathtt{a}, \mathtt{b}\}^*$. If $(v, w) \in \mathsf{PCP}_n$, then $\Gamma \vdash_{\mathcal{C}} x \triangleright y_{i_1} \triangleright y_{i_2} \triangleright \dots \triangleright y_{i_n} : \langle [v], [w] \rangle$ for some index sequence $i_1 \dots i_n$.*

Proof. Routine induction on n (cf. `pcp_reduction.pcp_set_to_piped_term`).

Basis Step: $(v, w) \in \mathsf{PCP}_0$ implies $v = w$. Using the substitution $a \mapsto [v] = [w]$ we obtain $\Gamma \vdash_{\mathcal{C}} x : \langle [v], [w] \rangle$.

Inductive Step: $(v, w) \in \mathsf{PCP}_{n+1}$ implies $(vv_l, ww_l) \in \mathsf{PCP}_n$ for some $l \in \{1, \dots k\}$. By induction hypothesis $\Gamma \vdash_{\mathcal{C}} x \triangleright y_{i_1} \triangleright y_{i_2} \triangleright \dots \triangleright y_{i_n} : \langle [vv_l], [ww_l] \rangle$ for some index sequence $i_1 \dots i_n$. Using the substitution $a \mapsto [v], b \mapsto [w]$ we have $\Gamma \vdash_{\mathcal{C}} y_l : \langle [vv_l], [ww_l] \rangle \to \langle [v], [w] \rangle$. By ($\to$E), we obtain the claim $\Gamma \vdash_{\mathcal{C}} x \triangleright y_{i_1} \triangleright y_{i_2} \triangleright \dots \triangleright y_{i_n} \triangleright y_l : \langle [v], [w] \rangle$. ◀

Finally, we can prove the following key Lemma 26 (cf. `pcp_reduction.aba_iff_pcp_set`) which relates membership of (v_1, w_1) in some PCP_n and derivability of $a \to b \to a$ from $|\Gamma|$.

▶ **Lemma 26.** *We have $|\Gamma| \vdash_{\mathcal{H}} a \to b \to a$ iff $(v_1, w_1) \in \mathsf{PCP}_n$ for some $n \geq 0$.*

Proof. $\Longrightarrow$: Assume $|\Gamma| \vdash_{\mathcal{H}} a \to b \to a$. By Lemma 24 we have $\Gamma \vdash_{\mathcal{C}} F : \zeta(\langle [v_1], [w_1] \rangle)$ for some substitution ζ and combinatory term F with $\text{size}(F) < \mathfrak{n}$. By Lemma 23 we have $F = x \triangleright y_{i_1} \triangleright y_{i_2} \triangleright \dots \triangleright y_{i_n}$ for some index sequence $i_1 \dots i_n$. Finally, by Lemma 22 we have $(v_1, w_1) \in \mathsf{PCP}_n$.

$\Longleftarrow$: Assume $(v_1, w_1) \in \mathsf{PCP}_n$. By Lemma 25 we have $\Gamma \vdash_{\mathcal{C}} F : \langle [v_1], [w_1] \rangle$ for some term F. Using an appropriate substitution, we obtain $\Gamma \vdash_{\mathcal{C}} z\, F : a \to b \to a$. ◀

Principality of Axioms $|\Gamma|$

In this subsection we inspect evidence that axioms $|\Gamma|$ are, in fact, principal by means of examples. Moreover, we explore the correspondence between compositions of principal inhabitants and solutions to the given PCP instance. Particularly, a composition of principal inhabitants which shows at type level that the given PCP instance has a solution also constructs and verifies the corresponding solution at term level.

Let us fix the following PCP instance

k	$(v_1,\ \ w_1)$	$(v_2,\ \ w_2)$	$(v_3,\ \ w_3)$	$v_1\ v_2\ v_1\ v_3 = w_1\ w_2\ w_1\ w_3$
3	(ba, b)	(ab, aa)	(a, baa)	ba ab ba a = b aa b baa

In this case, $|\Gamma|$ consists of the following five axioms

$$\begin{aligned}
\sigma^x &= \langle a, a\rangle \\
&= (\bullet \to \bullet \to \bullet) \to (a \to a \to \bullet) \to (\bullet \to a) \to (\bullet \to a) \to \bullet \to \bullet \to \bullet \\
\sigma^z &= \langle [v_1], [w_1]\rangle \to \bullet \to a \to \bullet \\
&= \langle (\bullet \to \bullet) \to (\bullet \to \bullet \to \bullet) \to \bullet, (\bullet \to \bullet \to \bullet) \to \bullet\rangle \to \bullet \to a \to \bullet \\
\sigma^y_1 &= \langle a \cdot v_1, b \cdot w_1\rangle \to \langle a, b\rangle \\
&= \langle (\bullet \to \bullet) \to (\bullet \to \bullet \to \bullet) \to a, (\bullet \to \bullet \to \bullet) \to b\rangle \to \langle a, b\rangle \\
\sigma^y_2 &= \langle a \cdot v_2, b \cdot w_2\rangle \to \langle a, b\rangle \\
&= \langle (\bullet \to \bullet \to \bullet) \to (\bullet \to \bullet) \to a, (\bullet \to \bullet) \to (\bullet \to \bullet) \to b\rangle \to \langle a, b\rangle \\
\sigma^y_3 &= \langle a \cdot v_3, b \cdot w_3\rangle \to \langle a, b\rangle \\
&= \langle (\bullet \to \bullet) \to a, (\bullet \to \bullet) \to (\bullet \to \bullet) \to (\bullet \to \bullet \to \bullet) \to b\rangle \to \langle a, b\rangle
\end{aligned}$$

We utilize Haskell[3] to implement principal inhabitants of the above axioms. Further, we use GHC's built-in type inference to attest principality[4]. In the following code fragments '`\f -> M`' represents the λ-term '$\lambda f.M$' and '`--comment`' marks a comment.

The following λ-term $\mathrm{pair}_{\mathrm{eq}}$ is a principal inhabitant of σ^x and is implemented in the subsequent Listing 1 together with its inferred principal type.

$$\begin{aligned}
\mathrm{pair}_{\mathrm{eq}} = &\lambda f.\lambda \mathrm{is}_{vw}.\lambda \mathrm{make}_v.\lambda \mathrm{make}_w.\lambda x.\lambda y. \\
&f\ x\ (f\ (\mathrm{is}_{vw}\ (\mathrm{make}_v\ x)\ (\mathrm{make}_w\ x))\ (\mathrm{is}_{vw}\ (\mathrm{make}_w\ y)\ (\mathrm{make}_v\ y)))
\end{aligned}$$

Listing 1 Principal Inhabitant of σ^x.

```
pair_eq = \f -> \is_vw -> \make_v -> \make_w -> \x -> \y ->
    f x (f (is_vw (make_v x) (make_w x)) (is_vw (make_w y) (make_v y)))

--inferred principal type
pair_eq
  :: (t1 -> t1 -> t1)
     -> (t2 -> t2 -> t1) -> (t1 -> t2) -> (t1 -> t2) -> t1 -> t1 -> t1
```

Intuitively, f is used as conjunction and is_{vw}, make_v, make_w are used to attest equality of v and w. Following this intuition, the type $\bullet$ is inhabited by tautologies (e.g. the Haskell value `True`), which leads to a principal inhabitant of σ^z implemented in the following Listing 2.

Listing 2 Principal Inhabitant of σ^z.

```
check_ba_b = \pair -> \x -> \z ->
    let f = pair (\x1 -> \x2 -> x) (\v -> \w -> x)
            (\x' -> \v1c2 -> \v1c1 -> x) (\x' -> \w1c1 -> x)
    in pair
        (\x1 -> \x2 -> f x1 x2) --and
        (\v -> \w -> f (v (f x) f) (w f)) --is_vw
        (\x' -> \v1c2 -> \v1c1 -> f x' (f (v1c2 x) (v1c1 x x))) --make_v
        (\x' -> \w1c1 -> f x' (w1c1 x x)) --make_w
        x --True
        x --True
```

[3] `https://www.haskell.org/`
[4] Although Haskell's type system is more expressive than simple types, we argue that the additional features play no role in the given examples.

```
check_ba_b --inferred principal type
  :: ((p1 -> p1 -> p1)
      -> (((p1 -> p1) -> (p1 -> p1 -> p1) -> p1)
          -> ((p1 -> p1 -> p1) -> p1) -> p1)
      -> (p1 -> (p1 -> p1) -> (p1 -> p1 -> p1) -> p1)
      -> (p1 -> (p1 -> p1 -> p1) -> p1)
      -> p1 -> p1 -> p1) -> p1 -> p2 -> p1
```

Note that the `let ... in` construction in this case does not contribute to principality (replacing all occurrences of `f` by copies of its implementation does not change the principal type) and is just syntactic sugar. In the above, `f` is used not only as conjunction but also to construct inhabitants `f x` of type $\bullet \to \bullet$ (represents `a`) and `f` of type $\bullet \to \bullet \to \bullet$ (represents `b`). As a result, `v (f x) f` is of type $\bullet$ iff `v` is of type [ba]. Since `f` allows for arbitrary nesting, we argue that the above principal inhabitant construction can be generalized for any pair of words v_1, w_1.

Finally, we implement principal inhabitants of $\sigma_1^y, \sigma_2^y, \sigma_3^y$ in the following Listing 3.

Listing 3 Principal Inhabitants of $\sigma_1^y, \sigma_2^y, \sigma_3^y$.

```
step_ba_b = \pair -> \f -> \is_vw -> \make_v -> \make_w -> \x -> \y ->
    f (f x y) (f (is_vw (make_v x) (make_w y))
        (pair (\x1 -> \x2 -> f x1 x2)
            (\v' -> \w' -> is_vw (v' (f x) f) (w' f)) --is_v'w'
            (\x' -> \v'c2 -> \v'c1 ->
                make_v (f x' (f (v'c2 x) (v'c1 x x)))) --v'
            (\x' -> \w'c1 ->
                make_w (f x' (w'c1 x x))) --w'
            x y))

step_ba_b --inferred principal type
  :: ((t1 -> t1 -> t1)
      -> (((t1 -> t1) -> (t1 -> t1 -> t1) -> t2)
          -> ((t1 -> t1 -> t1) -> t3) -> t1)
      -> (t1 -> (t1 -> t1) -> (t1 -> t1 -> t1) -> t2)
      -> (t1 -> (t1 -> t1 -> t1) -> t3)
      -> t1 -> t1 -> t1)
     -> (t1 -> t1 -> t1) -> (t2 -> t3 -> t1) -> (t1 -> t2) -> (t1 -> t3)
     -> t1 -> t1 -> t1

step_ab_aa = \pair -> \f -> \is_vw -> \make_v -> \make_w -> \x -> \y ->
    f (f x y) (f (is_vw (make_v x) (make_w y))
        (pair (\x1 -> \x2 -> f x1 x2)
            (\v' -> \w' -> is_vw (v' f (f x)) (w' (f x) (f x)))
            (\x' -> \v'c2 -> \v'c1 ->
                make_v (f x' (f (v'c2 x x) (v'c1 x))))
            (\x' -> \w'c2 -> \w'c1 ->
                make_w (f x' (f (w'c2 x) (w'c1 x))))
            x y))

step_a_baa = \pair -> \f -> \is_vw -> \make_v -> \make_w -> \x -> \y ->
    f (f x y) (f (is_vw (make_v x) (make_w y))
        (pair (\x1 -> \x2 -> f x1 x2)
            (\v' -> \w' -> is_vw (v' (f x)) (w' (f x) (f x) f))
            (\x' -> \v'c1 ->
                make_v (f x' (v'c1 x)))
            (\x' -> \w'c3 -> \w'c2 -> \w'c1 ->
                make_w (f x' (f (w'c3 x) (f (w'c2 x) (w'c1 x x)))))
            x y))
```

Similarly, to the principal inhabitant of σ^z, we need to ensure that the principal type of the first argument `pair` is suffixed by the given words. Now, we may (and have to) use the additional arguments `is_vw` of type $a \to b \to \bullet$ and `make_v`, `make_w` of types $\bullet \to a$ and $\bullet \to b$.

Using `f` as conjunction and a way to construct inhabitants of character representations, `\v' -> \w' -> is_vw (v' (f x) f) (w' f)` implements the principal inhabitant of $(a \cdot \mathtt{ba}) \to (b \cdot \mathtt{b}) \to \bullet$. Having `x` of type $\bullet$, `\x' -> \v'c2 -> \v'c1 -> make_v (f x' (f (v'c2 x) (v'c1 x x)))` implements the principal inhabitant of $\bullet \to (a \cdot \mathtt{ba})$. Again, we argue that the principal inhabitant construction can be generalized inductively.

Knowing a solution to the above PCP instance, let us compose the given implementations in the following Listing 4 to an implementation of $a \to b \to a$.

Listing 4 Inhabitant of $a \to b \to a$.

```
(|>) x y = y x
k = pair_eq |> step_a_baa |> step_ba_b |> step_ab_aa |> check_ba_b
k :: p1 -> p2 -> p1 --inferred principal type
```

As expected, evaluating '`k "Lazy␣Hello" undefined`' via GHC results in `"Lazy␣Hello"`, which shows that the second argument `undefined` is not evaluated. For some higher-order pleasure, the expression '`k ((.)$(.)) "Ask␣the␣owl" (==) 2 (1+) 1`' evaluates (as expected) to `True`.

Interestingly, the above principal implementation of the given axioms has more computational meaning than just preserving an argument. Viewing `True` as the only inhabitant of $\bullet$, consider the following piece of code.

Listing 5 Composition of Inhabitants.

```
a = \x -> x
b = \x -> \y -> x && y

--c == a
is_a = \c -> (c True == a True) && (c False == a False)
--c == b
is_b = \c -> (c True True == b True True)
        && (c True False == b True False)
        && (c False True == b False True)
        && (c False False == b False False)

make_ba = \x -> \c2 -> \c1 -> x && (is_a c2) && (is_b c1)
make_b = \x -> \c1 -> x && (is_b c1)

is_ba_b = \v -> \w -> (v a b) && (w b)

composition = pair_eq |> step_a_baa |> step_ba_b |> step_ab_aa
f = composition (&&) is_ba_b make_ba make_b

show_function_table = do
    putStrLn ("f␣True␣␣True␣␣=␣" ++ (show (f True True)))
    putStrLn ("f␣True␣␣False␣=␣" ++ (show (f True False)))
    putStrLn ("f␣False␣True␣␣=␣" ++ (show (f False True)))
    putStrLn ("f␣False␣False␣=␣" ++ (show (f False False)))
```

The terms `a` and `b` implement representations of characters a and b. Given a character `c`, the function `is_a` (resp. `is_b`) extensionally verifies equality to `a` (resp. `b`). The functions `make_ba`, `make_b` and `is_ba_a` construct and verify inhabitants of representations of the corresponding words.

Using the solution to the given PCP instance we have that `composition` is of type

$$\langle [ba], [b] \rangle = (\bullet \to \bullet \to \bullet) \to ([ba] \to [b] \to \bullet) \to (\bullet \to [ba]) \to (\bullet \to [b]) \to \bullet \to \bullet \to \bullet$$

and, therefore, `f` is of type $\bullet \to \bullet \to \bullet$. Displaying the function table of `f` reveals that if conjunction (`&&`) is the first argument of `composition`, then `f` remains a conjunction. On closer examination of the underlying implementation, the term `f True True` evaluates to `True` by constructing the representation of the actual solution to the given PCP instance and verifying the correctness of the constructed solution via `pair_eq`. Viewing $|\Gamma|$ as an intuitionistic axiomatization of the corresponding PCP instance, this provides an additional twist to the Curry-Howard isomorphism. The constructed λ-term not only corresponds to the proof that a formula is derivable, but actually solves the underlying problem on the term level.

Derivability from $a \to b \to a$

We conclude this section by the systematic observation that the condition $\{a \to b \to a\} \vdash_{\mathcal{H}} \sigma_i$ for $1 = 1 \dots n$ is decidable. The key observation is that any formula derivable from $a \to b \to a$ is a prefixed instance of $a \to b \to a$ (cf. Lemma 27).

▶ **Lemma 27.**
$[\{a \to b \to a\}]_{\mathcal{H}} = \{\sigma_1 \to \dots \to \sigma_n \to \sigma_{n+1} \to \tau \to \sigma_{n+1} \mid n \geq 0 \textit{ and } \sigma_1, \dots, \sigma_{n+1}, \tau \in \mathbb{T}\}$

Proof. **$\supseteq$:** n-fold use of Lemma 20 starting with $\{a \to b \to a\} \vdash_{\mathcal{H}} \sigma_{n+1} \to \tau \to \sigma_{n+1}$.
$\subseteq$: Let $\Gamma = \{x : a \to b \to a\}$. Assume $\Gamma \vdash_{\mathcal{C}} F : \sigma$ such that size(F) is minimal. We show the claim by induction on size(F).
Case $F = x$: We have $\sigma = \zeta(a \to b \to a)$ for some substitution ζ, showing the claim.
Case $F = x\ G$: We have $\sigma = \zeta(b \to a)$ and $\Gamma \vdash_{\mathcal{C}} G : \zeta(a)$ for some substitution ζ. By the induction hypothesis $\zeta(a) = \sigma_1 \to \dots \to \sigma_n \to \sigma_{n+1} \to \tau \to \sigma_{n+1}$ for some $n \geq 0$ and $\sigma_1, \dots, \sigma_{n+1}, \tau \in \mathbb{T}$. Therefore, $\sigma = \zeta(b \to a) = \zeta(b) \to \sigma_1 \to \dots \to \sigma_n \to \sigma_{n+1} \to \tau \to \sigma_{n+1}$, which shows the claim.
Case $F = x\ G_1 \dots G_l$ for some $l \geq 2$: We have $\Gamma \vdash_{\mathcal{C}} x : \tau_1 \to \dots \to \tau_l \to \sigma = \zeta(a \to b \to a)$ for some substitution ζ, therefore $\tau_1 = \tau_3 \to \dots \to \tau_l \to \sigma$. Additionally, we have $\Gamma \vdash_{\mathcal{C}} G_i : \tau_i$ for $i = 1 \dots l$. As a result, $\Gamma \vdash_{\mathcal{C}} G_1\ G_3 \dots G_l : \sigma$ with size$(G_1\ G_3 \dots G_l) <$ size(F) which contradicts minimality of size(F). ◀

As a result of the above syntactic characterization by Lemma 27 of formulae derivable from $a \to b \to a$, it is decidable in linear time whether for a given formula σ we have $\sigma \in [\{a \to b \to a\}]_{\mathcal{H}}$. This contrasts PSPACE-completeness to decide $\sigma \in [\{a \to b \to a, (a \to b \to c) \to (a \to b) \to (a \to c)\}]_{\mathcal{H}}$ [18].

4 Recognizing Axiomatizations of $a \to a$

In this section, we record that axiomatizations of the Hilbert-style calculus containing only the axiom $a \to a$ are recognizable in linear time. The key observation is that you cannot meaningfully compose axioms that are instances of $a \to a$. Therefore, the only derivable formulae are instances of the given axioms (cf. Lemma 28).

▶ **Lemma 28.** *Given $\sigma_1, \dots, \sigma_n \in \mathbb{T}$ we have*

$$[\{\sigma_1 \to \sigma_1, \dots, \sigma_n \to \sigma_n\}]_{\mathcal{H}} = \bigcup_{i=1}^{n} \{\zeta(\sigma_i \to \sigma_i) \mid \zeta \textit{ is a substitution}\}$$

Proof. **$\supseteq$:** holds by instantiation of $\sigma_i \to \sigma_i$ for $i = 1 \dots n$.
$\subseteq$: Let $\Gamma = \{x_1 : \sigma_1 \to \sigma_1, \dots, x_n : \sigma_n \to \sigma_n\}$. Assume $\Gamma \vdash_{\mathcal{C}} F : \sigma$ such that size(F) is minimal.
Case $F = x_i$ for some $i \in \{1, \dots, n\}$: We have $\sigma = \zeta(\sigma_i \to \sigma_i)$ for some substitution ζ, which shows the claim.

Case $F = x_i\ G_1 \dots G_l$ for some $i \in \{1, \dots, n\}$: We have $\Gamma \vdash_{\mathcal{C}} x_i : \tau_1 \to \dots \to \tau_l \to \sigma = \zeta(\sigma_i \to \sigma_i)$ for some substitution ζ, therefore $\tau_1 = \tau_2 \to \dots \to \tau_l \to \sigma$. Additionally, we have $\Gamma \vdash_{\mathcal{C}} G_i : \tau_i$ for $i = 1 \dots l$. As a result, $\Gamma \vdash_{\mathcal{C}} G_1\ G_2 \dots G_l : \sigma$ with $\text{size}(G_1\ G_2 \dots G_l) < \text{size}(F)$ which contradicts minimality of $\text{size}(F)$. ◀

As a side note, the above proof of Lemma 28 implies that the set of minimal proofs from axioms $\{\sigma_1 \to \sigma_1, \dots, \sigma_n \to \sigma_n\}$ is of finite cardinality n.

▶ **Corollary 29.** $[\{a \to a\}]_{\mathcal{H}} = \{\tau \to \tau \mid \tau \in \mathbb{T}\}$.

▶ **Lemma 30.** *If $[\Delta]_{\mathcal{H}} = [\{a \to a\}]_{\mathcal{H}}$, then $b \to b \in \Delta$ for some $b \in \mathbb{A}$.*

Proof. Since $[\{a \to a\}]_{\mathcal{H}} \supseteq [\Delta]_{\mathcal{H}}$ implies $[\{a \to a\}]_{\mathcal{H}} \supseteq \Delta$ we have $\Delta = \{\sigma_1 \to \sigma_1, \dots, \sigma_n \to \sigma_n\}$ for some $\sigma_1, \dots, \sigma_n \in \mathbb{T}$ by Corollary 29. By Lemma 28 we obtain $[\Delta]_{\mathcal{H}} = \bigcup_{i=1}^{n} \{\zeta(\sigma_i \to \sigma_i) \mid \zeta \text{ is a substitution}\}$. Due to $[\{a \to a\}]_{\mathcal{H}} \subseteq [\Delta]_{\mathcal{H}}$ we obtain $a \to a = \zeta(\sigma_i \to \sigma_i)$ for some $i \in \{1, \dots, n\}$ and some substitution ζ, which holds iff $\sigma_i \to \sigma_i = b \to b$ for some $b \in \mathbb{A}$. ◀

▶ **Corollary 31.** *We have $[\Delta]_{\mathcal{H}} = [\{a \to a\}]_{\mathcal{H}}$ iff $\Delta \subseteq \{\sigma \to \sigma \mid \sigma \in \mathbb{T}\}$ and $b \to b \in \Delta$ for some $b \in \mathbb{A}$.*

As a result of Corollary 31, recognizing axiomatizations of $a \to a$ is decidable in linear time.

5 Recognizing Axiomatizations of $a \to b \to b$

In this section, we extend linear time recognizability to axiomatizations of the Hilbert-style calculus containing only the axiom $a \to b \to b$. Similarly to $a \to a$, meaningful logical compositions of instances of $a \to b \to b$ are limited (cf. Lemma 32).

▶ **Lemma 32.** $[\{a \to b \to b\}]_{\mathcal{H}} = \{\sigma \to \tau \to \tau \mid \sigma, \tau \in \mathbb{T}\} \cup \{\tau \to \tau \mid \tau \in \mathbb{T}\}$

Proof. **$\supseteq$:** Instantiation resp. derivability of $a \to a$.
$\subseteq$: Let $\Gamma = \{x : a \to b \to b\}$. Assume $\Gamma \vdash_{\mathcal{C}} F : \sigma$ such that $\text{size}(F)$ is minimal.

Case $F = x$: We have $\sigma = \zeta(a \to b \to b)$ for some substitution ζ, which shows the claim.

Case $F = x\ G$: We have $\Gamma \vdash_{\mathcal{C}} x : \tau \to \sigma = \zeta(a \to b \to b)$ for some substitution ζ, therefore $\sigma = \sigma' \to \sigma'$, which shows the claim.

Case $F = x\ G_1 \dots G_l$ where $l \geq 2$: We have $\Gamma \vdash_{\mathcal{C}} x : \tau_1 \to \dots \to \tau_l \to \sigma = \zeta(a \to b \to b)$ for some substitution ζ, therefore $\tau_3 = \tau_3 \to \dots \to \tau_l \to \sigma$. Additionally, we have $\Gamma \vdash_{\mathcal{C}} G_i : \tau_i$ for $i = 1 \dots l$. As a result, $\Gamma \vdash_{\mathcal{C}} G_2\ G_3 \dots G_l : \sigma$ with $\text{size}(G_2\ G_3 \dots G_l) < \text{size}(F)$ which contradicts minimality of $\text{size}(F)$. ◀

As a side note, the above proof of Lemma 32 implies that the set of minimal proofs starting from the axiom $a \to b \to b$ is finite (in fact of cardinality 2).

Using the above observation, we can characterize axiomatizations of $a \to b \to b$ syntactically (cf. Lemma 33).

▶ **Lemma 33.** *We have $[\Delta]_{\mathcal{H}} = [\{a \to b \to b\}]_{\mathcal{H}}$ iff $c \to d \to d \in \Delta$ for some $c, d \in \mathbb{A}$ and $\Delta \subseteq \{\sigma \to \tau \to \tau \mid \sigma, \tau \in \mathbb{T}\} \cup \{\tau \to \tau \mid \tau \in \mathbb{T}\}$.*

Proof. **$\Longleftarrow$:** $\sigma \to \tau \to \tau$ and $\tau \to \tau$ are derivable from $c \to d \to d$ for any $\sigma, \tau \in \mathbb{T}$.

$\Longrightarrow$: Due to $[\{a \to b \to b\}]_{\mathcal{H}} \supseteq \Delta$ we have $\Delta \subseteq \{\sigma \to \tau \to \tau \mid \sigma, \tau \in \mathbb{T}\} \cup \{\tau \to \tau \mid \tau \in \mathbb{T}\}$ by Lemma 32. Due to $[\{a \to b \to b\}]_{\mathcal{H}} \subseteq [\Delta]_{\mathcal{H}}$ we obtain $\Delta \vdash_{\mathcal{H}} a \to b \to b$. By case analysis (similar to the proof of Lemma 32) the minimal derivation of $\Delta \vdash_{\mathcal{H}} a \to b \to b$ is an instantiation of some $\sigma \to \tau \to \tau \in \Delta$, i.e. $a \to b \to b = \zeta(\sigma \to \tau \to \tau)$ for some substitution ζ. Therefore, $\sigma \to \tau \to \tau = c \to d \to d$ for some $c, d \in \mathbb{A}$. ◀

As a result of the above Lemma 33, recognizing axiomatizations of $a \to b \to b$ is decidable in linear time. One reason why recognizing axiomatizations of $a \to a$ and $a \to b \to b$ is trivial is that the set of minimal proofs in the corresponding calculi is finite, which is not the case for $a \to b \to a$.

6 Conclusion

We have shown that even under two severe restrictions subintuitionistic propositional calculi have undecidable derivability. In particular, it is undecidable whether from a given set of axioms (all of which are principal and derivable from $a \to b \to a$) the formula $a \to b \to a$ is derivable. In contrast, with respect to the formula $a \to b \to b$ derivability is decidable in linear time. Our result sheds some light on the lower end of the spectrum of propositional calculi. In future, it may be of systematic interest to inspect (sets of) axioms that correspond to principal types of other well known combinators such as **S**.

Under the Curry-Howard isomorphism, our result is related to type-based composition synthesis. Particularly, even under the the assumption that the given building blocks are 'natural' (in the sense of principality) and 'plain' (in the sense of their types are derivable from the axiom $a \to b \to a$) synthesis remains undecidable. The research program in type-based composition synthesis outlined in [15] is based on bounded variants of the inhabitation problem in combinatory logic [8].

Additionally, the reduction from the Post correspondence problem proving of our main result is formalized in the Lean theorem prover.

References

1 H. P. Barendregt. *The Lambda Calculus. Its Syntax and Semantics.* Studies in Logic and the Foundations of Mathematics, 2nd Edition. Elsevier Science Publishers, 1984. `doi:10.2307/2274112`.

2 H. P. Barendregt, W. Dekkers, and R. Statman. *Lambda Calculus with Types.* Perspectives in Logic, Cambridge University Press, 2013.

3 Marcin Benke, Aleksy Schubert, and Daria Walukiewicz-Chrzaszcz. Synthesis of Functional Programs with Help of First-Order Intuitionistic Logic. In *FSCD 2016*, volume 52 of *LIPIcs*, pages 12:1–12:16, 2016. `doi:10.4230/LIPIcs.FSCD.2016.12`.

4 Grigoriy V. Bokov. Completeness problem in the propositional calculus. *Intelligent Systems*, 13(1-4):165–182, 2009.

5 Grigoriy V. Bokov. Undecidability of the problem of recognizing axiomatizations for propositional calculi with implication. *Logic Journal of the IGPL*, 23(2):341–353, 2015. `doi:10.1093/jigpal/jzu047`.

6 Grigoriy V. Bokov. Undecidable problems for propositional calculi with implication. *Logic Journal of the IGPL*, 24(5):792–806, 2016. `doi:10.1093/jigpal/jzw013`.

7 Sabine Broda and Luís Damas. On Long Normal Inhabitants of a Type. *J. Log. Comput.*, 15(3):353–390, 2005. `doi:10.1093/logcom/exi016`.

8 Boris Düdder, Moritz Martens, Jakob Rehof, and Paweł Urzyczyn. Bounded Combinatory Logic. In *CSL 2012, Proceedings of Computer Science Logic*, volume 16 of *LIPIcs*, pages 243–258. Schloss Dagstuhl, 2012.

9 Andrej Dudenhefner and Jakob Rehof. The Complexity of Principal Inhabitation. In *FSCD 2017*, pages 15:1–15:14, 2017. `doi:10.4230/LIPIcs.FSCD.2017.15`.

10 J. R. Hindley. *Basic Simple Type Theory.* Cambridge Tracts in Theoretical Computer Science, vol. 42, Cambridge University Press, 2008.

11 J. R. Hindley and J. P. Seldin. *Lambda-calculus and Combinators, an Introduction.* Cambridge University Press, 2008.

12 AV Kuznecov and E Mendelson. Undecidability of the general problems of completeness, decidability and equivalence for propositional calculi. *The Journal of Symbolic Logic*, 1972.

13 Samuel Linial and Emil L. Post. Recursive Unsolvability of the Deducibility, Tarski's Completeness and Independence of Axioms Problems of Propositional Calculus. *Bulletin of the American Mathematical Society*, 55:50, 1949.

14 Emil L. Post. A variant of a recursively unsolvable problem. *Bulletin of the American Mathematical Society*, 52(4):264–268, 1946.

15 Jakob Rehof. Towards Combinatory Logic Synthesis. In *BEAT'13, 1st International Workshop on Behavioural Types.* ACM, 2013.

16 W. E. Singletary. Many-one Degrees Associated with Partial Propositional Calculi. *Notre Dame Journal of Formal Logic*, XV(2):335–343, 1974.

17 M.H. Sørensen and P. Urzyczyn. *Lectures on the Curry-Howard Isomorphism*, volume 149 of *Studies in Logic and the Foundations of Mathematics.* Elsevier, 2006.

18 R. Statman. Intuitionistic Propositional Logic Is Polynomial-space Complete. *Theoretical Computer Science*, 9:67–72, 1979.

19 Mary Katherine Yntema. A detailed argument for the Post-Linial theorems. *Notre Dame Journal of Formal Logic*, 5(1):37–50, 1964. `doi:10.1305/ndjfl/1093957737`.

20 Evgeny Zolin. Undecidability of the Problem of Recognizing Axiomatizations of Superintuitionistic Propositional Calculi. *Studia Logica*, 102(5):1021–1039, 2014. `doi:10.1007/s11225-013-9520-5`.