Principality and Approximation under Dimensional Bound

ANDREJ DUDENHEFNER, TU Dortmund University, Germany
JAKOB REHOF, TU Dortmund University, Germany

We develop an algebraic and algorithmic theory of principality for the recently introduced framework of intersection type calculi with dimensional bound. The theory enables inference of principal type information under dimensional bound, it provides an algebraic and algorithmic theory of approximation of classical principal types in terms of computable bases of abstract vector spaces (more precisely, semimodules), and it shows a systematic connection of dimensional calculi to the theory of approximants. Finite, computable bases are shown to span standard principal typings of a given term for sufficiently high dimension, thereby providing an approximation to standard principality by type inference, and capturing it precisely for sufficiently large dimensional parameter. Subsidiary results include decidability of principal inhabitation for intersection types (given a type does there exist a normal form for which the type is principal?). Remarkably, combining bounded type inference with principal inhabitation allows us to compute approximate normal forms of arbitrary terms without using beta-reduction.

CCS Concepts: · Theory of computation → Type theory;

Additional Key Words and Phrases: Lambda Calculus, Type Theory, Approximation, Intersection Types, Inhabitation, Type Inference, Principal Type

ACM Reference Format:
Andrej Dudenhefner and Jakob Rehof. 2019. Principality and Approximation under Dimensional Bound. Proc. ACM Program. Lang. 3, POPL, Article 8 (January 2019), 29 pages. https://doi.org/10.1145/3290321

1 INTRODUCTION

Intersection type systems [Barendregt et al. 1983; Coppo and Dezani-Ciancaglini 1980; Coppo et al. 1981; Pottinger 1980] are important in the theory of typed λ-calculus [Barendregt et al. 2013] as well as in the theory and practice of programming languages. Application areas include refinement types, abstract interpretation, model checking, object calculi, process types, and synthesis (see e.g. Hüttel et al. [2016] for an overview) as well as type systems for modern and experimental programming languages (see e.g. the language Forsythe by Reynolds [1996] and the discussion and references of Alpuim et al. [2017] for an up-to-date perspective on intersection types in modern programming languages). Because intersection types characterize semantic properties of λ-terms, including normalization and solvability properties [Barendregt et al. 2013], standard type theoretic decision problems are undecidable, including the problem of type checking (given a term and a type, does the term have the type?) and inhabitation (given a type, does there exist a term having the type?). The recently introduced framework of dimensional intersection type calculi [Dudenhefner and Rehof 2017b,c] offers a principled way of bounding the systems and thereby exposing meaningful fragments for which important decision problems are decidable.

Authors’ addresses: Andrej Dudenhefner, TU Dortmund University, Dortmund, Germany, andrej.dudenhefner@cs.tu-dortmund.de; Jakob Rehof, TU Dortmund University, Dortmund, Germany, jakob.rehof@cs.tu-dortmund.de.

This work is licensed under a Creative Commons Attribution 4.0 International License.

© 2019 Copyright held by the owner/author(s).
2475-1421/2019/1-ART8
https://doi.org/10.1145/3290321
The main contribution of the present paper is the development of an algebraic and algorithmic theory of principality for dimensional intersection type calculi. The theory enables inference of principal type information under dimensional bound, it provides an algebraic and algorithmic theory of approximation of classical principal types in terms of computable bases of abstract vector spaces (more precisely, semimodules), and it shows a systematic connection of dimensional calculi to the theory of approximants [Barendregt 1984]. Subsidiary results include decidability of principal inhabitation for intersection types (given a type does there exist an approximate normal form for which the type is principal?). Remarkably, principal inhabitation combined with bounded type inference constructs approximate normal forms of arbitrary terms without using β-reduction.

In the remainder of this introduction we provide necessary background on dimensional calculi (Section 1.1) and on the classical theory of principality for intersection types seen from that perspective (Section 1.2). We then (Section 2) give a more technical summary of our main contributions together with an overview of the organization of the remainder of the paper.

### 1.1 Dimensional Calculi

Intersection type systems are characterized by assigning multiple types to one and the same term, using the principle of intersection introduction: if a term has type σ and also type τ, then it has type σ ∩ τ. In dimensional intersection type calculi [Dudenhefner and Rehof 2017b,c] intersection introduction is considered a measurable resource, giving rise to meaningful bounded subsystems for which important decisions problems are decidable, while approximating the full system arbitrarily under increasing dimensional bound.

Following the presentation of Dudenhefner and Rehof [2017c] let us here briefly recall (informally, complete definitions will be given below) that dimensional intersection type calculi are based on judgements of the form \( Γ ⊢ P : σ \), where \( P \) is an elaboration, that is, a pure λ-term \( M \) in which each subterm is decorated with the set of types assigned to it in a type derivation according to an instrumentation of the typing rules of the intersection type system (and we write \( M = [P] \)).

**Example 1.1.** Let \( I = λx.x \). Some possible elaborations of \( I \) in the set-theoretic elaboration system of Dudenhefner and Rehof [2017c] are:

\[ \begin{align*}
&⊢ I_1 : A → A, \text{ with } I_1 = ⟨\lambda x.⟨[A]\rangle⟩⟨[A → A]\rangle \\
&⊢ I_2 : A ∩ B → A, \text{ with } I_2 = ⟨\lambda x.⟨[A]\rangle⟩⟨[A ∩ B → A]\rangle \\
&⊢ I_3 : (A → A) ∩ (B → B), \text{ with } I_3 = ⟨\lambda x.⟨[A,B]\rangle⟩⟨[A → A,B → B]\rangle
\end{align*} \]

In this notation, each subterm occurrence carries a *decoration* of the form \( ⟨S⟩ \) where \( S \) is a finite set of types written as \( [A_1, \ldots, A_n] \). Such a decoration indicates that the subterm occurrence has been assigned the types \( A_1, \ldots, A_n \) during the typing derivation of the term in which it occurs. We use the so-called *strict* variants of the intersection type system (see van Bakel [2011]) in which types are stratified such that components \( A_i \) of intersections \( A_1 ∩ \cdots ∩ A_n \) are not themselves intersections. This way, we obtain a canonical representation of decoration sets. Elaboration systems instrument intersection introduction such that, if \( Γ ⊢ P_1 : σ_1 \) and \( Γ ⊢ P_2 : σ_2 \), then \( Γ ⊢ P_1 ∪ P_2 : σ_1 \cap σ_2 \). Here, the operation \( P_1 ∪ P_2 \) recursively unions corresponding sets of types in the decorations of the elaborations in the premises, \( P_1 \) and \( P_2 \). The operation is defined for elaborations of the same underlying pure term, denoted \( [P] \), arising by erasing all decorations (so, above we have \( [I_1] = [I_2] = [I_3] = ⟨λx.x⟩ \)). Thus, \( I_3 \) above arises as \( I_3 = I_A ∪ I_B \) with \( I_A = ⟨λx.⟨[A]\rangle⟩⟩⟨[A → A]\rangle \) and \( I_B = ⟨λx.⟨[B]\rangle⟩⟩⟨[B → B]\rangle \).

Elaborations are equipped with a *norm*, \( ||P|| \), which measures the maximal size of decoration sets in \( P \), and satisfies the triangle inequality \( ||P ∪ Q|| ≤ ||P|| + ||Q|| \). Referring to Example 1.1, we have \( ||I_1|| = ||I_2|| = 1 \) and \( ||I_3|| = 2 \). The norm can be understood as a measure of the amount of

---

**Proc. ACM Program. Lang., Vol. 3, No. POPL, Article 8. Publication date: January 2019.**
intersection introduction required to type the term at its type and can be used to bound the system. We write $\Gamma \vdash_n M : \sigma$ if $\Gamma \vdash P : \sigma$ for some $P$ with $M = [P]$ and $\|P\| \leq n$. The \textit{dimension} of a $\lambda$-term $M$ at $\Gamma$ and $\sigma$ is the smallest number $n$ such that $\Gamma \vdash_n M : \sigma$. Any typing of the original intersection type system can be recovered in the system $\vdash_n$ for sufficiently large dimensional parameter $n$.

It was shown by Dudenhefner and Rehof [2017b] that each dimension-bounded system $\vdash_n$ enjoys subject reduction, thereby constituting a meaningful type system for each dimensional bound $n$. Moreover, whereas the dimension-bounded inhabitation problem (given a type and a dimension $n$, is there a term having the type in $\vdash_n$?) is undecidable for the standard set-theoretic interpretation of intersection (as an associative, commutative and idempotent operator on types), inhabitation is decidable (and ExpSpace-complete) when decorating intersections are treated as multisets (that is, intersection is treated as a non-idempotent operator).

The typability problem (given a term and a dimension $n$, does the term have a type in $\vdash_n$?) was shown by Dudenhefner and Rehof [2017c] to be decidable (and PSPACE-complete) under the standard set-theoretic interpretation of intersection types. This result was obtained by reducing typability to a standard type unification problem on type constraints over $n$-dimensional elaborations decorated with type variables. However, it is not clear how the typings discovered by the algorithm by Dudenhefner and Rehof [2017c] might relate to the structure of all typings of the given term. In particular, it is natural to ask whether there exists some way to represent and compute \textit{principal typing information} within bounded dimension.

Because the theory of principality for intersection types is related to the semantic theory of \textit{approximants} [Barendregt 1984, Chap.14] in the $\lambda$-calculus, we need to provide some of that background, as a final preparation for a more technical summary of our results.

### 1.2 Principality and Approximants

Since the founding papers of the theory of intersection types (see Barendregt et al. [2013] for an overview) it is known that they are related to semantic properties, both operational and denotational, of $\lambda$-terms. When equipped with a universal type, $\omega$, they systematically characterize both solvability and normalization properties [Coppo et al. 1981], and they can be used to define $\lambda$-models via intersection type filters [Barendregt et al. 1983]. Connections to the semantic theory of \textit{approximants} [Barendregt 1984, Chap.14] in the $\lambda$-calculus were used in the seminal paper by Coppo et al. [1980] to establish the foundations for the theory of principality for intersection types. Approximants are $\beta$-normal forms extended with the constant $\bot$ denoting undefined computation (approximate normal forms are essentially finite rooted segments of Böhm-trees). The set $\mathcal{A}(M)$ of approximate normal forms of a term $M$ constitute a meet-semilattice under the approximation ordering $\sqsubseteq_\bot$, where smaller approximants are less defined than larger ones, the ordering being generated from positing $\bot$ as the smallest element. Approximate normal forms can be inductively endowed with unique (up to renaming) most general typings in the form of \textit{principal pairs} (type environment, type). From the principal pair of a term all typings of the term can be generated under a specified set of \textit{generating operations}, consisting of \textit{chains} of so-called type expansions, liftings (or, rise) and substitutions [Coppo et al. 1980]\footnote{Although we provide some background here, we also recommend the overview by van Bakel [2011] for more information. We use the term \textit{principal pair} ($\Gamma$, $A$) for approximate normal forms in the exact sense of Coppo et al. [1980, Definition 8] (see also van Bakel [2011, Definition 10.6]).}. Furthermore, a term $M$ has a type in the intersection type system of Coppo et al. [1980], if and only if some approximate normal form of $M$ has the type [Coppo et al. 1979]. Indeed, the meet-semilattice of approximate normal forms ordered by $\sqsubseteq_\bot$ is isomorphic to the set of their principal pairs ordered by the relation $\sqsubseteq_\omega$ in which smaller elements are obtained from larger ones by replacing type variables by the universal type $\omega$ [Coppo et al. 1980]. If an
arbitrary (possibly non-normal form) term has only finitely many approximate normal forms, it has a principal pair \((\Gamma, A)\), obtainable as the least upper bound wrt. \(\sqsubseteq_\omega\) of the principal pairs of the approximate normal forms of the term.

**Example 1.2.** Let \(\omega = \lambda z. zz\) and \(M = \omega (\omega x)\). We list some approximants \(F_i \in \mathcal{A}(M)\) in increasing order wrt. \(\sqsubseteq_\perp\) together with their principal typings \(\Gamma_i \vdash F_i : \sigma_i\) shown to the right.

\[
\begin{align*}
F_0 &= \bot & \{x : \omega\} \vdash F_0 : \omega \\
F_1 &= x \bot \bot & \{x : \omega \to \omega \to \alpha\} \vdash F_1 : \alpha \\
F_2 &= x \bot (x \bot) & \{x : (\omega \to \alpha \to \alpha) \cap (\omega \to \beta)\} \vdash F_2 : \alpha \\
F_3 &= x x (x x) & \{x : (\delta \to \alpha \to \alpha) \cap (\gamma \to \beta)\} \vdash F_3 : \alpha \\
F_4 &= x x (x x) & \{x : (\delta \to \beta \to \beta) \cap (\gamma \to \beta)\} \vdash F_4 : \alpha
\end{align*}
\]

The type \(\omega\) satisfies \(\omega = \sigma \to \omega\) and \(\sigma \cap \omega = \sigma\) for all \(\sigma\). Using these equations one can see that \(\Gamma_i(x) \sqsubseteq_\omega \Gamma_{i+1}(x)\) for \(i = 0, 1, 2, 3\). If we consider elaborations \(\mathcal{M}_i\) of \(M\) corresponding to the typings of the approximants \(F_i\) (\(i = 0, \ldots, 4\)), it is interesting to observe\(^2\) the decorations of variable \(x\): the occurrence of \(x\) will give rise to increasing norm (regarding a decoration with \(\omega\) as the empty intersection [] with \(|x([])| = 0\), \(x\) being typed at intersections with an increasing number of components so that \(|\mathcal{M}_i|| = i\). For example, corresponding to \(F_4\) we have the elaboration

\[
\mathcal{M}_4 = (\Omega_1 (\Omega_2 x ([\delta \to \beta \to \alpha, \gamma \to \beta, \delta])) ([\beta \to \alpha, \beta])) ([\alpha])
\]

with \(|\mathcal{M}_4|| = 4\) and where

\[
\begin{align*}
\Omega_1 &= (\lambda z. z([\beta \to \alpha]) z([\beta])) ([((\beta \to \alpha) \cap \beta) \to \alpha]) \\
\Omega_2 &= (\lambda z. z([\delta \to \beta \to \alpha, \gamma \to \beta]) z([\delta, \gamma])) ([((\delta \to \beta) \cap \gamma) \to \beta, ((\delta \to \beta) \cap \delta) \to \beta \to \alpha])
\end{align*}
\]

**Example 1.3.** Consider the Church numerals \(c_n = \lambda f. \lambda x. f^n x\) (where \(f^0 = M = f(f^0 M)\)) shown with their principal types:

\[
\begin{align*}
c_0 &= \lambda f. \lambda x. x : \alpha \to \alpha_0 \to \alpha_0 \\
c_1 &= \lambda f. \lambda x. f x : (\alpha_0 \to \alpha_1) \to \alpha_0 \to \alpha_1 \\
c_2 &= \lambda f. \lambda x. f (f x) : ((\alpha_0 \to \alpha_1) \cap (\alpha_1 \to \alpha_2)) \to \alpha_0 \to \alpha_2 \\
c_n &= \lambda f. \lambda x. f^n x : ((\alpha_0 \to \alpha_1) \cap (\alpha_1 \to \alpha_2) \cap \cdots \cap (\alpha_{n-1} \to \alpha_n)) \to \alpha_0 \to \alpha_n
\end{align*}
\]

Consider the fixed-point combinator \(Y = \lambda f. (\lambda x. f(xx))(\lambda x. f(xx))\). The set of its approximants \(\mathcal{A}(Y) = \{\bot\} \cup \{\lambda f. f^n \bot \mid n \geq 1\}\) is infinite. There is no single typing of \(Y\) from which all its typings can be obtained under the generating operations mentioned above.

One may notice here that, in the intersection type system, the generating operations for obtaining arbitrary typings from principal typings are significantly more complicated than in most other systems (for example, in simple types there is just a single generating operation, type substitution). In particular, the operation of type expansion is rather complex. Intuitively, type expansions replace certain subformulae \(\tau\) within a type \(\sigma\) by an intersection \(\tau_1 \cap \cdots \cap \tau_n\) where each \(\tau_i\) is a fresh copy of \(\tau\).\(^3\) The following observation is important to the perspective on principality developed in the present paper (cf. van Bakel [2011, Lemma 10.16]). Expansions can be divided into two kinds. The first kind copies the entire type of the term and corresponds to a top-level application of intersection introduction to the typing of the term. We shall refer to such expansions as top-level expansions.

**Example 1.4.** The term \(I = \lambda x. x\) has principal typing \(\theta \vdash I : \alpha \to \alpha\). Another typing which \(I\) has is \(\emptyset \vdash I : (\alpha_1 \to \alpha_1) \cap (\alpha_2 \to \alpha_2)\). It is obtained from the principal one by expanding the whole type \(\alpha \to \alpha\) into \((\alpha_1 \to \alpha_1) \cap (\alpha_2 \to \alpha_2)\).

\(^2\)To a reader who might not see this intuitively: proceed reading to Figure 1 and then return to the example.

\(^3\)See van Bakel [2011, Definition 10.12] for the exact definition of type expansion used here. See Carlier and Wells [2005] for a survey of expansion.
The second kind of expansion copies a strict subformula of the type of the term. Such will be referred to as deep expansions. On a normal form (considered here for simplicity) such expansions correspond to proof transformations on subterms replacing \( \{x : \tau \rightarrow \tau', \ldots\} \vdash xN : \tau' \) by \( \{x : (\tau_1 \cap \cdots \cap \tau_n) \rightarrow \tau', \ldots\} \vdash xN : \tau' \), where the premise typing of \( N \) at \( \tau \) is replaced by a premise typing of \( N \) at \( \tau_1 \cap \cdots \cap \tau_n \).

Example 1.5. The term \( J = \lambda x.xI \) has principal typing \( \Theta \vdash J : ((\alpha \rightarrow \alpha) \rightarrow \beta) \rightarrow \beta \). By expanding the subformula \( \alpha \rightarrow \alpha \) into \( (\alpha_1 \rightarrow \alpha_1) \cap (\alpha_2 \rightarrow \alpha_2) \) we obtain the typing \( \Theta \vdash J : (((\alpha_1 \rightarrow \alpha_1) \cap (\alpha_2 \rightarrow \alpha_2)) \rightarrow \beta) \rightarrow \beta \) by a deep expansion. Returning to elaborations (see Example 1.1), we have the corresponding elaborations \( J_1 \) and \( J_2 \):

\[
J_1 = (\lambda x. x((\alpha \rightarrow \alpha) \rightarrow \beta)I_\alpha)((\alpha \rightarrow \alpha) \rightarrow \beta),
\]

where \( I_\alpha = (\lambda x. x([\alpha]))([\alpha \rightarrow \alpha]) \)

\[
J_2 = (\lambda x. x((\alpha_1 \rightarrow \alpha_1) \cap (\alpha_2 \rightarrow \alpha_2) \rightarrow \beta)I_{\alpha_1, \alpha_2}((\beta)\rightarrow ([A]))
\]

where \( I_{\alpha_1, \alpha_2} = (\lambda x. x([\alpha_1, \alpha_2]))([\alpha_1 \rightarrow \alpha_1, \alpha_2 \rightarrow \alpha_2]) \)

and \( A = (((\alpha_1 \rightarrow \alpha_1) \cap (\alpha_2 \rightarrow \alpha_2)) \rightarrow \beta) \rightarrow \beta \)

Notice that, not unexpectedly, expansions may be norm-increasing on elaborations. Thus we have \( \|J_1\| = 1 \) and \( \|J_2\| = 2 \) (since \( \|I_{\alpha_1, \alpha_2}\| = 2 \)).

It turns out [van Bakel 2011] that the generating operations of principality can be simplified by factoring typings through the so-called relevant variant of the intersection type system, \( \vdash R \). For \( \{x : A_1 \cap \cdots \cap A_n, \ldots\} \vdash R M : \sigma \) to be derivable, each component type \( A_i \) of \( x \) must actually be used in the derivation (if \( x \) does not occur in \( M \), then \( x \) must be assigned the type \( \omega \)). For example, the typing given by elaboration \( I_2 \) in Example 1.1 is not relevant, because \( x \) is not used at type \( B \). Similarly, for the term \( K = \lambda x. \lambda y.x \), we can not derive the type \( \alpha \rightarrow \beta \rightarrow \alpha \), but we have \( \Theta \vdash R K : \alpha \rightarrow \omega \rightarrow \alpha \). Every typing of a term in the relevant system can be obtained from a principal pair using only expansions followed by a single type substitution: for any approximate normal form \( F \) with principal pair \( (\Gamma^*, \sigma^*) \), whenever \( \Gamma \vdash R F : \sigma \), one has \( (\Gamma, \sigma) = [E; S](\Gamma^*, \sigma^*) \) for some sequence of expansions \( E \) followed by some type substitution \( S \) [van Bakel 2011, Theorem 10.26]. Moreover, it can be shown [van Bakel 2011, Theorem 10.27] that any typing in the full intersection type system (as given in the so-called essential variant [van Bakel 2011, Sec.4]) can be obtained from the relevant typing by a further step of lifting, which is a subtyping relation on pairs (such a step would lead from the typing \( I_1 \) to the typing \( I_2 \) of Example 1.1).

2 TECHNICAL SUMMARY AND OUTLINE

Although the principal typings of normal forms are known to exist in dimension 1 [Dudenhefner and Rehof 2017b, Proposition 25], principal typings of arbitrary terms may require arbitrarily high dimension (see Example 1.2). Since we cannot decide finiteness of the set of approximants of a given term, the existence of a dimensional bound sufficient to discover a principal typing is undecidable. The most desirable state of affairs one could hope for would be to have a computable notion of principality under dimensional bound which would, in some appropriate sense, coincide with standard (unbounded) principal type information (when it exists) for sufficiently high dimension, thereby approximating standard principality under increasing bound. The main objective of the present study is to show that there is such a notion of principality in bounded dimension, which is both mathematically natural and computationally effective. Our notion of principality arises by an analysis of the classical theory of intersection type principality in terms of abstract vector spaces (semimodules). Based on this notion, we extend the algorithmic framework of Dudenhefner and Rehof [2017c] to achieve type inference of principal type information for intersection types under dimensional bound.
2.1 Semimodular Structure and Compact Bases

In the first part of this paper (Section 4 and Section 5) we observe that the notion of dimension can be explained in terms of abstract linear algebra, specifically, in the form of semimodular structure of types and elaborations (Section 4). A semimodule over a semiring \( R \) (see, e.g., Golan [1999], and see Section 4) generalizes a module over a ring by requiring only that the semiring \( R \) be a commutative monoid rather than an abelian group.

The semimodular structure is identified in detail in Section 4. Intuitively, an intersection type \( A_1 \cap \cdots \cap A_n \) (represented in strict form [van Bakel 2011] where the \( A_i \) are not intersections) can be seen as an abstract vector (element of a semimodule) with \( n \) components, and type substitutions can be organized as the accompanying semimodules. From this standpoint, intersection introduction can be seen as vector addition: when inferring \( \Gamma \vdash P_1 \sqcup P_2 : A_1 \cap A_2 \) from \( \Gamma \vdash P_1 : A_1 \) and \( \Gamma \vdash P_2 : A_2 \), we perform abstract vector addition both on the types \( A_1 \) and \( A_2 \) and on the elaborations \( P_1 \) and \( P_2 \). Here, the elaborations of a term are also organized as a semimodule, viewed as abstract matrices over the semimodule of intersection types, just as the space of \( n \times m \)-matrices over elements of a vector space is again a vector space. Under the semimodular view, if for example we have substitutions \( a_1 = \{ \alpha := \alpha_1 \}, a_2 = \{ \alpha := \alpha_2 \}, a_3 = \{ \alpha_1 := A \}, \) and \( a_B = \{ \alpha_2 := B \} \), the type \( (A \to A) \cap (B \to B) \) can be seen as a linear combination,

\[
(A \to A) \cap (B \to B) = ((a_A \circ a_1) + (a_B \circ a_2))(\alpha \to \alpha) = (a_A \circ a_1)(\alpha \to \alpha) \cap (a_B \circ a_2)(\alpha \to \alpha)
\]

the type (“vector”) \( \alpha \to \alpha \) being “multiplied” by the “scalar” (substitution) \( ((a_A \circ a_1) + (a_B \circ a_2)) \), where “multiplication” \( \circ \) (composition) and “addition” \( + \) (given by \( (a + b)A = aA \cap bA \)) is defined in the semiring of substitutions. One can then talk about linear combinations and linear independence of types and elaborations.

A fundamental problem with the standard theory of principality from the perspective of approximating principal type information in bounded dimension turns out to be this: because expansion may be arbitrarily norm-increasing, this operation cannot be used as a generator in bounded dimension. We solve this problem using semimodular structure, providing a new analysis of the generating operations (expansion and instantiation). By considering linear combination as a generator (span in linear algebra), top-level expansions (Example 1.4) are seen to play a different role than do deep expansions (Example 1.5): chains of top-level expansions and substitutions can be seen as linear combinations, and in Section 5 we show that, generally, it is possible to span all relevant typings in a given dimension from a semimodular basis of typings understood as a linearly independent spanning set. In contrast, deep expansions (Example 1.5) can be seen as a way of producing new basis vectors of increasing dimension. We use this analysis to achieve a finitary and canonical representation of all relevant typings in a given dimension: we prove the existence of unique, finite compact bases \( B_{M,n} \) spanning all \( n \)-dimensional typings of \( M \), the main technical contribution of the first part of the paper. Compactness is a new proof-theoretic notion capturing strongly irredundant typings. Since bases of higher dimension span ever larger types wrt. the relation \( \subseteq_\omega \), bases \( B_{M,n} \) approximate standard principality under the semimodular notion of principality for bounded dimension, capturing standard principal types for sufficiently high dimension.

2.2 Type Inference and Approximation

In the second part of the paper (Section 6 and Section 7) we develop algorithms for the construction of compact bases, leading to inference of principal typings of approximate normal forms, for sufficiently high dimension. As a consequence of compactness and uniqueness properties of principal typings we obtain (Section 6) a polynomial time algorithm for constructing the principal approximate normal inhabitant of a principal pair and show decidability of principal inhabitation (is there an approximate normal form having a given pair as its principal pair?). This result is striking in view...
of the fact that the general intersection type inhabitation problem is undecidable [Urzyczyn 1999]. Based on these results, we provide in Section 7 a sound and complete algorithm for constructing approximate normal forms of a given term $M$ via dimension-bounded type inference. Our algorithm is an extension of the typability algorithm given by Dudenhefner and Rehof [2017c]. Moreover, the algorithm is used to compute compact bases in bounded dimension, thereby yielding a complete algorithmic realization of the preceding theory of compact semimodular bases. Interestingly, terms are “evaluated” to approximate normal forms under increasing dimension by type inference combined with principal inhabitation.

Related work is discussed in Section 8, and Section 9 concludes the paper with a broader reflection on the significance of our results and future work. Section 10 contains a few proofs left out of the main text for readability.

3 PRELIMINARIES

Following Dudenhefner and Rehof [2017b,c] we introduce an elaboration system for the intersection type system, here given in the relevant variant studied by van Bakel [2011]. In comparison to Dudenhefner and Rehof [2017b,c] we extend elaborations to encompass the empty intersection type, $\omega$, which is the universal type [Barendregt et al. 1983; Coppo et al. 1980]. This extension is algebraically significant, because it provides a neutral element both for vector addition in the semimodular structure of types (the type $\omega$ satisfying $\sigma \cap \omega = \sigma$) and in the semimodular structure of elaborations (the null elaboration $0_M$ satisfying $P \sqcup 0_M = P$) developed in Section 4, and it is important for the theory of approximation (Section 5).

3.1 Terms, Approximants, and Types

The set $\Lambda$ of untyped $\lambda$-terms is defined by: $M, N ::= x | (\lambda x.M) | (MN)$. The set $\Lambda_\bot$ of $\bot$-terms extends $\Lambda$ and is given by: $M, N ::= \bot | x | (\lambda x.M) | (MN)$. Unless otherwise stated, terms ranged over by $M, N, P, Q$ are members of $\Lambda_\bot$. We denote by $FV(M)$ the set of free term variables in $M$. The set $\mathcal{A}$ of approximate normal forms (or $\bot$-$\beta$-normal forms), ranged over by $F, G$ is defined by: $F, G ::= \bot | (x \, F_1 \cdots F_n), n \geq 0 | (\lambda x.F)$ where $F \neq \bot$. Following strict restrictions of intersection types [van Bakel 2011, Definition 4.1] we stratify intersection types into strict types $(A, B, \ldots)$ and strict intersection types $(\sigma, \tau, \ldots)$:

$$T_0 \ni A, B ::= \alpha \mid \sigma \rightarrow A, \quad \mathcal{T} \ni \sigma, \tau ::= A_1 \sqcap \cdots \sqcap A_n \ (n \geq 0)$$

where $\alpha, \beta, \ldots$ range over type variables. We may write $A_1 \sqcap \cdots \sqcap A_n$ as $\bigcap_{i=1}^n A_i$. The empty intersection $\bigcap_{i=1}^n A_i$ with $n = 0$ is denoted $\omega$. The set of type variables occurring in $\sigma$ is denoted by $TV(\sigma)$. The operator $\cap$ is tacitly regarded as associative, commutative and idempotent. We shall tacitly assume that types are normalized under equivalence such that $\omega$ is used as representative for any type $\tau$ with $\tau = \omega$, and such that in a type of the form $A_1 \sqcap \cdots \sqcap A_n$, $n > 1$, we have $A_i \neq A_j$ for $i \neq j$. Notice that $A \sqcap \omega = A$ for all $A \in T_0$, since $\omega$ is the empty intersection. We write $\sigma = \tau$ if and only if $\sigma$ and $\tau$ are identical modulo associativity, commutativity and idempotence of $\cap$ together with normalization. Note that we have (using normalized form of types): $A_1 \sqcap \cdots \sqcap A_n = B \Rightarrow (n = 1 \land A_1 = B)$.

3.2 Elaborations

We introduce set-theoretic elaborations [Dudenhefner and Rehof 2017b,c] with the generalization that decoration sets may be empty. Set-theoretic elaborations, ranged over by $P, Q, R$, are $\lambda \bot$-terms decorated with finite (possibly empty) sets, ranged over by $S$, of strict types, written $S = [A_1, \ldots, A_n]$, and such sets are referred to as decoration sets:

$$P, Q ::= \bot \langle S \rangle \mid x \langle S \rangle \mid (\lambda x.P)\langle S \rangle \mid (PQ)\langle S \rangle$$

Proc. ACM Program. Lang., Vol. 3, No. POPL, Article 8. Publication date: January 2019.
When convenient, we identify intersections \( \sigma = A_1 \cap \ldots \cap A_n \) with sets \([A_1, \ldots, A_n] \), i.e. we identify decorations \( \langle \sigma \rangle = \langle [A_1, \ldots, A_n] \rangle \), \( \langle \omega \rangle = \langle [] \rangle \), and \( A \in \sigma \) means \( A = A_i \) for some \( i \in \{1, \ldots, n\} \).

For each untyped term \( M \) we define the special elaboration \( 0_M \) by:

\[
0_\bot \equiv \bot \langle \omega \rangle \quad 0_x \equiv x \langle \omega \rangle \quad 0_{\lambda x. M} \equiv (\lambda x.0_M) \langle \omega \rangle \quad 0_{(MN)} \equiv (0_M 0_N) \langle \omega \rangle
\]

Let \( [P] \) denote the untyped term \( M \in \Lambda_{\bot} \) arising from erasing all decorations from \( P \). Define the operation \( P \sqcup Q \) on elaborations \( P, Q \) with \( [P] \equiv [Q] \):

\[
\begin{align*}
\bot \langle S \rangle \sqcup \bot \langle S' \rangle & \equiv \bot \langle S \sqcup S' \rangle \\
\lambda \langle S \rangle \sqcup \lambda \langle S' \rangle & \equiv \lambda \langle S \sqcup S' \rangle \\
P \langle S \rangle \sqcup (P \langle S' \rangle) & \equiv P \langle S \sqcup S' \rangle \\
(P \langle S \rangle) \sqcup (Q \langle S' \rangle) & \equiv (P \sqcup Q) \langle S \sqcup S' \rangle
\end{align*}
\]

Notice that, for \( [P] \equiv M \), we evidently have \( 0_M \sqcup P = P \sqcup 0_M = P \).

Following Dudenhefner and Rehof [2017b,c] we introduce an elaboration system for untyped terms which records the usage of \((\cap \cap)\). We provide (Figure 1, left) an elaborated version of the relevant intersection type system, as defined by van Bakel [2011, Def. 10.1], and denoted (both by van Bakel [2011] and here) as \( \vdash_{\cap} \). A judgement \( \Gamma \vdash_{\cap} P : [A_1, \ldots, A_n] \) signifies that the term \( [P] \) elaborates to \( P \) in the context of \( \Gamma \) at the type \( A_1 \cap \ldots \cap A_n \), and \( \Gamma \vdash_{\cap} [P] : A_1 \cap \ldots \cap A_n \) in the system \( \vdash_{\cap} \) of van Bakel [2011]. We denote by \( P : A_1 \cap \ldots \cap A_n \) that the top-level decoration of \( P \) is \( \langle [A_1, \ldots, A_n] \rangle \). The system \( \mathcal{R} \) (Figure 1, right) is a technically useful simplified presentation of \( \vdash_{\cap} \).

Notice that for \( [P] \equiv M \), \( x : \vdash \vdash_{\cap} P : \sigma \) if and only if \( P \equiv 0_\bot \) and \( \sigma = \omega \). For \( x \in \mathrm{FV}(M) \) and \( [P] \equiv M \) and we let \( T_x(P) = \{ A \in \mathcal{T}_\sigma \mid A \in S \text{ for some subterm } x\langle S \rangle \text{ of } P \} \). The context \( \Gamma_P \) is defined by \( \Gamma_P = \{ (x : \vdash \vdash_{\cap} A) \mid T_x(P) = \{ A_1, \ldots, A_n \} \} \).

It is easy to see that whenever \( \Gamma \vdash_{\cap} P : \sigma \) one has \( \Gamma_P \vdash_{\cap} P : \sigma \). As a consequence, we have \( \Gamma_P \vdash_{\cap} P : \sigma \iff P : \sigma \text{ and } \sigma \in \mathcal{R} \). Therefore, \( \mathcal{R} \) is the set of well-typed elaborations in the relevant type system such that for any elaboration \( P \in \mathcal{R} \) the corresponding relevant context \( \Gamma \) and the assigned type \( \sigma \) are implicitly given by \( \Gamma = \Gamma_P \) and \( P : \sigma \).

Fig. 1. Elaborations for the Relevant Type System

\[
\begin{align*}
\Gamma, x : \sigma & \vdash_{\cap} P : [A] \\
\Gamma & \vdash_{\cap} (\lambda x.P)(\langle \sigma \rightarrow [A] \rangle) : [\sigma \rightarrow [A]] \quad (\rightarrow_{\cap} 1) \\
\Gamma & \vdash_{\cap} P : [A] \quad (x \notin \mathrm{dm}(\Gamma)) \quad (\rightarrow_{\cap} 2) \\
\Gamma_1 \cap \Gamma_2 & \vdash_{\cap} (PQ)(\langle [B] \rangle) : [B] \quad (\rightarrow_{E_{\cap}}) \\
\bigcap_{i=1}^n \Gamma_i & \vdash_{\cap} \bigcap_{i=1}^n P_i : [A_1, \ldots, A_n] \quad (\cap_{E_{\cap}}) \\
\emptyset & \vdash_{\cap} 0_M : \omega \quad (\omega_{E_{\cap}})
\end{align*}
\]
Let $R_M$ denote the set of well-typed elaborations of $M$ in the relevant type system, that is, $R_M = \{ P \mid \exists \Gamma, \sigma. \Gamma \vdash_R P : \sigma \land [P] = M \} = \{ P \in R \mid [P] = M \}$.  

Definition 3.1 (Norm, dimension). Following Dudenhefner and Rehof [2017b,c] define the max-norm $\| \| \|$ on elaborations, where $|S|$ denotes the cardinality of finite set $S$: $\| \bot(S) \| = |S|$, $\| x(S) \| = |S|$, $\| (\lambda x.P)(S) \| = \max (\|P\|, |S|)$, $\| (PQ)(S) \| = \max (\|P\|, \|Q\|, |S|)$.  

Write $\Gamma \vdash_R M : \sigma$ if there exists $P$ such that $\Gamma \vdash_R P : \sigma$, $\| P \| \leq n$, and $[P] = M$. It is clear that $\Gamma \vdash_R M : \sigma$ if and only if $\Gamma \vdash_{R,n} M : \sigma$ for some $n \geq 0$. The dimension of $M$ at $\Gamma$ and $\sigma$ is the smallest $n \geq 0$ such that $\Gamma \vdash_{R,n} M : \sigma$ (and $\infty$ if it is not the case that $\Gamma \vdash_{R,n} M : \sigma$ for any $n \geq 0$).

4 SEMIMODULAR STRUCTURE

4.1 Semimodules over Types and Elaborations

Following van Bakel [1993, Definition 4.1] we define strict substitutions $s : T \rightarrow T$ (notice that we use $\alpha$ for type variables and $\varphi$ for types in $T_0 \cup \{ \omega \}$, whereas the notation used by van Bakel [1993] is exactly the other way around). For $\varphi \in T_0 \cup \{ \omega \}$ the strict substitution $\{ \alpha := \varphi \} : T \rightarrow T$ is defined as follows:

- $\{ \alpha := \varphi \}(\alpha) = \varphi$  
- $\{ \alpha := \varphi \}(\beta) = \beta$, if $\beta \neq \alpha$  
- $\{ \alpha := \varphi \}(\sigma \rightarrow A) = \omega$, if $\{ \alpha := \varphi \}(A) = \omega$  
- $\{ \alpha := \varphi \}(\sigma \rightarrow A) = \{ \alpha := \varphi \}(\sigma) \rightarrow \{ \alpha := \varphi \}(A)$, if $\{ \alpha := \varphi \}(A) \neq \omega$  
- $\{ \alpha := \varphi \}(A_1 \cap \cdots \cap A_n) = \{ \alpha := \varphi \}(A_1) \cap \cdots \cap \{ \alpha := \varphi \}(A_n)$

Notice that application of substitution automatically implements the equation $\sigma \rightarrow \omega = \omega$ by normalizing to $\omega$. A strict substitution $\{ \alpha := \varphi \}$ is everywhere defined on $T$, since it acts as the identity on all variables other than $\alpha$. Let the substitution $s_\omega$ be given by $\{ \alpha := \omega \}$ for all $\alpha$, and let $id$ be the identity substitution $\{ \alpha := \alpha \}$ for all $\alpha$. For functions $a, b$ from intersection types to intersection types, define $a \circ b$ by setting $(a \circ b)(\sigma) = a(b(\sigma))$, and define $a + b$ by setting $(a + b)(\sigma) = a(\sigma) \cap b(\sigma)$. Let $R$ be the smallest set containing the strict substitutions and closed under formation with $\circ$ and $\cup$. We shall sometimes consider the subset $R_0 \subseteq R$ containing only compositions under $\circ$.

**Lemma 4.1.** Let $a \in R$. Then $\sigma \in T$ implies $a(\sigma) \in T$, and $a(A_1 \cap \cdots \cap A_n) = a(A_1) \cap \cdots \cap a(A_n)$.

A semimodule $M$ over a semiring $R$ (see, e.g., Golan [1999]) generalizes a module over a ring by requiring only that the semiring $(R, +, \cdot)$ be an additive commutative monoid rather than an abelian group. In detail, a left $R$-semimodule consists of an additively-written ($\oplus$) commutative monoid $(M, @)$ and a map $R \times M \rightarrow M$ satisfying the axioms: $r(m \oplus n) = rm \oplus rn$, $(r + s)m = rm \oplus sm$, $(rm)s = r(sm)$, $1m = m$, $0Rm = r0 = 0_M$.

**Proposition 4.2.** The structure $(R, +, \circ)$ is a semiring with $s_\omega$ the neutral element for $+$ and $id$ the neutral element for $\circ$.

For $a \in R$ and $\sigma \in T$ we write $a \sigma = a(\sigma)$. We can organize $T$ as a left $R$-semimodule by considering $\cap$ as vector addition ($\oplus$) and $\omega$ as the null vector and neutral element for $\cap$. With these definitions $(T, \cap)$ is evidently a commutative monoid.

**Theorem 4.3.** The structure $(T, \cap)$ is a left $(R, +, \circ)$-semimodule.

For each term $M \in A_\perp$ the structure $(R_M, \cup)$ is a commutative monoid with $0_M$ as neutral element. Define for $a \in R$ and $P \in R_M$ the map $R \times R_M \rightarrow R_M$ sending $(a, P)$ to $aP$, where the elaboration $aP$ is given by pointwise application of $a$ to the sets in $P$. More precisely, for $S = \{ A_1, \ldots, A_n \}$ we let $aS = \{ a(A_i) \mid 1 \leq i \leq n, a(A_i) \neq \omega \}$ and $a(\bot(S)) = \bot(aS), a(x(S)) = x(aS)$,  

Proc. ACM Program. Lang., Vol. 3, No. POPL, Article 8. Publication date: January 2019.
We say that $a((\lambda x.P)(S)) = (\lambda x.aP)(aS)$, and $a((PQ)(S)) = ((aP)(aQ))(aS)$. We have $aP \in \mathbb{R}_M$ for all $a \in \mathbb{R}$ and $P \in \mathbb{R}_M$ (proven by induction on derivation of $P$).

**Theorem 4.4.** The structure $(\mathbb{R}_M, \sqcup)$ is a left $(\mathbb{R}, +, \circ)$-semimodule.

We have the properties of the norm (Definition 3.1) on $\mathbb{R}_M$:

$$\|P\| \geq 0 \quad \|0_M\| = 0 \quad \|P\| = 0 \Rightarrow P = 0_M \quad \|P \sqcup Q\| \leq \|P\| + \|Q\|$$

The semimodular structure of $\mathbb{R}_M$ corresponds exactly to the fact that the $m \times n$-matrices over a field constitute a vector space under matrix addition and scalar multiplication [Artin 2011].

The structures defined above allow us to take an algebraic view of the generating operations of instantiation and top-level expansion, as mentioned informally in Section 2.1, illustrated by the following Example 4.5.

**Example 4.5.** Consider the elaboration $I = (\lambda x.x\langle[\alpha]\rangle\langle[\alpha \to \alpha]\rangle)$ of the identity $I = \lambda x.x$. For this particular term, $I$, we can in fact generate all its relevant types from the top-level type $\alpha \to \alpha$ by multiplication with elements of the semiring $\mathbb{R}$: we have $s_\alpha\langle\alpha \to \alpha\rangle = \omega$ and with $r_A \in \mathbb{R}$ mapping $\alpha$ to $A$ for $A \in T_0$ we can generate any type of the form $\cap_{i=1}^n (A_i \to A_i)$ as we have $\Sigma_{i=1}^n r_{A_i}(\alpha \to \alpha) = \cap_{i=1}^n (r_{A_i}(\alpha) \to r_{A_i}(\alpha)) = \cap_{i=1}^n (A_i \to A_i)$. We can consider the corresponding actions on the “matrix”, $I$, in the $\mathbb{R}$-semimodule $\mathbb{R}_I$, where we get the algebraic relations:

$$s_\omega I = ((\lambda x.x\langle[s_\omega(\alpha)]\rangle\langle[s_\omega(\alpha) \to \alpha]\rangle) = (\lambda x.x\langle[\omega]\rangle\langle[\omega]\rangle = 0_I$$

and

$$\Sigma_{i=1}^n r_{A_i} I = \bigsqcup_{i=1}^n r_{A_i} I$$

It is important to notice here that, by passing to corresponding actions on the “matrices” in the $\mathbb{R}$-semimodule $\mathbb{R}_M$, we obtain a way of measuring instances according to the norm $\|\bullet\|$ which is defined on $\mathbb{R}_M$. In our example we have $\|s_\omega I\| = \|0_I\| = 0$ and $\|\Sigma_{i=1}^n r_{A_i} I\| = n$. This way, we have a proof-theoretic algebra on elaborations and types which is connected to the measuring system defined by the norm.

### 4.2 Span and Bases

The following elementary notions are abstracted from vector spaces in the theory of semimodules. Let $(M, \oplus)$ be any left $R$-semimodule. Then there is a notion of linear combinations in $M$ of the form $r_1 X_1 \oplus \cdots \oplus r_n X_n$, where $X_i \in M$ and $r_i \in R$. A subset $S \subseteq M$ is said to be linearly dependent if and only if some element of $S$ can be written as a linear combination of the other members of $S$, that is, for some $X \in S$ there exist $X_1, \ldots, X_n \in S \setminus \{X\}$ and $r_1, \ldots, r_n \in R$ such that $X = r_1 X_1 \oplus \cdots \oplus r_n X_n$. We say that $S$ is linearly independent if $S$ is not linearly dependent. For a subset $S \subseteq M$ we define the span of $S$, denoted span$(S)$, to be the set of all linear combinations from $S$, in detail:

$$\text{span}(S) = \{ \bigoplus_{i=1}^n r_i X_i \mid r_i \in R, X_i \in S, i = 1 \ldots n \}$$

[4]Concretely, letting $|M|$ denote the size of $M$, an elaboration $P \in \mathbb{R}_M$ can be regarded as a $|M| \times \|P\|$-matrix $\tilde{P}$ over $T_0 \cup \{\omega\}$, indexed by an enumeration $i \in \{1 \ldots |M|\}$ of subterm occurrences of $M$ and an enumeration of types $j \in \{1, \ldots, \|P\|\}$, where we have $\tilde{P}[i,j] = A$ if and only if the decoration set in $P$ at subterm occurrence $i$ of $M$ contains the element $A$ at the $j$th position.
We say that \( \text{span}(S) \) is \textit{generated} from \( S \). Clearly, \( \text{span}(S) \) is the smallest subsemimodule of \( M \) containing \( S \). A linearly independent subset \( B \subseteq M \) is said to be a \textit{basis} for \( S \subseteq M \) if and only if \( S \subseteq \text{span}(B) \).

Due to Theorems 4.3 and 4.4 these notions apply to the \( \mathcal{R} \)-semimodule \((T, \cap)\) and the \( \mathcal{R} \)-semimodule \((R_M, \sqcup)\).

**Example 4.6.** Continuing Example 4.5, we have that \( B = \{\alpha \to \alpha\} \) is a basis for the set of all types of \( I \), since \( \text{span}(B) = \{\bigcap_{i=1}^{n}(A_i \to A_i) \mid A_i \in T_0, n \geq 0\} \) (remember that \( \omega \) is the empty intersection, obtained here for \( n = 0 \)). The type in the basis \( B \) is given by the elaboration \( I = (\lambda x.(\{\alpha\}))([\alpha \to \alpha]) \) with \( ||I|| = 1 \).

It is important to realize that the notion of span allows the representation of infinite sets of typings of unbounded norm, even though the basis is bounded. Thus, in the example above, we have the elaboration \( I \) witnessing the typing of the basis \( B \) with \( ||I|| = 1 \), but the span of \( B \) contains typings which require unbounded norm, since the types \( \bigcap_{i=1}^{n}(A_i \to A_i) \) require elaborations of the form \( I_n = (\lambda x.([A_1, \ldots, A_n]))(A_1 \to A_1, \ldots, A_n \to A_n) \) with \( ||I_n|| = n \).

**Example 4.7.** Recall the term \( J = \lambda x.xI \) from Example 1.5. There is an infinite set of deep expansions of its standard principal type \( ((\alpha \to \alpha) \to \beta) \to \beta \), which can be written as \( \{((\bigcap_{i=1}^{n}(\alpha_i \to \alpha_i) \to \beta) \to \beta \mid n > 0\} \). There is no \textit{finite} basis for the set of all types \( T_J \) of \( J \). But, with \( B_n = \{((\bigcap_{i=1}^{n}(\alpha_i \to \alpha_i) \to \beta) \to \beta) \) (for \( n > 0 \)) we have \( B_n \subset \text{span}(B_{n+1}) \) (use multiplication with, e.g., \( r_{n+1} \) mapping \( \alpha_{n+1} \) to \( \alpha_n \)), and \( T_J = \bigcup_{n>0} B_n \), so the family of bases \( B_n \) may be said to approximate \( T_J \) in the limit.

Reflecting on Example 4.6 and Example 4.7 we may observe that the impossibility of representing all types of the term \( J \) by a single finite basis is related to the fact that there are infinitely many deep expansions needed to generate them (in the standard theory), whereas all types of \( I \) can be generated by top-level expansions followed by substitution. We may also observe that the types defining the sets \( B_n \) in Example 4.7 correspond to elaborations \( J_n \) of increasing norm, with \( ||J_n|| = n \) (recall Example 1.5). Finally, we may note that in both examples we have the standard principal types entering at some index (and norm), which happens to be \( n = 1 \) in both of these examples. These observations are motivating for the following development, because they suggest that it might be possible to \textit{approximate} standard principal typings systematically by constructing bases of subspaces in bounded norm and approximating under increasing norm. We shall see, though, that doing so in general in a \textit{finitary} and \textit{canonical} way allowing for an \textit{algorithmic} approach requires more work. This challenge will be taken up in Section 5.

Before we proceed, observe that bases can always be assumed to be given as subsets of \( T_0 \). For consider a non-strict type \( \sigma = A_1 \cap \cdots \cap A_n \). We can assume w.l.o.g. that \( \sigma \) has componentwise disjoint variables, \( \text{TV}(A_i) \cap \text{TV}(A_j) = \emptyset \) for \( i \neq j \) (if not, we could generate \( \sigma \) from such a disjoint-variable intersection type by instantiation, and the rule \((\cap)\) allows us to infer such a type whenever we can infer \( \sigma \)). Clearly, anything we can generate from \( B = \{\sigma\} \) we can also generate from \( B_0 = \{A_1, \ldots, A_n\} \), i.e., we have \( \text{span}(B) \subseteq \text{span}(B_0) \). But the reverse is also true, because we have \( A_i = r_i \sigma \) where \( r_i \) is the identity on \( \text{TV}(A_i) \) and maps other variables to \( \omega \) (\( r_i \) thereby “nulls out” other types than \( A_i \), recalling that the equation \( \tau \to \omega = \omega \) is implemented by application of substitutions). Therefore, the following definition will be useful in the sequel.

**Definition 4.8 (Strict span).** For \( S \subseteq T_0 \) let \( \text{span}_0(S) = \text{span}(S) \cap T_0 \).

It is also useful to notice that, whenever we have a \textit{strict} type \( A \in T_0 \) with \( A = r_1 B_1 \cap \cdots \cap r_n B_n \) \((r_1 \in \mathcal{R}, B_i \in T_0, i = 1 \ldots n)\), then we must in fact have \( A = r_j B_j \) for some \( j \in \{1, \ldots, n\} \) and \( r_j \in \mathcal{R}_0 \). It follows that a strict type from \( T_0 \) is in the span of a set of strict types if and only if it is a strict instance of one of those types.
5 COMPACTNESS AND BASES

In this section we develop the first main technical result of the paper: for an arbitrary term M and a dimensional bound n ≥ 0, there exists a unique (up to renaming) finite basis ℬ_{M,n} spanning all possible typings in dimension n. This result is shown in two main steps. First (Section 5.1) we introduce the key notion of compactness, a strongly irrelevant form of elaborations, and show that all types of n-dimensional compact elaborations are spanned by finite compact bases ℬ_{M,n} (Theorem 5.10). Then (Section 5.2), by adapting the technique of filtration of Dudenhefner and Rehof [2017c], we can show that, in fact, all n-dimensional typings are spanned by ℬ_{M,n} and that ℬ_{M,n} is unique (up to renaming) with this property (Theorem 5.20). Finally (Section 5.3), we collect a number of results concerning the relation between compactness, principality and approximation, which will be used in Section 7.

5.1 Compactness

We seek a finitary and canonical representation of all types of a given term witnessed by elaborations, combining two different notions of “being a useful subformula in a derivation”. The property of compactness is new, but tightness was implicitly used by Dudenhefner and Rehof [2017c] in the sense that the filtrations used there ensure the property.

Definition 5.1 (Decoration Types). The set of types T(P) ⊆ ℰ₀ appearing in the decoration sets of an elaboration P are referred to as the decoration types of P, in detail: T(⊥⟨S⟩) = S, T(x⟨S⟩) = S, T((λx.P)(S)) = T(P) ∪ S, T((PQ)(S)) = T(P) ∪ T(Q) ∪ S.

Definition 5.2 (Tightness, Compactness). A strict subformula of a type σ ∈ ℰ is a strict type A ∈ ℰ₀ such that A is a subformula of σ. An elaboration P is called tight if and only if every strict subformula of a decoration type in P is itself a decoration type in P (i.e., the set T(P) is closed under strict subformulae), and P is called a compact elaboration of M if and only if P ∈ ℐₘ and P is tight.

Compactness (the combination of relevance and tightness) captures a notion of strongly irrelevant elaborations, combining two different notions of “being a useful subformula in a derivation”. The property of compactness is new, but tightness was implicitly used by Dudenhefner and Rehof [2017c] in the sense that the filtrations used there ensure the property.

Example 5.3. Consider the following elaborations of P₁ of I = λx.x and Q of K = λx.λy.x:

\[
\begin{align*}
P₁ &= (λx.(α → α))((α → α)) \quad \text{compact} \\
P₂ &= (λx.((α → α))(((α → α) → (α → α)))) \quad \text{relevant, not tight} \\
P₃ &= (λx.((α → α)))((α → β) → α) \quad \text{not relevant, not tight} \\
Q &= (λx.λy.(α → α))((α → α) → α) \quad \text{not relevant, tight}
\end{align*}
\]

The elaboration P₂ is not tight, because it has “unnecessary structure” in the form of a strict subformula, α, which is not a decoration type.

Let the subsets Cₘ, Cₘ[ₙ] ⊆ ℰ₀ of strict types witnessed by compact elaborations be given by

\[
\begin{align*}
Cₘ[ₙ] &= \{A ∈ ℰ₀ \mid ∃P, Γ ⊢ P : A \land [P] = M \land ||P|| ≤ n \land P \text{ is compact}\} \\
Cₘ &= \bigcup_{n ≥ 0} Cₘ[ₙ]
\end{align*}
\]

Definition 5.4. Let V be a set of type variables. An element r ∈ ℐₚ is called a renaming on V, if r maps type variables in V to type variables and is injective on V. For σ ∈ ℰ we say that r is a renaming on σ, if r is a renaming on TV(σ).
Lemma 5.5. The types in $C_M[n]$ are of bounded size.

Proof. On types $A \in T_0$ define the measure $d$ by induction on $A$: $d(\alpha) = 0$, $d(\omega) = 0$, $d(\sigma \rightarrow A) = 1 + \max\{d(\sigma), d(A)\}$, and $d(A_1 \cap \cdots \cap A_n) = \max\{d(A_i) \mid i = 1 \ldots n\}$, $n \geq 1$. There are at most $|M| \cdot n$ distinct decoration types in an elaboration $P$ with $|P| \leq n$, where $|M|$ denotes the size of the term $M$. Hence, by compactness there can be at most a bounded number, say, $m$, of strict subformulae of any type $A \in C_M[n]$. It follows that there is a bound $D$ such that for any $A \in C_M[n]$ one has $d(A) \leq D$. Since each intersection $A_1 \cap \cdots \cap A_k$ has at most $m$ components (so $k \leq m$ for all $k$) the syntax trees of types in $C_M[n]$ have bounded branching and bounded depth.

Definition 5.6 (Relation $\sqsubseteq_{\mathcal R_0}$). Let the relation $\sqsubseteq_{\mathcal R_0} \subseteq C_M \times C_M$ be defined by $A \sqsubseteq_{\mathcal R_0} B$ if and only if there exists $r \in \mathcal R_0$ such that $A = rB$ and $r$ is not a renaming on $B$.

Lemma 5.7. The sets $C_M[n]$ do not contain any infinite ascending chains wrt $\sqsubseteq_{\mathcal R_0}$.

Proof. If $A \sqsubseteq_{\mathcal R_0} B$, then either $A$ has a branch with larger depth than the corresponding branch in $B$, or two or more variables in $B$ are mapped to the same type variable or to $\omega$. So, whenever $A \sqsubseteq_{\mathcal R_0} B$, either a branch in the syntax tree of $A$ is deeper than the corresponding branch in $B$, or $A$ has fewer type variables than does $B$. Since the former measure is upwards bounded (Lemma 5.5), and the latter measure is bounded from below by 0, the claim follows.

Define the equivalence relation $\sim$ on $T_0$ by setting $A \sim B$ iff there exists a renaming $r$ on $A$ such that $rA = B$. Let $[A]_\sim$ denote the induced equivalence classes. There is a choice function $f$ mapping $[A]_\sim$ to some canonical representative $f([A]_\sim) \in [A]_\sim$. For $X \subseteq T_0$ let $\mathcal B_X = \{f([A]_\sim) \mid A \in X\}$.

Definition 5.8 (Basis $\mathcal B_{M,n}$). For a given term $M$ and $n \geq 0$ define $\mathcal B_{M,n} = \mathcal B_{X^*}$, where $X^*$ is the set of $\sqsubseteq_{\mathcal R_0}$-maximal elements in $C_M[n]$, i.e., $X^* = \{A \in C_M[n] \mid \#B \in C_M[n], A \sqsubseteq_{\mathcal R_0} B\}$.

For $X \subseteq T_0$, $A \in T_0$, write $X < A$ iff $\forall B \in X. \forall r \in \mathcal R_0. A = rB \Rightarrow r$ is a renaming on $B$.

So one has $X < A$ if and only if $X$ does not contain any types generalizing $A$ other than renamings. For $Y \subseteq T_0$, write $X < Y$ iff $\forall B \in X. Y < B$.

Lemma 5.9. Let $X, Y \subseteq T_0$ with $Y \subseteq \text{span}_0(X)$. The set $\mathcal B_X$ is a basis for $Y$ if and only if $\text{span}_0(X^*) < X$.

Proof. See Section 10.

Theorem 5.10 (Compact Basis). For all $M$ and $n \geq 0$, the set $\mathcal B_{M,n}$ is a finite basis for $C_M[n]$.

Proof. Fix $\mathcal B_{M,n} = \mathcal B_{X^*}$, where $X^*$ is the set of $\sqsubseteq_{\mathcal R_0}$-maximal elements in $C_M[n]$. We first show that $\text{span}_0(X^*) < X^*$. So let $A \in X^*$ and $B \in \text{span}_0(X^*)$ with $A = rB$ for some $r \in \mathcal R_0$. Since $B \in \text{span}_0(X^*)$ we have $B = sA'$ for some $A' \in X^*$. Then $A = r \circ sA'$. Since both $A$ and $A'$ are members of $X^*$ it follows from the definition of $X^*$ that $r \circ s$ is a renaming on $A'$, hence $r$ is a renaming on $sA' = B$, thereby showing $\text{span}_0(X^*) < A$ for any $A \in X^*$. From Lemma 5.9 we can conclude that $\mathcal B_{X^*}$ is a basis for $\text{span}_0(X^*)$. To see that $C_M[n] \subseteq \text{span}_0(X^*)$, assume $B \in C_M[n] \setminus X^*$. By Lemma 5.7 there exists $A \in X^*$ such that $B \sqsubseteq_{\mathcal R_0} A$. This shows $B \in \text{span}_0(X^*)$. It remains to show that $\mathcal B_{X^*}$ is finite. But that follows from Lemma 5.5: Since the size of elements in $X^*$ is bounded, there is only a finite set of distinct type shapes (technically, domains of trees representing types) into which all elements of $X^*$ can be mapped (by mapping all variables to a single variable). The total number of distinct leaf variable occurrences in the set of shapes is finite. Hence, any actual type can be represented under renaming from a finite set of types over a fixed finite set of fresh variables. The equivalence class representatives in $\mathcal B_{X^*}$ can thus be represented by renaming into this finite set.
5.2 Filtration and Basis Theorem

We are now in a position to prove one of the main technical results of the paper: the compact bases $\mathcal{B}_{M,n}$ span all possible typings of a term in dimension $n$ (Theorem 5.20 below). The proof is by reduction to compact elaborations. From Theorem 5.10 we know that $\mathcal{B}_{M,n}$ spans all compact elaborations with norm bounded by $n$. We now show that an arbitrary elaboration can be transformed to a compact one such that the type of the original elaboration is in the span of its type under non-increasing norm. The result is then obtained using Theorem 5.10.

Technically, we adapt the notion of filtration to the relevant $(\mathcal{I}_P)$ setting. Filtrations were used by Dudenhefner and Rehof [2017c] to bound the width of intersection, and by Dudenhefner and Rehof [2017a] to establish a necessary condition for principal inhabitation in simple types. Intuitively, for a set $X \subseteq \mathcal{T}_0$ of strict types the filtration $\mathcal{F}_X : \mathcal{T} \rightarrow \mathcal{T}$ removes subformulae of its input type that are not supported by $X$. In our application, $X$ is chosen to be the set of decoration types in an elaboration of a given term, eliminating subformulae that are not necessary to type the term. We show that applying filtration to a relevant elaboration results in a compact elaboration.

**Definition 5.11 (Filtration $\mathcal{F}_X$).** Let $X \subseteq \mathcal{T}_0$ be a set of strict types. We define the filtration $\mathcal{F}_X : \mathcal{T} \rightarrow \mathcal{T}$ by

$$\mathcal{F}_X(\beta) = \beta$$

$$\mathcal{F}_X(A_1 \cap \ldots \cap A_n) = \mathcal{F}_X(A_1) \cap \ldots \cap \mathcal{F}_X(A_n)$$

$$\mathcal{F}_X(\sigma \rightarrow B) = \left\{ \begin{array}{ll} \mathcal{F}_X(\sigma) \rightarrow \mathcal{F}_X(B) & \text{if } \{\sigma \rightarrow B, A_1, \ldots, A_n, B\} \subseteq X \text{ where } \sigma = A_1 \cap \ldots \cap A_n \\ \alpha_{\sigma \rightarrow B} & \text{otherwise} \end{array} \right.$$  

We define $\mathcal{F}_X(P)$ pointwise on decorations, observing that $\mathcal{F}_X$ maps strict types to strict types.

**Lemma 5.12.** For any set of strict types $X \subseteq \mathcal{T}_0$, the filtration $\mathcal{F}_X$ is injective.

The following Example 5.13 shows the elimination of unnecessary subformulae in a given elaboration. It also hints at useful properties that an elaboration $P$ has under filtration $\mathcal{F}_{\mathcal{T}(P)}$. Namely that, $\mathcal{F}_P$ preserves relevance, ensures tightness, does not increase norm, and the original elaboration can be obtained by a single substitution.

**Example 5.13.** Let $A = \alpha \rightarrow \alpha$, $P = (\lambda x.([A])\langle[A \rightarrow A]\rangle) \in \mathcal{R}$, and $X = \mathcal{T}(P) = \{A, A \rightarrow A\}$. We have $\mathcal{F}_X(A) = \alpha_A$, $\mathcal{F}_X(A \rightarrow A) = \alpha_A \rightarrow \alpha_A$, and $\mathcal{F}_X(P) = (\lambda x.([\alpha_A])\langle[\alpha_A \rightarrow \alpha_A]\rangle) \in \mathcal{R}$. Observe that $\mathcal{F}_X(P)$ is tight (whereas $P$ is not because the subformula $\alpha$ is not a decoration type), $\mathcal{F}_X(P)$ has norm 1, and for the substitution $S(\alpha_A) = A$ we have $S(\mathcal{F}_X(P)) = P$.

Complementary to the above example, if an elaboration is tight, then the corresponding filtration behaves on it as the identity (Lemma 5.14).

**Lemma 5.14.** For $P \in \mathcal{R}$, if $P$ is tight, then $\mathcal{F}_{\mathcal{T}(P)}(P) = P$.

**Proof.** Since $P$ is tight, the case $\mathcal{F}_{\mathcal{T}(P)}(\sigma \rightarrow B) = \alpha_{\sigma \rightarrow B}$ cannot occur. In all other cases $\mathcal{F}_{\mathcal{T}(P)}$ is the identity. □

Filtrations preserve relevance (Lemma 5.15), ensure tightness (Lemma 5.16), and do not increase norm (Lemma 5.17).

**Lemma 5.15.** For $P \in \mathcal{R}$ we have $\mathcal{F}_{\mathcal{T}(P)}(P) \in \mathcal{R}$.

**Proof.** We show a stronger claim: for any $X \supseteq \mathcal{T}(P)$ we have $\mathcal{F}_X(P) \in \mathcal{R}$. This is seen by induction on the derivation of $P \in \mathcal{R}$. □

Proc. ACM Program. Lang., Vol. 3, No. POPL, Article 8. Publication date: January 2019.
LEMMA 5.16. For $P \in \mathcal{R}$ the elaboration $\mathcal{F}_{T}(P)$ is tight.

PROOF. Let $Q = \mathcal{F}_{T}(P)$. We show for any $A \in T(Q)$ that for any strict subformula $C$ occurring in $A$ we have $C \in T(Q)$. This follows by induction in the size of the syntax tree of $A$. □

LEMMA 5.17. For $P \in \mathcal{R}$ we have $\|\mathcal{F}_{T}(P)\| \leq \|P\|$.

PROOF. Since $\mathcal{F}_{T}(P)$ maps strict types to strict types, the claim immediately follows. □

Let us combine the above Lemmas 5.15, 5.16, and 5.17 in the following Corollary 5.18 establishing compactness under non-increasing norm.

COROLLARY 5.18. For $P \in \mathcal{R}$ the elaboration $\mathcal{F}_{T}(P)$ is compact and $\|\mathcal{F}_{T}(P)\| \leq \|P\|$.

The following Lemma 5.19 shows that filtrations may be reversed by a single strict substitution.

LEMMA 5.19. For $P \in \mathcal{R}$, there exists a strict substitution $s \in \mathcal{R}_0$ such that $s(\mathcal{F}_{T}(P)) = P$.

PROOF. Let $\{\alpha_{A_1}, \ldots, \alpha_{A_m}\}$ be the type variables introduced in $\mathcal{F}_{T}(P)$. Define $s(\alpha_{A_i}) = A_i$ for $i = 1 \ldots m$. For $A \in T(P)$ one shows that $s(\mathcal{F}_{T}(P)(A)) = A$ by induction in the size of the syntax tree of $A$. □

By Theorem 5.10 we know that the compact bases $\mathcal{B}_{M,n}$ span $C_{M}[n]$, the types witnessed by the subset of compact elaborations bounded by norm $n$. The following Theorem 5.20 is a central result of the paper, showing that the entire set of types which can be obtained by any (also non-compact) $n$-dimensional elaborations is contained in the subsemimodule spanned by $\mathcal{B}_{M,n}$. We thereby obtain a finite representation of the unbounded subspaces generated from all $n$-dimensional typings of a term in terms of standard notions from abstract linear algebra. Moreover, this representation is unique and can therefore be taken as canonical for the $n$-dimensional systems.

Consider therefore the sets $T(M, n)$ (resp. $T_0(M, n)$) of all types (resp. strict types) of $M$ in dimension $n$: $T(M, n) = \{\sigma \in T | \exists \Gamma. \Gamma \vdash_{R,n} M : \sigma\}$ and $T_0(M, n) = \{A \in T_0 | \exists \Gamma. \Gamma \vdash_{R,n} M : A\}$.

THEOREM 5.20 (Basis). Given $M$ and $n \geq 0$, let $\mathcal{B}_{M,n}$ be as given by Definition 5.8. We have:

1. $T_0(M, n) = \text{span}_0(\mathcal{B}_{M,n})$
2. $T(M, n) \subseteq \text{span}(\mathcal{B}_{M,n})$
3. $\mathcal{B}_{M,n}$ is unique up to renaming among bases in $T_0(M, n)$

PROOF. To prove claim (1), consider first the inclusion $\subseteq$. Assume $A \in T_0(M, n)$ so we have $\Gamma \vdash_{R} P : A$ for some elaboration $P$ such that $\|P\| \leq n$. Define $Q = \mathcal{F}_{T}(P)$. By Corollary 5.18 $Q$ is compact with $Q : \mathcal{F}_{T}(P)(A)$ and $\|Q\| \leq n$. By Lemma 5.19 there exists a strict substitution $s \in \mathcal{R}_0$ such that $s(Q) = P$, therefore $s(\mathcal{F}_{T}(P)(A)) = A$. By Theorem 5.10 we have $\mathcal{F}_{T}(P)(A) \in \text{span}(\mathcal{B}_{M,n})$, therefore $A = s(\mathcal{F}_{T}(P)(A)) \in \text{span}(\mathcal{B}_{M,n})$. Because $A \in T_0$, we have $A \in \text{span}_0(\mathcal{B}_{M,n})$. To see the inclusion $\supseteq$, assume $A \in \text{span}_0(\mathcal{B}_{M,n})$, so for some $B \in \mathcal{B}_{M,n}$ we have, for some $s \in \mathcal{R}_0$ and $P : B$, that $\|P\| \leq n$ and $A = s(B)$, then $s(P) : A$ with $\|sP\| \leq n$, which shows the claim.

To prove claim (2), assume $\sigma \in T(M, n)$ with $\Gamma \vdash_{R,n} M : \sigma$. Write $\sigma = A_1 \cap \cdots \cap A_m$, where $m \leq n$ and $A_i \neq A_j$ for $i \neq j$. Then, by inversion of ($\cap$), we have $\Gamma \vdash_{R,n_i} M : A_i$ with $n_i \leq n$. By (1) we have $A_i \in \text{span}_0(\mathcal{B}_{M,n})$ for $i = 1 \ldots m$, hence $\sigma \in \text{span}(\mathcal{B}_{M,n})$.

To prove claim (3), consider any linearly independent subset $\mathcal{B} \subseteq T_0(M, n)$ with the property $T_0(M, n) \subseteq \text{span}_0(\mathcal{B})$. Let $B \in \mathcal{B}$. Since $B \in T_0(M, n)$, we have $B \in \text{span}_0(\mathcal{B}_{M,n})$, so $B = sA$ for some $A \in \mathcal{B}_{M,n}$ and $s \in \mathcal{R}_0$. Since $A \in T_0(M, n)$ we have $A \in \text{span}_0(\mathcal{B})$, so $A = rB$ for some $B \in \mathcal{B}$ and $r \in \mathcal{R}_0$. Hence $B = sA = (s \circ r)B$. By linear independence of $\mathcal{B}$, we must have $B = B'$ with $s \circ r$ the identity on $B'$, hence $s$ is a renaming on $A$. This shows that a renaming of $B$ can be found in $\mathcal{B}_{M,n}$. The dual inclusion (modulo renaming) of $\mathcal{B}_{M,n}$ in $\mathcal{B}$ is shown analogously. □
\[ n = 0 : \]
\[ \omega \]
\[ 0_D = (\lambda x. \lambda y. (y \ x ([|]]) \ x ([|])) \ z ([|]) \]
\[ n = 1 : \]
\[ (\alpha \rightarrow \alpha \rightarrow \delta) \rightarrow \delta \]
\[ (\alpha \rightarrow \omega \rightarrow \delta) \rightarrow \delta \]
\[ (\omega \rightarrow \alpha \rightarrow \delta) \rightarrow \delta \]
\[ (\lambda. \lambda y. (y \ x ([|]) \ x ([|])) \ z ([|]) \]
\[ n = 2 : \]
\[ (\alpha \rightarrow \beta \rightarrow \delta) \rightarrow \delta \]
\[ (\alpha \cap \beta \rightarrow \alpha \cap \beta \rightarrow \delta) \rightarrow \delta \]
\[ (\alpha \cap \beta \rightarrow \omega \rightarrow \delta) \rightarrow \delta \]
\[ (\omega \rightarrow \alpha \cap \beta \rightarrow \delta) \rightarrow \delta \]
\[ (\lambda. \lambda y. (y \ x ([|])) \ x ([|])) \ z ([|]) \]
\[ n = 3 : \]
\[ (\alpha \cap \beta \rightarrow \gamma \rightarrow \delta) \rightarrow \delta \]
\[ (\alpha \cap \beta \rightarrow \alpha \cap \gamma \rightarrow \delta) \rightarrow \delta \]
\[ (\alpha \rightarrow \beta \cap \gamma \rightarrow \delta) \rightarrow \delta \]
\[ (\alpha \cap \beta \cap \gamma \rightarrow \alpha \cap \beta \cap \gamma \rightarrow \delta) \rightarrow \delta \]
\[ (\alpha \cap \beta \cap \gamma \rightarrow \omega \rightarrow \delta) \rightarrow \delta \]
\[ (\omega \rightarrow \alpha \cap \beta \cap \gamma \rightarrow \delta) \rightarrow \delta \]
\[ (\lambda. \lambda y. (y \ x ([|])) \ x ([|])) \ z ([|]) \]

Fig. 2. Bases \( B_{D,n} \) and elaborations, for \( n = 0, 1, 2, 3 \)

With Theorem 5.20 we have obtained a finite and canonical representation of the semimodular subspaces of \((T, \cap)\) spanned by the set of all \( n \)-dimensional typings. Moreover, we obviously have \( B_{M,n} \subseteq \text{span}_0 (B_{M,n+1}) \) for all \( n \geq 0 \). These properties together underlie the possibility of approximating type inference of principal type information in a mathematically ordered way, as we shall see in the following. Besides providing a feel for what bases \( B_{M,n} \) may look like, the following Example 5.21 illustrates a number of important aspects in this regard.

Example 5.21. Consider the term \( D = (\lambda x. \lambda y. (y \ x) \ x) \ z \). By inspecting the structure of possible compact elaborations we can determine \( B_{D,n} \) for \( n \geq 0 \). In Figure 2 we show the types in \( B_{D,n} \) (to the left) together with their witnessing compact elaborations (to the right, decoration sets only partially shown), for \( n = 0, 1, 2, 3 \). Notice that, for this term \( D \), the standard principal type (which is \( (\alpha \rightarrow \beta \rightarrow \delta) \rightarrow \delta \)) enters the basis \( B_{D,n} \) at \( n = 2 \). The variable \( z \), being the argument in a \( \beta \)-redex of a non-linear \( \lambda \)-abstraction, evidently pumps up the dimension \( n \) by acting as an “accumulator” of decoration types\(^5\). For our term \( D \) we see the principal types of the approximate normal forms at \( n = 1 \): \( \lambda y. (y \ x) \perp \) (typed at \( (\alpha \rightarrow \omega \rightarrow \delta) \rightarrow \delta \)) and \( \lambda y. (y \perp) x \) (typed at \( (\omega \rightarrow \alpha \rightarrow \delta) \rightarrow \delta \)), which are incomparable wrt. \( \subseteq \perp \) (resp. \( \subseteq_\omega \)), corresponding to linear independence of the types. The approximant \( \lambda y. (y \perp) \perp \) is present as an instance: we have \((\omega \rightarrow \omega \rightarrow \delta) \rightarrow \delta \subseteq_\omega (\alpha \rightarrow \alpha \rightarrow \delta) \rightarrow \delta \). The smallest approximant, \( \perp \), is as always present at \( n = 0 \). The other basis types at higher \( n \) are either deep expansions (explained right before Example 1.5) of the standard principal type, or they are special cases of such (the more general expanded forms in those cases being prevented from entering yet by the given bound \( n \)). Linear independence of the bases forces the different structures of intersections (\( \omega \) being the empty one) to be placed at “orthogonal” positions in the

\(^5\)This phenomenon is systematic. By Dudenhefner and Rehof [2017b, Proposition 25] we know that principal elaborations of \( \beta \)-normal forms have dimension 1, but generally (for non-normal forms) dimension can be arbitrarily high (cf. also Section 2). By subject reduction in bounded dimension [Dudenhefner and Rehof 2017b, Theorem 19] we know that dimension is non-increasing (and eventually decreases) under \( \beta \)-reduction.
basis types. Coming back to Example 1.2 in this light, it can similarly be seen that the standard principal type of the term $\omega \,(\omega \,x)$ enters at $n = 4$. A central topic in the remainder of this paper will be to prove that standard principal types (insofar as they exist for the given term $M$) and those of approximate normal forms will always be in $B_{M,n}$ for sufficiently high $n$, and that we can algorithmically recognize these situations when they occur under increasing bound $n$. Finally, we can see from this example that we can get combinatorial growth of the size of $B_{M,n}$, especially as we go beyond the standard principal dimension, i.e., the smallest $n_0$ such that the standard principal type enters the basis $B_{M,n_0}$. In many applications one could envision of the theory developed here, one would be able to limit search to that dimensional bound $n_0$. From this perspective it is important that we are able to recognize this index together with the standard principal type of $M$ as it enters $B_{M,n_0}$.

5.3 Compactness, Principality, and Approximants

We end the first part of our technical development by collecting a number of results relating compactness to the standard notions of principality and approximation in the $\lambda$-calculus, which will be used in the remainder of the paper, where we proceed to consider type inference of principal type information and its relation to the theory of approximants. In particular, all results collected here will be used in Section 7 to prove completeness of \textit{approximation by type inference} (Theorem 7.6). A key property is Theorem 5.26 which shows that compactness is preserved under subject expansion. Since, in addition, approximate normal forms have principal pairs witnessed by compact elaborations (Lemma 5.24), we can “pull back” compactness properties for arbitrary terms from their approximate normal forms. In the sequel, the following notation will be useful. We write $\Gamma \vdash_R M \rightarrow P : [A_1, \ldots, A_n]$ (and alternatively, $\Gamma \vdash_R M \rightarrow P : A_1 \cap \cdots \cap A_n$) if and only if $\Gamma \vdash_R P : [A_1, \ldots, A_n]$ with $[P] \equiv M$.

We use the term \textit{principal pair} $(\Gamma, A)$ for approximate normal forms $F \in \mathcal{A}$ in the (exact) sense of Coppo et al. [1980, Definition 8] (see also van Bakel [2011, Definition 10.6]). Following Coppo et al. [1980], for approximate normal forms $F \in \mathcal{A}$ one can define the principal pair $(\Gamma, A)$ of $F$ by induction in $F$ (note that principal pairs also type $F$ in the relevant system). We provide (Definition 5.22) an elaborated version of this definition, thereby defining the \textit{principal elaboration} of $F$, denoted $\Pi_F$, such that $\Gamma \vdash_R F \mapsto \Pi_F : A$.

\textit{Definition 5.22 (Principal Elaboration).} We define the function $\Pi$ mapping approximate normal forms $F \in \mathcal{A}$ to pairs $(\Gamma, P)$ such that $\Gamma \vdash_R P : A$ and $(\Gamma, A)$ is the principal pair for $F$ in the (exact) sense of Coppo et al. [1980, Definition 8].

\[
\begin{align*}
\Pi(\bot) & \equiv (\emptyset, \bot) \\
\Pi(x) & \equiv (\{x : \alpha\} \cup \{\sigma \mapsto x(\alpha)\}, \alpha \text{ fresh}) \\
\Pi(\lambda x . F) & \equiv (\Gamma \uplus \{\sigma \mapsto \Pi(F)\}, \text{ if } x \notin \text{ FV}(F)) \quad \Pi(F) = (\Gamma \cup \{x : \sigma\}, P : A) \\
\Pi(\Pi(x) F_1 \cdots F_n) & \equiv (\Pi(F), \Pi(F_1) \cdots \Pi(F_n)) \\
\end{align*}
\]

\[
\begin{align*}
\Gamma = \bigcap_{i=1}^n \Gamma_i \cap \{x : A_i \rightarrow \cdots \rightarrow A_n \rightarrow \alpha\} \\
P \equiv (\Pi(F_1) A_1 \rightarrow \cdots \rightarrow A_n \rightarrow \alpha) \quad (P_1) A_2 \rightarrow \cdots \rightarrow A_n \rightarrow \alpha \\
\end{align*}
\]

when $\Pi(F_i) = (\Gamma_i, P_i : A_i)$, $\alpha$ fresh

For $F \in \mathcal{A}$ and $\Pi(F) = (\Gamma, P)$, we let $\Pi_F \equiv P$, and we refer to $\Pi_F$ as the \textit{principal elaboration} of $F$. Notice that $\|\Pi_F\| \leq 1$.

The partial order $\sqsubseteq_{\bot} \subseteq \Lambda_\bot \times \Lambda_\bot$ is the smallest preorder satisfying: $\bot \sqsubseteq_{\bot} M$, $P \sqsubseteq_{\bot} Q \rightarrow \lambda x . P \sqsubseteq_{\bot} \lambda x . Q$, and $(P \sqsubseteq_{\bot} P' \land Q \sqsubseteq_{\bot} Q') \rightarrow (PQ) \sqsubseteq_{\bot} (P'Q')$. For $F \in \mathcal{A}$ and $M \in \Lambda$, if $F \sqsubseteq_{\bot} M$, Proc. ACM Program. Lang., Vol. 3, No. POPL, Article 8. Publication date: January 2019.
then \( F \) is called a direct approximant of \( M \). The set \( \mathcal{A}(M) \) of approximants of \( M \) is defined as 
\[ \mathcal{A}(M) = \{ F \in \mathcal{A} \mid \exists M'. M =_\beta M' \land A \subseteq M' \} \]. In fact, \( \beta \)-reduction and direct approximation suffices to define the set of approximants (Lemma 5.23).

**Lemma 5.23.** For any \( F \in \mathcal{A}(M) \) there exists a term \( M' \) such that \( F \subseteq M' \) and \( M \rightarrow_\beta M' \).

**Proof.** First we show that if \( M \rightarrow_\beta M' \) and \( F \subseteq M \) such that \( F \in \mathcal{A} \), then \( F \subseteq M' \). Proceed by induction on the number of reduction/expansion steps using the Church-Rosser theorem. \( \square \)

**Lemma 5.24.** For every \( F \in \mathcal{A} \), \( \Pi_F \) is compact.

**Proof.** Induction on the definition of \( \Pi_F \). \( \square \)

**Lemma 5.25.** If \( \Gamma \uparrow_R M \mapsto P : \sigma \) with \( P \) compact and \( M \subseteq M' \), then there exists a compact \( Q \) such that \( \Gamma \uparrow_R M' \mapsto Q : \sigma \).

**Proof.** Induction on the definition of \( \subseteq_\cdash \). For the base case \( M \equiv \bot \) we have \( P \equiv 0_\bot \) and \( \sigma = \omega \), and in this case we take \( Q \equiv 0_{M'} \). \( \square \)

**Theorem 5.26 (Compact Subject Expansion).** Assume \( \Gamma \uparrow_R P \mapsto P : \sigma \) with \( P \) compact, and \( Q \rightarrow_\beta P \). Then there exists compact \( Q \) such that \( \Gamma \uparrow_R Q \mapsto Q : \sigma \).

**Proof.** For an elaboration \( P : \sigma \) let us refer to the type \( \sigma \) together with the types in \( \Gamma_P \) as the observable types in \( P \). An elaboration \( P \) is said to have the dual subformula property if and only if every strict subformula of an observable type in \( P \) is also a decoration type in \( P \). Notice (by induction on derivations) that on a relevant elaboration every observable type is a decoration type, and for such elaborations tightness therefore implies the dual subformula property.

Below we write \( T(\sigma) \) to indicate an elaboration with decoration set determined by the type \( \sigma \), and \( P_x[T_{(i)}(\sigma_i)] \) denotes the result of replacing each occurrence \( x_{(i)} \ldots \) of \( x \) in \( P \) by \( T_{(i)}(\sigma_i) \), for \( i = 1 \ldots k \).

Assume first that \( Q \rightarrow_\beta P \) by contraction of a top level \( \lambda I \)-redex, so \( Q \equiv (\lambda x.M)N \), \( x \in \text{FV}(M) \) and \( P \equiv M\{x := N\} \). Consider the case where \( \sigma = A \in \mathcal{T}_0 \), and we have a compact elaboration \( P \equiv P_x[T_{(i)}(\sigma_i)] \) \( x \in \text{FV}(P) \). It expands to the elaboration \( Q \) given by
\[
((\lambda x.P_x[x_{(i)}(\sigma_i)])(\prod_{i=1}^k (\sigma_i) \rightarrow A)) \prod_{i=1}^k (T_{(i)}(\sigma_i))(\prod_{i=1}^k (\sigma_i))(A)
\]

Every observable type in \( Q \) is observable in \( P \), and clearly any subformula appearing in a decoration set in \( P \) also appears as such in \( Q \). Therefore, since \( P \) has the dual subformula property, so has \( Q \). So \( Q \) satisfies the tightness condition with respect to observable types. Moreover, by the form of \( Q \) and \( x \in \text{FV}(M) \), \( Q \) is relevant, and any strict subformula of a decoration type in \( Q \) is also a subformula of a decoration type in \( P \). Hence the tightness condition for the remaining types in \( Q \) follows from tightness of \( P \). It follows that \( Q \) is a compact elaboration of \( Q \). For a \( \lambda K \)-redex where \( x \notin \text{FV}(M) \) we take the elaboration \( (\lambda x.P)(\omega \rightarrow A) \cdot_0 (\omega)(A) \).

The general case where \( \sigma \in \mathcal{T} \) follows from the observation that whenever \( Q_1 \) and \( Q_2 \) are compact, then so is \( Q_1 \sqcup Q_2 \). Finally, the claim is generalized to reduction in an arbitrary context \( C \) by induction on \( C \), using the above argument for the base case where \( C \) is the empty context. \( \square \)

### 6 Uniqueness of Principal Inhabitants

We now begin the second part of our technical development, in which we provide a computational realization of the preceding theory of semimodular bases.

In this section, we define the algorithm \( \mathcal{T} \) which, given a principal pair \((\Gamma, A)\), constructs an elaboration of the corresponding principal inhabitant in polynomial time. Algorithm \( \mathcal{T} \) is a key
ingredient not only to decide principal inhabitation but also to construct approximants of a given λ-term from inferred type information in Section 7. Per se, the principal inhabitation procedure is a direct inversion of the principal pair generation procedure of Coppo et al. [1980] and, therefore, is not a new result [Neergaard and Mairson 2004; Tsukada and Ong 2016]. However, what is new is the complexity analysis of principal inhabitation and its application to approximant synthesis in Section 7.

The ability to construct principal inhabitants in polynomial time appears surprising in light of undecidability of intersection type inhabitation in general. Our approach is based on the following weak balance property (cf. "balanced sequent" by Mints [1981]), that any principal pair satisfies by construction (Definition 5.22).

**Definition 6.1.** We say \((\Gamma, \tau)\) is weakly balanced, if each type variable \(\alpha\) occurs at most once positively and at most once negatively in \((\Gamma, \tau)\).

**Algorithm 1 (\(T(\Gamma, \tau)\)).** Given a type environment \(\Gamma\) and a type \(\tau\) (either strict or \(\omega\)), construct a tight elaboration \(F \in \mathcal{R}\) such that \(F : \tau\).

If \((\Gamma, \tau)\) is weakly balanced, then distinguish the following cases:

**Case** \((\Gamma, \omega)\): Return \(\bot(\omega)\).

**Case** \((\Gamma, \sigma \rightarrow A)\): Compute \(F = T(\Gamma \cup \{x : \sigma\}, A)\) where \(x\) is fresh. If \(\sigma \neq T_x(F)\) then fail. Otherwise, return \((\lambda x.F)((\sigma \rightarrow A))\).

**Case** \((\Gamma, \alpha)\) where \(A = \sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \alpha \in \Gamma(x)\) and each \(\sigma_i\) is strict or \(\omega\):

Let \(\Gamma'(x) = \Gamma(x) \setminus A\) and \(\Gamma'(y) = \Gamma(y)\) for \(y \neq x\). Compute \(F_i = T(\Gamma', \sigma_i)\) for \(i = 1 \ldots n\) and return \((x([\sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \alpha]) F_1)([\sigma_2 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \alpha]) \ldots F_n)([\alpha])\).

**Otherwise:** Fail.

By construction, if \(F = T(\Gamma, \tau)\), then the term \([F] \in \mathcal{A}\) is tight. Thus, \(T\) is a polynomial time algorithm.

Algorithm \(T\) is sound wrt. compact elaborations (Lemma 6.2) and, more importantly, complete wrt. principal inhabitants (Lemma 6.3). Both proofs are by routine induction.

**Lemma 6.2.** If \(F = T(\Gamma, A)\), then \(F \in \mathcal{R}\) and \(F\) is tight.

**Lemma 6.3 (Principal Inhabitant Reconstruction).** For any \(F \in \mathcal{A}\) having the principal pair \((\Gamma, A)\) we have \([T(\Gamma, A)] = F\).

An immediate consequence of Lemma 6.3 is uniqueness of principal inhabitants (this is "Torino folklore", communicated to the second author by M. Dezani in 2013).

**Corollary 6.4 (Principal Inhabitant Uniqueness).** If \(F, F' \in \mathcal{A}\) both have the principal pair \((\Gamma, A)\), then we have \(F = F'\).

A consequence of the ability to reconstruct principal inhabitants is polynomial time decidability of principal inhabitation (Theorem 6.5).

**Theorem 6.5 (Decidability of Principal Inhabitation).** Given a pair \((\Gamma, A)\), it is decidable in polynomial time whether there exists \(F \in \mathcal{A}\) such that \((\Gamma, A)\) is the principal pair of \(F\).

**Proof.** It suffices to compute \(F = [T(\Gamma, A)]\) and verify that the actual principal pair of \(F\), constructible in polynomial time by using Definition 5.22, is \((\Gamma, A)\) up to renaming. \(\Box\)

Principal inhabitant uniqueness is based on several restrictions including that principal inhabitants are necessarily \(\eta\)-long and \(\bot\) is the only inhabitant of \(\omega\). In general however, a type derivation
of $\Gamma \vdash_R F' : A$ such that $F' \in \mathcal{A}$, even if $(\Gamma, A)$ is a principal pair of some $F \in \mathcal{A}$, does not imply $F' \equiv_{\beta \perp} F$, as the following Example 6.6 shows. Here we refer to the standard $\beta \perp$-reduction on $\perp$-terms [van Bakel 2011, Section 2.1] given by adding the following rules to $\beta$-reduction: $\lambda x. \perp \rightarrow \perp \lambda x \perp$ and $\perp M \rightarrow \perp \perp$.

**Example 6.6.** Let $A = \omega \rightarrow \alpha$ and $\Gamma = \{ x : A \}$. Although $(\Gamma, A)$ is the principal pair of $\lambda y. x \perp$, the $\perp$-minimal term $x$ satisfies $\Gamma \vdash_R x : A$ and $\lambda y. x \perp \not\equiv_{\beta \perp} x$.

Still, we can narrow down the structure of $\perp \beta$-normal terms $F'$ with $\Gamma \vdash_R F' : A$ (Lemma 6.7). If $x \notin \text{FV}(F)$, then the term $\lambda x. F \perp$ is assigned the type $\omega \rightarrow A$, making $x$ and $\perp$ interchangeable wrt. relevant typability. Therefore, we may apply $\lambda x. F \perp \subseteq \perp \lambda x. F x \rightarrow_{\eta} F$. Hence, in the above Example 6.6 we can relate the two given terms by $\lambda y. x \perp \subseteq \perp \lambda y. x y \rightarrow_{\eta} x$.

**Lemma 6.7.** Let $\Gamma \vdash_R F' : A$ such that $F'$ is $\perp \beta$-normal and $\perp$-minimal. If $[\mathcal{T}(\Gamma, A)] = F$, then there exists $F''$ such that $F \subseteq F'' \rightarrow_{\eta} F'$.

**Proof.** We show a more general claim: let $\Gamma' \vdash_R F' : \tau$ such that $\tau$ is either strict or $\omega$, and $F' \in \mathcal{A}$ is $\perp$-minimal. If $\mathcal{T}(\Gamma, \tau) = F$ such that $\Gamma'(x) \subseteq \Gamma(x)$ for all $x$, then $[F] \subseteq F'' \rightarrow_{\eta} F'$ for some $F''$. We proceed by induction on the size of $F$.

Using the above Lemma 6.7 we will establish soundness (up to $\eta$-reduction) of approximant construction in Section 7. In fact, an inspection of the proof of Lemma 6.7 shows that $\perp$ is only expanded to variables relating $F \subseteq F''$. Therefore, no additional $\rightarrow_R \beta \perp$-reduces can arise, and the size of the term does not increase, thereby limiting the space of potential candidates for $F'$.

### 7 APPROXIMATION BY TYPE INFERENCE

We now complete our algorithmic realization of the theory of semimodular bases to achieve inference of principal typing information under dimensional bound. By combining automatic construction of compact bases $\mathcal{B}_{M,n}$ described in Section 5 with the principal inhabitation algorithm $\mathcal{T}$ from Section 6 we can recognize principal typings, and we can compute the approximants of a given $\lambda$-term $M$ by type inference under increasing bound.

We first extend the typability algorithm introduced by Dudenhefner and Rehof [2017c] to construct compact elaborations of $M$ (Algorithm $\mathcal{T}$). Subsequently, algorithm $\mathcal{T}$ is used to compute the finite basis $\mathcal{B}_{M,n}$ (Theorem 7.4). Feeding type information to the principal inhabitation algorithm $\mathcal{T}$ we obtain a sound and complete procedure for computing approximate normal forms of $M$ (Theorem 7.5 and Theorem 7.6).

**Algorithm 2 ($\mathcal{T}(M, d)$).** Given a $\lambda$-term $M$ and dimension $d$ we non-deterministically construct a corresponding elaboration in dimension $d$ as follows.

1. For each subterm of $M$ choose $l \in \{0, \ldots, d\}$ and decorate the subterm with $\langle [a_1, \ldots, a_l] \rangle$ where $a_i$ are fresh for $i = 1 \ldots l$, thereby constructing the elaboration $\hat{P}$.
2. For each type variable $\alpha$ occurring in $\hat{P}$ choose $l \in \{-1, 0, \ldots, d \cdot |M|\}$ and define $\hat{\tau} \alpha^\gamma = \begin{cases} \alpha, & \text{if } l = -1 \\ (\beta_1 \cap \ldots \cap \beta_l) \rightarrow \beta_0, & \text{otherwise where } \beta_0, \ldots, \beta_l \text{ are fresh} \end{cases}$
3. Construct the set of constraints $C = \{ \alpha \doteq \hat{\tau} \alpha^\gamma | \alpha \text{ occurs in } \hat{P} \}$.
4. For each type variable $\alpha$ such that $\hat{\tau} \alpha^\gamma = (\beta_1 \cap \ldots \cap \beta_l) \rightarrow \beta_0$ and each $j = 0 \ldots l$ choose $\alpha'$ occurring in $\hat{P}$ and add $\alpha' \doteq \beta_j$ to $C$.
5. Starting with the elaboration $\hat{P}$, extend the set of constraints $C$ using the following steps:
   - **Case $\hat{Q} : \omega$:** verify that $\hat{Q} = 0_{[\hat{Q}]}$, otherwise fail.
Case $\hat{Q} : \alpha_1 \cap \ldots \cap \alpha_l$ and $l > 0$: choose a decomposition $\hat{Q}_1 : \alpha_1, \ldots, \hat{Q}_l : \alpha_l$ such that

$$\hat{Q} = \bigsqcup_{l=1}^l \hat{Q}_l.$$ Proceed recursively with $\hat{Q}_i$ for $i = 1 \ldots l$.

Case $x(\langle \alpha \rangle)$: continue.

Case $(x.\hat{Q})(\langle \alpha \rangle)$ where $\hat{Q} : \alpha'$:

If $\alpha' = (\beta_1 \cap \ldots \cap \beta_l) \rightarrow \beta_0$ (otherwise fail), let $T_\chi(\hat{Q}) = \{\alpha_1, \ldots, \alpha_k\}$ and

- add $\alpha' \doteq \beta_0$ to $C$,
- for each $i = 1 \ldots k$ choose $j \in \{1, \ldots, l\}$ and add $\alpha_i \doteq \beta_j$ to $C$,
- for each $j = 1 \ldots l$ choose $i \in \{1, \ldots, k\}$ and add $\alpha_i \doteq \beta_j$ to $C$.

Proceed recursively with $\hat{Q}$.

Case $(\hat{Q} \hat{R})(\langle \alpha \rangle)$ where $\hat{Q} : \alpha'$ and $\hat{R} : \alpha_1 \cap \ldots \cap \alpha_k$:

If $\alpha' = (\beta_1 \cap \ldots \cap \beta_l) \rightarrow \beta_0$ (otherwise fail)

- add $\alpha \doteq \beta_0$ to $C$,
- for each $i = 1 \ldots k$ choose $j \in \{1, \ldots, l\}$ and add $\alpha_i \doteq \beta_j$ to $C$,
- for each $j = 1 \ldots l$ choose $i \in \{1, \ldots, k\}$ and add $\alpha_i \doteq \beta_j$ to $C$.

Proceed recursively with $\hat{Q}$ and $\hat{R}$.

Otherwise: Fail (if none of the above cases apply).

(6) Compute the most general unifier $U$ for the syntactic unification instance $C$. If $C$ has no solution or $U(\alpha)$ is not strict for some variable $\alpha$ occurring in $C$, then fail.

(7) Choose a variable renaming $s$ and succeed by returning the elaboration $s(U(\hat{P}))$.

Algorithm $\mathcal{S}$ terminates because the recursive descent in step (5) is term-driven. Although typability can be decided in polynomial space [Dudenhefner and Rehof 2017c], particular elaborations can be (and are at most) of exponential size, due to syntactic unification.

Let us illustrate a run of $\mathcal{S}$ in the following Example 7.1.

Example 7.1 ($\mathcal{S}(\langle \lambda x. x \rangle y, 2)$. In step (1) the size (number of strict types) of individual decorations is guessed. We obtain $\hat{P} = (((\lambda x. (x(\langle \alpha_2 \rangle)(\langle \alpha_3 \rangle)(\langle \alpha_4 \rangle)))(\langle \alpha_5 \rangle)y(\langle \alpha_5, \alpha_6 \rangle))(\langle \alpha_7 \rangle))$.

In step (2) the shape (type variable or arrow) of individual decorating types is guessed, which initializes the set of constraints $C$ in step (3). We obtain $\alpha_1 = \alpha_i$ for $i = 2, 3, 6, 7$, and

$$\alpha_4 = (\beta_14 \cap \beta_24) \rightarrow \beta_04$$

Since we are interested in tight typings, each strict subformula also has to occur as a decoration, which is ensured in step (4). We obtain

$$C = \{\alpha_i \doteq \alpha_1 \mid i = 1 \ldots 7\} \cup \{\beta_1 \doteq \beta_2, \beta_01 \doteq \beta_3, \beta_14 \doteq \beta_5, \beta_24 \doteq \beta_6, \beta_04 \doteq \beta_7, \beta_15 \doteq \beta_2, \beta_05 \doteq \beta_3\}$$

Step (5) adds constraints that are enforced by the derivation rules wrt. the given term $M$. We obtain

$$C = \{\alpha_i \doteq \alpha_1 \mid i = 1 \ldots 7\} \cup \{\beta_1 \doteq \beta_2, \beta_01 \doteq \beta_3, \beta_14 \doteq \beta_5, \beta_24 \doteq \beta_6, \beta_04 \doteq \beta_7, \beta_15 \doteq \beta_2, \beta_05 \doteq \beta_3\}$$

$$\cup \{\alpha_1 \doteq \beta_04, \alpha_5 \doteq \beta_14, \alpha_6 \doteq \beta_24\} \cup \{\alpha_3 \doteq \beta_04, \alpha_1 \doteq \beta_14, \alpha_2 \doteq \beta_24\} \cup \{\alpha_3 \doteq \beta_01, \alpha_2 \doteq \beta_11\}$$

In steps (6) and (7) the constructed set of constraints is solved. The most general unifier of $C$ is

$$U = \{\alpha_1 := y \rightarrow \delta, \alpha_2 := y, \alpha_3 := \delta, \alpha_4 := (y \rightarrow \delta) \cap y) \rightarrow \delta, \alpha_5 := y \rightarrow \delta, \alpha_6 := y, \alpha_7 := \delta, \beta_11 := y, \beta_01 := \delta, \beta_14 := y \rightarrow \delta, \beta_24 := y, \beta_04 := \delta, \beta_15 := y, \beta_05 := \delta\}$$

resulting in the compact elaboration

$$U(\hat{P}) = \left(((\lambda x. (x(\langle y \rightarrow \delta \rangle)(\langle y \rightarrow \delta \rangle)(\langle y \rightarrow \delta \rangle)(\langle y \rightarrow \delta \rangle)(\langle y \rightarrow \delta \rangle)(\langle y \rightarrow \delta \rangle)(\langle y \rightarrow \delta \rangle))\right)$$

Observe that the introduced type variables are substituted by strict types. If the most general unifier would introduce a top-level intersection, then step (6) would fail.
Next, we establish soundness (Lemma 7.2) and, more importantly, completeness of Algorithm $\mathcal{I}$ wrt. compact elaborations (Lemma 7.3). The latter aspect is the main difference between Algorithm $\mathcal{I}$ and the typability decision procedure of Dudenhefner and Rehof [2017c].

**Lemma 7.2.** Given a $\lambda$-term $M$ and a dimension $d$, if $\mathcal{I}(M, d)$ returns the elaboration $P$, then $[P] = M$, $\|P\| \leq d$ and $P \in \mathcal{R}$.

**Proof.** Let $S = s \circ U$ be the substitution constructed in step 7 of Algorithm $\mathcal{I}$. We verify by induction on the recursive descent of $\mathcal{I}$ that for each intermediate elaboration $\hat{Q}$ in step (5) we have $S(\hat{Q}) \in \mathcal{R}$. Since $S$ is strict, $\|S(\hat{Q})\| \leq \|\hat{Q}\| \leq d$. □

**Lemma 7.3.** For any tight elaboration $P \in \mathcal{R}$ there is a run of $\mathcal{I}([P], \|P\|)$ returning $P$.

**Proof.** Let $[P] = M$ and $\|P\| = d$. Wlog. $P$ contains only normalized types (wrt. ACI of $\cap$). Since $P$ is tight, each subformula $\beta_1 \cap \ldots \cap \beta_l$ occurring in $P$ satisfies $l \leq |T(P)| \leq d \cdot |M|$. This allows us to construct a successful run of $\mathcal{I}(M, d)$ by explicitly giving a strict substitution $S$ that satisfies the set of constraints $C$. The detailed construction is given in Section 10. □

Using the above soundness and completeness of Algorithm $\mathcal{I}$ we describe in the following Theorem 7.4 a method to construct the basis $\mathcal{B}_{M,d}$ of a given term $M$ in a given dimension $d$.

**Theorem 7.4 (Basis Construction).** Given a $\lambda$-term $M$ and a dimension $d$, consider the set $C = \{A \in T_\circ \mid \mathcal{I}(M, d) \text{ returns a tight elaboration } P : A\}$. The finite basis $\mathcal{B}_{M,d}$ for $C_M[d]$ consists of the $\sqsubseteq_{\mathcal{B}_0}$-maximal elements of $C$.

**Proof.** By completeness of Algorithm $\mathcal{I}$ wrt. compact elaborations (Lemma 7.3) we discover all basis elements. By soundness of Algorithm $\mathcal{I}$ (Lemma 7.2) any additional tight elaborations are in $C_M[d]$. By Theorem 5.10, the $\sqsubseteq_{\mathcal{B}_0}$-maximal elements form the finite basis $\mathcal{B}_{M,d}$. □

Finally, combining type inference $\mathcal{I}$ with principal inhabitation $\mathcal{J}$ we can construct approximants of a given $\lambda$-term up to $\rightarrow_\eta$ (Theorem 7.5) and recognize the principal pairs of such.

**Theorem 7.5 (Soundness of Approximation by Type Inference).** Let $M$ be a $\lambda$-term and $d$ a dimension. If $\mathcal{I}(M, d)$ returns an elaboration $P : A$ such that $[\mathcal{I}(\Gamma_P, A)] = F$, then there exists an approximant $F' \in \mathcal{A}(M)$ such that $F \sqsubseteq F' \rightarrow_\eta F''$ for some $F''$.

**Proof.** Assume that $\mathcal{I}(M, d)$ returns $P : A$ such that $[\mathcal{I}(\Gamma_P, A)] = F$. By soundness of $\mathcal{I}$ (Lemma 7.2) we have $\Gamma_P \vdash_R M : A$. By the approximation theorem [van Bakel 2011, Thm. 10.4], there is a (wlog. $\sqsubseteq_\eta$-minimal) $F' \in \mathcal{A}(M)$ such that $\Gamma_P \vdash_R F' : A$. By Lemma 6.7, $F \sqsubseteq F' \rightarrow_\eta F''$ for some $F''$. □

In addition to soundness of the above approach, we show its completeness (Theorem 7.6), i.e. any approximant can be constructed in some dimension using the presented algorithms.

**Theorem 7.6 (Completeness of Approximation by Type Inference).** For any $F \in \mathcal{A}(M)$ there exists a dimension $d$ such that $\mathcal{I}(M, d)$ returns an elaboration $P : A$ and $[\mathcal{I}(\Gamma_P, A)] = F$.

**Proof.** Let $F \in \mathcal{A}(M)$. By Lemma 5.23 there exists $M'$ such that $F \sqsubseteq M'$ and $M \rightarrow_\beta M'$. The term $F$ has principal pair $(\Gamma, A)$ such that $\Gamma \vdash_R F \iff [A]$ where $F$ is compact by Lemma 5.24. By Lemma 5.25 together with compact subject expansion (Theorem 5.26) there exists a compact elaboration $P$ with $[P] = M$ such that $\Gamma \vdash_R P : [A]$ and $\Gamma = \Gamma_P$. By completeness of Algorithm $\mathcal{I}$ (Lemma 7.3), there is a run of $\mathcal{I}(M, \|P\|)$ returning $P : A$. Using principal inhabitant reconstruction (Lemma 6.3) we obtain $[\mathcal{I}(\Gamma_P, A)] = F$. □
Let us exemplify the above approach inspecting terms $c_2 = \lambda f. \lambda x. f \ (f \ x)$, $\omega = \lambda x. x$ and $I = \lambda x. x$. The following Example 7.7 shows that if a term is normalizable, then the presented approach (given a sufficient dimension) may discover the corresponding normal form. In fact, not performing $\beta$-reduction can be beneficial, which is illustrated in Example 7.8.

**Example 7.7.** Let $M = c_2 \ \omega \ I$. Let $A_1 = \alpha \rightarrow \alpha$, $A_2 = A_1 \rightarrow A_1$, $A_3 = A_2 \rightarrow A_2$ and $B_1 = A_1 \cap A_2 \rightarrow A_1$, $B_2 = A_2 \cap A_3 \rightarrow A_2$. In dimension 3 type inference $\mathcal{S}(M, 3)$ returns the compact elaboration (ignoring some decorations for readability)

$$ P = \left( (\lambda f. \lambda x. f \ (f \ (B_1)) \ (f \ (B_1, B_2)) \ x ((A_1, A_2, A_3))) \ \omega ((B_1, B_2)) \ I ((A_1, A_2, A_3)) \right) ((A_1)) $$

Particularly, $P : \alpha \rightarrow \alpha$ and $\Gamma_p = \emptyset$. Principal inhabitant reconstruction results in $[\mathcal{S} (\Gamma_p, \alpha \rightarrow \alpha)] = \lambda x. x$. Since $\lambda x. x$ is $\bot$-free and cannot be reduced using $\eta$, Theorem 7.5 guarantees that we have $\lambda x. x \in \mathcal{A}(M)$ and, containing no $\bot$, is the normal form of $M$.

**Example 7.8.** Let $M = c_2 \ldots c_2 \ I \ x$ where $c_2$ occurs $n$ times. Let $A_1 = \alpha \rightarrow \alpha$ and $A_{i+1} = A_i \rightarrow A_i$ for $i = 1 \ldots n$. In dimension 1 type inference $\mathcal{S}(M, 1)$ returns in exponential time the compact elaboration (ignoring some decorations for readability)

$$ P = \left( (\ldots ((c_2 ((A_{n+1})) \ c_2 ((A_n))) \ldots ((A_2)) \ c_2 ((A_2))) \ I ((A_1)) \right) ((A_1)) \ x ((\alpha)) \ ((\alpha)) $$

Again, $[\mathcal{S} (\{x : \alpha\}, \alpha)] = x$ is the normal form of $M$. More importantly, if $M$ would have been evaluated using $\beta$-reduction, one could obtain $I (I (\ldots (I (x) \ldots)))$ as an intermediate term containing a non-elementary number of occurrences of $I$ before arriving at the normal form. In fact, the notion of “amnesia” as idempotence in the exact sense of Neergaard and Mairson [2004] applies here. If non-idempotent intersection would have been used, then the dimensional bound would mirror reduction complexity.

**8 RELATED WORK**

Our main contribution, a theory of principality and approximation of principality in bounded dimension, cuts across three distinct fields of research.

First, decorating pure subterms with assigned types, elaborations can be considered Church-style $\lambda$-terms. While languages as early as Forsythe [Reynolds 1996] encompass Church-style $\lambda$-terms as well as a notion of intersection types, the line of work on concise ways to incorporate intersection types into Church-style $\lambda$-terms is still ongoing [Liquori and Stolze 2018] provide a recent overview). In contrast to this line of work, the elaboration calculus of dimensional intersection types is used as a measurement system under the norm to achieve principles of bounding.

Second, defining fragments of the standard intersection type calculus by a bounding principle underlies the possibility of type inference. In our case the bounding principle is the notion of dimension [Dudenhefner and Rehof 2017b] for which decidability of type inference is established [Dudenhefner and Rehof 2017c]. Our investigation extends the results of Dudenhefner and Rehof [2017c] by providing a new understanding of principality and approximation under dimensional bound. Notably, a different bounding principle is Leivant’s notion of rank [Leivant 1983]. Decidability of type inference under bounded rank as well as a theory of principality are established [Kfoury et al. 1999; Kfoury and Wells 1999, 2004], using so-called expansion variables to account for the intersection introduction rule. Since dimensional bound and rank are two orthogonal bounding principles (for a direct comparison see Dudenhefner and Rehof [2017c]), the notion of $\beta$-unification [Kfoury and Wells 2004] (for which only rank is known to be a sensible restriction) as well as expansion variables do not appear to be applicable here. A different restriction of the full system is to remove the intersection introduction rule [Kurata and Takahashi 1995; Rehof and
Urzyczyn 2012]. Since dimension measures the amount of intersection introduction, such systems are directly captured by the one-dimensional fragment. Various other restrictions alternative to that of rank have been studied, which typically do not allow for arbitrary levels of approximation of the full system. Coppo and Giannini [1995] introduce a restriction in which weakening of type assumptions and intersection elimination are not possible. They establish type inference and a notion of principal types for the system, but subject reduction fails as a consequence of the absence of weakening. Semi-decision procedures for typability or type inference have also been studied based on various forms of generalized unification encompassing type expansion, e.g. by Ronchi Della Rocca [1988] and Boudol [2008].

Third, our results fall in the line of work on principality in calculi with intersection types (deeply intertwined with the theory of approximants) dating back to the seminal paper by Coppo et al. [1980] and under development for over thirty years, including several adaptations and generalizations to variants of the intersection type system such as developed by Ronchi Della Rocca and Venneri [1984] for the “BCD” system of Barendregt et al. [1983], and for the “strict”, “relevant”, and “essential” variants (see van Bakel [2011] for an overview). Our notion of compactness uses van Bakel’s analysis of principality by factoring that theory through the relevant variant of the system [van Bakel 2011]. One common aspect among the existing work is the necessity to capture intersection introduction in general, using type expansion or expansion variables (see Carlier and Wells [2005] for a history of expansion). Since type expansion is a relatively complicated generating operation from principal typings, Carlier and Wells [2012] undertake an algebraic analysis of expansion in order to make the notion more accessible. In the present work dimensionality is used to bound the amount of intersection introduction, thereby allowing us to eliminate expansion as a generating operation for each dimensional fragment, using instead the linear operators (span) of the semimodule. As we have observed in the paper, one can see deep expansion as a mechanism for producing basis vectors in increasing dimension, whereas top-level expansion is absorbed into the generating operations of linear span. Thus the semimodular perspective perhaps “naturalizes” the theory of generation in terms of linear algebra while still retaining arbitrarily large fragments under increasing bound approximating standard principality. We therefore might hope that our analysis in terms of semimodular bases could also be seen as a contribution towards understanding and explaining the classical notion of principality for intersection types.

We are not aware of any previous work identifying the semimodular structure of intersection types. Structures of modules, semimodules, or semirings appear in different lines of work on type systems augmented with notions of resources [Chica and Smith 2014; McBride 2016].

For our technical results, we improve on existing techniques and algorithms. In particular, with the notion of compactness we consolidate a precise notion of tightness (implicitly used by Dudenhefner and Rehof [2017c] to bound the width on the intersection operator) with an invertible notion of filtration (also used in context of principality for simple types [Dudenhefner and Rehof 2017a]) to classify and compute strongly irredundant typings. The presented type inference algorithm is a modification of the typability algorithm developed by Dudenhefner and Rehof [2017c] establishing completeness for such strongly irredundant typings. The presented principal inhabitation algorithm (incorporating elaborations) is a direct inversion of the principal pair generation procedure [Coppo et al. 1980]. Uniqueness of inhabitation with intersection types was used in a different setting by Tsukada and Ong [2016], intersections being refinements of simple types there. General connections between type inference for intersection types and normalization under $\beta$-reduction have been studied by Neergaard and Mairson [2004], and Boudol [2008] shows that a notion of unification corresponds to a form of non-erasing $\beta$-reduction. Our notion of “normalization by type inference” can be seen as a different realization of such ideas, which is aligned with the theory of approximants for $\lambda$-calculus [Barendregt 1984], by approximating up along the dimensional parameter.
9 CONCLUDING DISCUSSION AND FUTURE WORK

The presented work contains three main developments encompassing organization and generation of typings in the intersection type system, and computation of approximants in the pure \(\lambda\)-calculus by type theoretic means.

First, for each untyped \(\lambda\)-term \(M\) and each dimension \(n\), intersection types that can be assigned to \(M\) within dimension \(n\) respecting relevance are shown to be spanned by unique, finite semimodular bases \(B_{M,n}\). Compactness of basis vectors allows us to algorithmically construct bases using bounded type inference. Bases thereby provide a natural notion of principality for the dimension-bounded systems as a canonical representation of the semimodular subspaces containing all the \(n\)-dimensional typings for each \(n\). This notion of principality approximates the standard notion of principality and captures standard principal types for sufficient dimensional parameter \(n\).

Second, using compactness and uniqueness properties of principal typings in the intersection type system we show polynomial time decidability of principal inhabitation by explicitly constructing and verifying candidate inhabitants.

Third, combining algorithms used to obtain the above results allows us to evaluate arbitrary terms to approximate normal forms without using \(\beta\)-reduction. Increasing the dimension parameter \(n\) we are guaranteed to discover any approximant and its standard principal type of the given term.

Besides being of theoretical interest as an analysis of the dimensional fine-structure of principality for intersection types, our semimodular theory of principality should have practical consequences due to its complete algorithmic realization shown in the second part of the paper. Future work includes the implementation of a type inference system with intersection types based on the results presented. From a practical standpoint, an advantage of the approach developed here should be its mathematical foundations in the theory of semimodules, yielding a natural, predictable and explicable form of approximation, at the same time allowing for a canonical presentation of typing information in the form of bases. Let us note here also that the following decision problems, which are of obvious interest in connection with type inference and presenting type information, are decidable:

1. Given a basis \(B_{M,n}\) and a type \(\sigma\), does \(\sigma \in \text{span}(B_{M,n})\) hold?
2. Given a basis \(B_{M,n}\) and a type \(\sigma\), does there exist \(\tau \in \text{span}(B_{M,n})\) such that \(\tau \leq \sigma\) holds?

Decidability of these problems follows by immediate reduction to the intersection type matching problem, which is known to be \(\text{NP}\)-complete [Düdder et al. 2013]: match \(\bigcap A \in B_{M,n} A\) against the goal type in each case (resp. without and with subtyping). In the second problem, subtyping can be taken to be any of the standard relations encompassed by the “BCD”-theory [Barendregt et al. 1983]. This observation also shows that our theory immediately covers the so-called essential intersection type system [van Bakel 2011], in which relevance is not imposed, but generation from principal types factors through the relevant system with a subtyping step (“lifting”) after the relevant chain (see the proof of van Bakel [2011, Theorem 10.27]).

10 ADDITIONAL PROOFS

Proof of Lemma 5.9: Let \(X, Y \subseteq \mathcal{T}_0\) with \(Y \subseteq \text{span}_0(X)\). The set \(B_X\) is a basis for \(Y\) if and only if \(\text{span}_0(X) < X\).

**Proof.** Call a subset \(X \subseteq \mathcal{T}_0\) separated if and only if \(\forall A, B \in X. (\exists r \in \mathcal{R}. A = rB) \Rightarrow A = B\). The following two properties are useful:

1. If \(X \subseteq \mathcal{T}_0\) is linearly independent, then \(\text{span}_0(X) < X\).

For assume \(X\) is linearly independent. To see that \(\text{span}_0(X) < X\), assume \(B \in \text{span}_0(X)\) and \(A \in X\) with \(A = rB\). We can write \(B = s_1D_1 \cap \cdots \cap s_nD_n\) for some \(D_i \in X\) and \(s_i \in \mathcal{R}_0\) \((i = 1 \ldots n)\). It
follows from $B \in \mathcal{T}_0$ that we have $B = s_i D_i$ for some $i$. Then $A = rB = r \circ s_i D_i$. Since $X$ is linearly independent and $A, D_i \in X$, it follows that $A = D_i$, hence $r \circ s_i$ is the identity on $TV(D_i)$. It follows that $s_i$ is a renaming on $D_i$ and $r$ is a renaming on $s_i D_i$ and therefore (by $B = s_i D_i$) also on $B$. We next show:

(II) A subset $X \subseteq \mathcal{T}_0$ is linearly independent if and only if $X$ is separated.

For assume $X$ is linearly independent. If $A, B \in X$ with $A = rB$, it follows from linear independence of $X$ that $A = B$. Hence, $X$ is separated. On the other hand, any separated subset is linearly independent. For, if $A = \sum_{i=1}^n r_i B_i$ for $B_i \in X$, $r_i \in \mathcal{R}$ ($i = 1 \ldots n$), then $A = r_j B_j$ for some $j$. Because $X$ is separated, $A = B_j$.

Let us now prove the lemma. Suppose $\mathcal{Y} \subseteq \text{span}_0(X)$.

$(\Rightarrow)$. Assume $\mathcal{B}_X$ is a basis for $\mathcal{Y}$. It follows from (I) that we have $\text{span}_0(\mathcal{B}_X) < \mathcal{B}_X$. To show $\text{span}_0(X) < X$, let $A \in X$, $B \in \text{span}_0(X)$ and $s \in \mathcal{R}$ with $A = sB$. We need to show that $s$ is a renaming on $B$. It follows from $B \in \text{span}_0(X)$ that $B \in \text{span}_0(\mathcal{B}_X)$. For $B = tC$ for some $C \in X$ implies $B = t \circ r^{-1}C'$ for $C' = f([C]_-) = rC \in \mathcal{B}_X$ for some renaming $r$, where $f$ is the choice function for $\mathcal{B}_X$. Let $A' = f([A]_-) = r_1 A \in \mathcal{B}_X$. We have $A' = r_1 \circ sB$. Since $A' \in \mathcal{B}_X$, $B \in \text{span}_0(\mathcal{B}_X)$ and $\text{span}_0(\mathcal{B}_X) < \mathcal{B}_X$, it follows that $r_1 \circ s$ is a renaming on $B$, hence $s$ is a renaming on $B$, thereby showing $\text{span}_0(X) < X$.

$(\Leftarrow)$. Assume $\text{span}_0(X) < X$. We show that $\mathcal{B}_X$ is a basis for $\mathcal{Y}$. First, $\mathcal{B}_X$ is separated. For suppose $A, B \in \mathcal{B}_X$ with $rB = A$. Then, since $\text{span}_0(X) < X$, $r$ is a renaming. Therefore, $A \sim B$, hence $f([A]_-) = f([B]_-)$. Since $\mathcal{B}$ is separated, it is linearly independent by (II). Now suppose $C \in \mathcal{Y}$. Then $C \in \text{span}_0(\mathcal{X})$. Then $C = tA$ for some $A \in X$. Since $A = s(f([A]_-))$ for some renaming $s$, we have $C = t \circ s(f([A]_-))$ which shows $C \in \text{span}_0(\mathcal{B})$. □

Proof of Lemma 7.3: For any tight elaboration $P \in \mathcal{R}$ there is a run of $\mathcal{S}(\|P\|, \|P\|)$ returning $P$.

Proof. Given a tight elaboration $P \in \mathcal{R}$ let $\|P\| = M$ and $\|P\| \leq d$. Wlog. $P$ contains only normalized types (wrt. ACI of $\cap$). Since $P$ is tight, each subformula $\beta_1 \cap \ldots \cap \beta_l$ occurring in $P$ satisfies $l \leq |T(P)| \leq d \cdot |M|$. We construct a successful run of $\mathcal{S}(M, d)$ by explicitly giving a strict substitution $S$ that satisfies the set of constraints $C$.

In step 1, for each subterm of $M$ with corresponding annotation $\langle [A_1, \ldots, A_l] \rangle$ in $P$ we have $l \leq d$ and the procedure may choose the annotation $\langle [a_1, \ldots, a_l] \rangle$ to construct the elaboration $\hat{P}$. Additionally, we fix $S(a_i) = A_i$ for $i = 1 \ldots l$.

In step 2, for each variable $a$ occurring in $\hat{P}$ we previously either fixed $S(a) = \gamma$ for some type variable $\gamma$ or $S(a) = (B_1 \cap \ldots \cap B_l) \rightarrow B_0$ with $l \leq d \cdot |M|$. In the first case, the procedure may choose $\Gamma a = a$. In the second case, the procedure may choose $\Gamma a = (B_1 \cap \ldots \cap B_l) \rightarrow B_0$. Additionally, we fix $S(\beta_i) = B_i$ for $i = 0 \ldots l$. In both cases the corresponding constraint $\alpha \models \Gamma a$ in step 3 is satisfied by $S$.

In step 4, due to tightness of $P$ for any decorating type $(B_1 \cap \ldots \cap B_l) \rightarrow B_0$ each $B_j$ for $j = 0 \ldots l$ are decorating types. Therefore, the procedure may choose corresponding decorations in $\hat{P}$ satisfying additional constraints in $C$.

Next, we show that in step 5 the recursive descent may exactly follow the derivation of $P \in \mathcal{R}$. Therefore, any additional constraints in $C$ are satisfied by $S$. We refer by $\hat{Q}$ to subterms in $P$ that correspond to subterms $Q$ in $P$. By construction, $Q = S(\hat{Q})$. We write $T(S)$ to indicate an elaboration with decoration set $S$.

Case $D_\|\$: The corresponding subterm is $\hat{0}_{T(\langle \omega \rangle)}$ satisfying the first case.
We are indebted to Mario Coppo, Mariangiola Dezani, Simona Ronchi Della Rocca, and the Torino $\alpha$-calculus group for numerous discussions on intersection types. Especially, we thank Mariangiola Dezani for providing an offprint of the indispensable Coppo et al. [1980]. We also appreciate the effort of Dominik Krüger to improve overall presentation. Finally, we would like to thank the anonymous referees for their valuable comments and helpful suggestions.
REFERENCES

João Alpuim, Bruno C. d. S. Oliveira, and Zhiyuan Shi. 2017. Disjoint Polymorphism. In Programming Languages and Systems - 26th European Symposium on Programming, ESOP 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings. 1–28. https://doi.org/10.1007/978-3-662-54434-1_1

Michael Arnt. 2011. Algebra (2nd edition). Pearson Education.

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

Henk P. Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. 1983. A Filter Lambda Model and the Completeness of Type Assignment. Journal of Symbolic Logic 48, 4 (1983), 931–940. https://doi.org/10.2307/2273659

Henk P. Barendregt, Wil Dekkers, and Richard Statman. 2013. Lambda Calculus with Types. Perspectives in Logic, Cambridge University Press.

Gérard Boudol. 2008. On strong normalization and type inference in the intersection type discipline. Theor. Comput. Sci. 398, 1-3 (2008), 63–81. https://doi.org/10.1016/j.tcs.2008.01.045

Sébastien Carlier and Joe B. Wells. 2005. Expansion: the Crucial Mechanism for Type Inference with Intersection Types: A Survey and Explanation. Electr. Notes Theor. Comput. Sci. 136 (2005), 173–202. https://doi.org/10.1016/j.entcs.2005.03.026

Sébastien Carlier and Joe B. Wells. 2012. The Algebra of Expansion. Fundam. Inform. 121, 1-4 (2012), 43–82. https://doi.org/10.3233/FI-2012-771

Mario Coppo and Mariangiola Dezani-Ciancaglini. 1980. An Extension of Basic Functionality Theory for Lambda-Calculus. Notre Dame Journal of Formal Logic 21 (1980), 685–693.

Mario Coppo, Mariangiola Dezani-Ciancaglini, and Patrick Sallé. 1979. Functional characterization of some semantic equalities inside λ-calculus. In ICALP International Colloquium on Automata, Languages, and Programming. Springer LNCS 71.

Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti Venneri. 1980. Principal Type Schemes and λ-Calculus Semantics. In R. Hindley and J. Seldin (Eds.), To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism. Aeccademic Press, London, 480–490.

Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti Venneri. 1981. Functional Characters of Solvable Terms. Zeitschrift für mathematische Logik und Grundlagen der Mathematik (1981), 45–58.

Mario Coppo and Paola Giannini. 1995. Principal Types and Unification for a Simple Intersection Type System. Inf. Comput. 122, 1 (1995), 70–96. https://doi.org/10.1006/inco.1995.1141

Boris Düdder, Moritz Martens, and Jakob Rehof. 2013. Intersection Type Matching with Subtyping. In Typed Lambda Calculi and Applications, 11th International Conference, TLCA 2013, Eindhoven, The Netherlands, June 26-28, 2013. Proceedings. 125–139. https://doi.org/10.1007/978-3-642-38946-7_11

Andrei Dudenhefner and Jakob Rehof. 2017a. The Complexity of Principal Inhabitation. In 2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017, September 3-9, 2017, Oxford, UK, 15:1–15:14. https://doi.org/10.4230/LIPIcs.FSCD.2017.15

Andrei Dudenhefner and Jakob Rehof. 2017b. Intersection type calculi of bounded dimension. In Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017. 653–665. https://doi.org/10.1145/3009837

Andrei Dudenhefner and Jakob Rehof. 2017c. Typability in bounded dimension. In 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017. 1–12. https://doi.org/10.1109/LICS.2017.8005127

Dan R. Ghica and Alex I. Smith. 2014. Bounded Linear Types in a Resource Semiring. In Programming Languages and Systems - 23rd European Symposium on Programming, ESOP 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings. 331–350. https://doi.org/10.1007/978-3-642-54833-8_18

Jonathan S. Golan. 1999. Semirings and their Applications. Springer.

Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luis Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, Hugo Torres Vieira, and Gianluigi Zavattaro. 2016. Foundations of Session Types and Behavioural Contracts. Comput. Surveys 49, 1, Article 3 (April 2016), 36 pages. https://doi.org/10.1145/2873052

Massimiliano M. Abadi, Martin Hofmann, Frank Pfenning, and Peter Urzyczyn. 2011. Reductio Calculi. In Advances in Modal Logic, Volume 6, Advances in Modal Logic, V. 6. College Publications. 1–20. https://doi.org/10.1017/S0007055405002057

Assaf J. Kfoury. 2004. Principality and decidability of type inference for intersection types. In POPL 1999, Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Antonio, TX, USA, January 20-22, 1999. 161–174. https://doi.org/10.1145/292540.292556

Assaf J. Kfoury and Joe B. Wells. 2004. Principality and type inference for intersection types using expansion variables. Theor. Comput. Sci. 311, 1-3 (2004), 1–70. https://doi.org/10.1016/j.tcs.2003.10.032
Toshihiko Kurata and Masako Takahashi. 1995. Decidable properties of intersection type systems. In TLCA 1995, Proceedings of Typed Lambda Calculus and Applications (LNCS), Vol. 902. Springer, 297–311. https://doi.org/10.1007/BFb0014060

Daniel Leivant. 1983. Polymorphic Type Inference. In POPL 1983, Proceedings of the 10th ACM Symposium on Principles of Programming Languages. 88–98. https://doi.org/10.1145/567067.567077

Luigi Liquori and Claude Stolze. 2018. The Delta-calculus: syntax and types. CoRR abs/1803.09660 (2018). arXiv:1803.09660 http://arxiv.org/abs/1803.09660

Daniel Leivant. 1983. Polymorphic Type Inference. In POPL 1983, Proceedings of the 10th ACM Symposium on Principles of Programming Languages. 88–98. https://doi.org/10.1145/567067.567077

Conor McBride. 2016. I Got Plenty o’ Nuttin’. In A List of Successes That Can Change the World - Essays Dedicated to Philip Wadler on the Occasion of His 60th Birthday. 207–233. https://doi.org/10.1007/978-3-319-30936-1_12

Grigory E. Mints. 1981. Closed categories and the theory of proofs. Journal of Soviet Mathematics 15, 1 (1981), 45–62.

Peter Møller Neergaard and Harry G. Mairson. 2004. Types, potency, and idempotency: why nonlinearity and amnesia make a type system work. In Proceedings of the Ninth ACM SIGPLAN International Conference on Functional Programming, ICFP 2004, Snow Bird, UT, USA, September 19-21, 2004, 138–148. https://doi.org/10.1145/1016850.1016871

Garrel Pottinger. 1980. A Type Assignment for the Strongly Normalizable Lambda-Terms. In To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism, J. Hindley and J. Seldin (Eds.). Academic Press, 561–577.

Jakob Rehof and Paweł Urzyczyn. 2012. The Complexity of Inhabitation with Explicit Intersection. In Logic and Program Semantics - Essays Dedicated to Dexter Kozen on the Occasion of His 60th Birthday (LNCS), Robert L. Constable and Alexandra Silva (Eds.), Vol. 7230. Springer, 256–270. https://doi.org/10.1007/978-3-642-29485-3_16

John C. Reynolds. 1996. Design of the Programming Language Forsythe. Technical Report CMU-CS-96-146. CMU.

Simona Ronchi Della Rocca. 1988. Principal Type Scheme and Unification for Intersection Type Discipline. Theor. Comput. Sci. 59 (1988), 181–209. https://doi.org/10.1016/0304-3975(88)90101-6

Simona Ronchi Della Rocca and Betti Venneri. 1984. Principal Type Schemes for an Extended Type Theory. Theor. Comput. Sci. 28 (1984), 151–169. https://doi.org/10.1016/0304-3975(83)90069-5

Takeshi Tsukada and C.-H. Luke Ong. 2016. Plays as Resource Terms via Non-idempotent Intersection Types. In Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS ’16, New York, NY, USA, July 5-8, 2016. 237–246. https://doi.org/10.1145/2933575.2934553

Paweł Urzyczyn. 1999. The Emptiness Problem for Intersection Types. Journal of Symbolic Logic 64, 3 (1999), 1195–1215. https://doi.org/10.2307/2586625

Steffen van Bakel. 1993. Principal Type Schemes for the Strict Type Assignment System. J. Log. Comput. 3, 6 (1993), 643–670. https://doi.org/10.1093/logcom/3.6.643

Steffen van Bakel. 2011. Strict Intersection Types for the Lambda Calculus. Comput. Surveys 43, 3 (2011). https://doi.org/10.1145/1922649.1922657