THE ALGEBRAIC INTERSECTION TYPE UNIFICATION PROBLEM

ANDREJ DUDENHEFNER, MORITZ MARTENS, AND JAKOB REHOF

Department of Computer Science, Technical University of Dortmund, Dortmund, Germany
e-mail address: {andrej.dudenhefner,moritz.martens,jakob.rehof}@cs.tu-dortmund.de

ABSTRACT. The algebraic intersection type unification problem is an important component in proof search related to several natural decision problems in intersection type systems. It is unknown and remains open whether the algebraic intersection type unification problem is decidable. We give the first nontrivial lower bound for the problem by showing (our main result) that it is exponential time hard. Furthermore, we show that this holds even under rank 1 solutions (substitutions whose codomains are restricted to contain rank 1 types). In addition, we provide a fixed-parameter intractability result for intersection type matching (one-sided unification), which is known to be NP-complete.

We place the algebraic intersection type unification problem in the context of unification theory. The equational theory of intersection types can be presented as an algebraic theory with an ACI (associative, commutative, and idempotent) operator (intersection type) combined with distributivity properties with respect to a second operator (function type). Although the problem is algebraically natural and interesting, it appears to occupy a hitherto unstudied place in the theory of unification, and our investigation of the problem suggests that new methods are required to understand the problem. Thus, for the lower bound proof, we were not able to reduce from known results in ACI-unification theory and use game-theoretic methods for two-player tiling games.

1. INTRODUCTION

Intersection type systems occupy a prominent place within the theory of typed λ-calculus [BDS13]. As is well known, variants of such systems characterize deep semantic properties of λ-terms, including normalization and solvability properties [BDS13]. As a consequence of the enormous expressive power of intersection types, standard type-theoretic decision problems are undecidable for general intersection type systems, 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?). A combinatorial problem centrally placed in many classical type-theoretic decision problems is that of *unification* over a term algebra of type expressions: given two types σ and τ, does there exist a substitution S of types for type variables such that $S(\sigma) = S(\tau)$ in a suitable equational theory (=) of types? In this paper we wish to initiate a study of the problem of *algebraic intersection type unification* which we believe to be of considerable systematic interest. We consider

an equational theory of intersection types induced by an important subtyping relation for intersection types [BCDC83] extended with type constants. Although decidability of algebraic intersection type unification appears to be surprisingly difficult and remains open, the present paper provides the first nontrivial lower bound indicating that the problem is of very high complexity: we prove that the problem is Exptime-hard. Our proof uses game-theoretic methods, in the form of two-player tiling games, which we believe to be of intrinsic interest and potentially helpful towards understanding the problem of decidability. Moreover, as we will show, the algebraic intersection type unification problem occupies a natural but hitherto (so far as we are aware) unstudied place in the theory of unification.

The algebraic unification problem considered in this paper is distinct from problems of unification with type schemes under *chains of substitutions and expansions*, as found in, e.g., the work by Ronchi Della Rocca [Ron88]. This line of work was motivated from the theory of principality for intersection types but contains the algebraic intersection type unification problem as a component. In the following, unless otherwise stated, when we talk about "unification" we shall understand algebraic unification considered here.

We hope with this paper to stimulate further work on a fascinating open problem in type theory as well as in unification theory. Many variants of intersection type subtyping have been introduced (see [BDS13] for an overview). For all of these theories corresponding algebraic intersection type unification problems can be induced along the lines of this paper, and consequently we should think of the algebraic unification problem considered here for one specific theory to be indicative of a whole *family* of such problems. For many, probably most, of these variations it is not known whether algebraic unification is decidable. The present paper establishes the result that for one important variant the problem is of high complexity and suggests that the question of decidability is both challenging and interesting for intersection type unification in general.

1.1. **Motivation.** Since the algebraic unification problem under study in this paper has not hitherto been considered *per se*, we discuss some general motivations for considering the problem here, whereas more specific remarks can be found in Sec. 2.

First, as will be explained in more detail below and in Sec. 2.1, the algebraic unification problem for intersection types appears be a natural problem of systematic interest in unification theory [BS01], occupying a hitherto unstudied position there and posing interesting challenges which do not seem to be readily amenable to known techniques.

The second main motivation for studying the problem comes from applications of the theory of intersection types in programming. It is well known that important decision problems (most prominently, typability and type checking) for various type systems can be characterized, modulo polynomial time transformations, by certain algebraic and combinatorial problems, which do not refer to the rules of the type system at all, thereby providing abstract combinatorial and algorithmic "signatures" for these systems. To mention a few well known examples, typability in simple types can be characterized as a standard unification problem over the free term algebra of simple type expressions and can be solved using the unification algorithm of Robinson [Hin08, Sec. 3D]. Typability in systems extending simple types with subtyping can be characterized by the problem of subtype satisfiability [Reh98]. Typability in ML can be characterized by acyclic semi-unification problems, whereas the problem becomes equivalent to general semi-unification in the presence of polymorphic recursion and is undecidable [KTU93]. In contrast to these examples, it does not appear to be obvious, in general, how to characterize typability problems for intersection type

systems in terms of such algebraic type constraint systems. The reason is, essentially, that intersection type systems are "proof functional" (rather than "truth functional"). More precisely, the rule of intersection introduction requires *one and the same term* to have both types τ and σ for the term to have the type $\tau \cap \sigma$. Because of this dependency on the identity of the subject term, it is not clear how the logical effects of the rule can be captured in a purely algebraic way. If we would attempt to characterize typability by extracting algebraic type constraints from a given term, we would have difficulty deciding, from the structure of term alone, how to apply the rule of intersection introduction.

Nonetheless, it turns out that intersection type unification as considered here is a natural *component* in deciding problems related to typability and type checking for intersection types. As already mentioned, the algebraic unification problem considered in this paper is a component problem in the problem of unification with type schemes under *chains of substitutions and expansions* studied by Ronchi Della Rocca [Ron88] (assuming we consider the problem with usual type constants, as suggested in [Ron88, p.182]), for which a semi-decision procedure is provided. Further remarks comparing to this notion of unification can be found in Sec. 2.2. The algebraic unification problem is relevant when investigating restrictions of the intersection type system. Such investigation, in turn, may be of particular interest, because the full system is undecidable (both with respect to typability, type checking, and inhabitation, see [BDS13] for an overview). An example of such a restriction is provided in Sec 5.1 (Example 4), where we consider combinators with intersection types over arbitrary bases but excluding the intersction introduction rule, motivated from applications in program synthesis. The typability problem for this fragment is polynomial time equivalent to intersection type unification. Another example is the restriction of λ-calculus with explicit intersection types studied in [KT95], which is equivalent to the intersection type system of [BCDC83] without intersection introduction. We know that type checking is decidable for β-normal forms in this system, as was shown in [KT95], and inhabitation is decidable and exponential space complete [RU12]. But we do not know whether type checking is decidable in general for this restriction. However, it is not difficult to see that the problem is reducible to intersection type unification. An interesting question to consider here is whether it might be equivalent to either the unification problem or to its associated *matching problem* considered in this paper. The algebraic unification problem is likely to be involved as soon as one attempts to combine intersection types with usual notions of type instantiation. A recent example is the so-called type tallying problem of [CNXA15], which is decidable. The main difference to the problem considered in this paper is the inclusion of union and negation type connectives as well as recursive types. Recursive types in particular allow for a direct solution of cyclic constraints such as $\alpha \doteq \alpha \cap (\alpha \rightarrow b)$ by $S(\alpha) = \mu X.(X \cap (X \rightarrow b))$. For all we know at this time, it could be the case that the algebraic intersection type unification problem considered in this paper is undecidable, but that it becomes decidable when recursive types are added. Finally, let us mention that it may be interesting, in the light of this paper, to compare the combinatorial complexity of the theory of subtyping and type equivalence in the *strict* intersection type system [vB11] with that of, say, [BCDC83], by considering associated algebraic unification problems.

Perhaps these theories will turn out to be distinguishable in that respect, but the question of decidability of unification is open for both systems.

It should be emphasized that, as already indicated, many variants of intersection type subtyping theories exist (a catalogue can be found in [BDS13]), giving rise to a whole *family* of intersection type unification problems analogous to the one studied in this paper. While we discuss some variations, other members of this family of problems are yet to be studied. Our choice of the specific variant of subtyping and type equivalence arises from the theory presented in [BCDC83] extended with type constants. This choice was motivated by two main considerations. For one, the theory of subtyping and equivalence of [BCDC83] is particularly rich, and several variants can be seen as subsystems thereof. Second, we adopted this theory for applications within type-based program synthesis based on combinatory logic with intersection types [Reh13, DMR14], where it is useful for supporting type refinement with semantic types (compare also [FP91]). Similar theories of intersection type subtyping and equivalence have been adopted, or adapted, in many contexts, an example being the programming language Forsythe by Reynolds [Rey96]. It should be noted that the exact theory of intersection types considered in [BCDC83] contains only a single type constant, denoted ω, which is equivalent to the empty intersection and is semantically a universal type. Because of the special properties of the type ω (in particular, the equation $\tau \to \omega = \omega$), the intersection type unification problem is trivial for this exact variant, because we can solve any such problem by mapping all type variables to ω (see also further remarks in Sec. 2). However, if we leave out the type ω (as is done in some variations), or if we introduce other type constants (such as `bool`, `int`, `real`), as is done in applications to programming languages, we arrive at theories for which the development in this paper is relevant. A more detailed analysis of further variants must, however, be postponed for future work to which the present paper may be seen as an invitation.

1.2. **Algebraic properties.** We briefly summarize some of the most important algebraic properties of the equational theory of intersection types needed to appreciate the systematic placement of the unification problem (full details are given later in the paper). Intersection type systems are characterized by the presence of an associative, commutative, idempotent operator, $\cap$ (intersection), which allows the formation of types of the form $\sigma \cap \tau$. In addition, we have function types, $\sigma \to \tau$. The standard equational theory, denoted $=$, of intersection types [BCDC83] is induced from a partial order $\leq$ on types, referred to as subtyping, by taking type equality to be the relation $\leq \cap \leq^{-1}$. Conversely, as will be discussed in the paper, it is also possible to give a purely equational presentation of subtyping. Because intersection is greatest lower bound with respect to subtyping, the intersection type unification problem is equivalent to the subtype satisfiability problem: given σ and τ, does there exist a type substitution S such that $S(\sigma) \leq S(\tau)$? The latter is equivalent to $S(\sigma) \cap S(\tau) = S(\tau)$, hence satisfiability is reducible to unification. The equational theory includes right-distributivity of $\to$ over $\cap$: $\sigma \to (\tau_1 \cap \tau_2) = (\sigma \to \tau_1) \cap (\sigma \to \tau_2)$ and left-contravariance of $\to$ with respect to subtyping: $\sigma_1 \to \tau_1 \leq \sigma_2 \to \tau_2$ whenever $\sigma_2 \leq \sigma_1$ and $\tau_1 \leq \tau_2$. As a consequence, one has "half left-distributivity" of $\to$ over $\cap$: $(\sigma_1 \to \tau) \cap (\sigma_2 \to \tau) \leq (\sigma_1 \cap \sigma_2) \to \tau$ (but the symmetric relation does not hold). Altogether, we could say for short that $\to$ is "$1\frac{1}{2}$-distributive" over $\cap$. Axioms specific to a special largest type, ω, are added in some variants of the theory (both variants, with or without ω, are important in type theory), including the recursion axiom $\omega = \omega \to \omega$, and we have the derived equation $\sigma \to \omega = \omega$. Thus, ω is unit (neutral element) with respect to $\cap$ and right-absorbing element with respect

to $\rightarrow$. Clearly, as already mentioned, in a system in which ω is the only constant any two intersection types are unifiable by replacing all occurring variables by ω. The variant of the problem considered in the following includes other type constants besides ω as well.

2. Related work

We consider the most closely related work within unification theory and type theory.

2.1. **Related work in unification theory.** The single most directly related piece of work in the literature is the study from 2004 by Anantharaman, Narendran, and Rusinowitch on unification modulo ACUI (associativity, commutativity, unit, idempotence) plus distributivity axioms [ANR04]. They consider equational theories over a binary ACUI symbol, denoted $+$, together with a binary operator, $*$, which distributes (left, right, or both) over $+$. Indeed, since (as summarized above) we have an ACUI theory of $\cap$ together with $\rightarrow$ enjoying distributivity properties over $\cap$, it would seem that we are temptingly close to the theories studied in [ANR04], by thinking of their $+$ as $\cap$ and their $*$ as $\rightarrow$. In particular, algebraically closest among the theories covered in that paper, ACUI-unification with one-sided (say, left) distributivity (ACUID$_l$) is shown to be Exptime-complete, using techniques from unification modulo homomorphisms [BN01]. But it turns out that there are fundamental obstacles to transferring results or techniques from ACUID$_l$-unification to intersection type unification, as will be summarized next.

With regard to any upper bound, the main obstacle is that, whereas decidability of the ACUID-problems can be relatively straight-forwardly obtained by appeal to an occurs-check (nontrivial cyclic equations have no solutions), this is very far from being clear in the case of intersection type unification. Indeed, even in the absence of the recursive type ω, we can solve nontrivial cyclic constraints, due to contravariance. For example, the constraint $\alpha \overset{.}{\leq} \alpha \rightarrow b$ (where $\overset{.}{\leq}$ denotes a formal subtyping constraint, α is a type variable and b is a constant) can be solved by setting $S(\alpha) = b \cap (b \rightarrow b)$. The theory of intersection types is *non-structural* in the sense that types with significantly different shapes (tree domains, when types are regarded as labeled trees) may be related, and this presents fundamental obstacles for bounding the depth of substitutions via any kind of standard occurs-check. Although "$1\frac{1}{2}$-distributivity" of $\rightarrow$ over $\cap$ may at first sight appear to be algebraically close to the ACUID-framework of [ANR04], the contravariant "$\frac{1}{2}$-distributivity" makes the theory of intersection types significantly different. We cannot exclude that some kind of restricted occurs-check might be possible, but our investigations lead us to believe that, in case it exists, it is likely to be very complicated, and we have been unable to find such a bounding principle. Hence, decidability remains a challenging open problem.

With regard to the exponential time lower bound, the results of [ANR04] (in fact, both the Exptime upper and lower bounds) rely essentially on reductions from unification modulo a set H of noncommuting homomorphisms (ACUIDH), which was shown to be Exptime-complete in [BN01]. The basic idea is to represent unification with distributivity to unification modulo homomorphisms by replacing $s * t$ by $h_s(t)$ where h_s is a homomorphism with respect to the AC(U)I-theory. However, again, such techniques fail in our case due to contravariance. The equational presentation of the theory of intersection types captures contravariant subtyping by the absorption axiom (written in the algebraic notation of [ANR04]): $s * t = s * t + (s + s') * t$. One could attempt to represent this axiom by $h_s(t) = h_s(t) + h_{s+s'}(t)$. But here the expression $h_{s+s'}(t)$ does not fall within the

homomorphic format, and it is therefore not clear how the homomorphic framework could be applied. Moreover, the bounding problem discussed above leads to the problem that it is not clear how the theory could be adequately represented using only a finite set of homomorphisms. We concluded that new methods are required in order to provide lower bounds for intersection type unification, and the route we present in this paper for the EXPTIME-lower bound is entirely different, relying on game theoretical results on tiling problems.

2.2. **Related work in type theory.** It may be surprising that computational properties (decidability, complexity) of the intersection type unification problem have not previously been systematically pursued *per se*. The theory of intersection type subtyping and its equational counterpart have rather been studied from semantic (operational and denotational) perspectives. Indeed, as mentioned already, the intersection type system captures deep operational properties of λ-terms, and undecidability of type checking and typability follows immediately. The theories of intersection type subtyping and equality studied here arose naturally out of model-theoretic considerations. For example, a fundamental result [Hin82, BCDC83] shows that intersection type subtyping and equality are sound and complete for set-theoretic containment in a class of λ-models: $\sigma \leq \tau$ holds, if and only if $[\![\sigma]\!]_v^{\mathcal{M}} \subseteq [\![\tau]\!]_v^{\mathcal{M}}$ for all models $\mathcal{M}$ in the class and valuations v. The intersection type unification problem can therefore also be endowed with semantic interpretations.

Several extensions and variations of the standard algebraic operations of unification studied here have been considered in connection with intersection type systems, foremostly motivated by questions related to notions of principality (principal types, principal typings, principal pairs) in such systems. Ronchi della Rocca, working from such motivations, defines a notion of unification in [Ron88] and gives a semi-decision procedure for the corresponding unification problem. But that problem involves operations (chains of substitutions together with special expansion operations) which are not present in the algebraic notion of unification we consider here. Similarly, so-called expansion variables with associated operations have been used by Kfoury and Wells to characterize principality properties [KW04] and so-called β-unification involving expansion variables has been shown to characterize strong normalization in the λ-calculus [Kfo99], see also [CG95, BZ05].

Since the contravariant "$\frac{1}{2}$-distributivity" makes the theory of intersection types significantly different compared to the well studied ACUID$_l$-unification, it is interesting to compare with problems that include left-contravariance of $\rightarrow$. One of them is the subtyping problem for second-order types, which is undecidable [TU96]. The authors point out that, if $\rightarrow$ is considered covariant in both arguments, then the corresponding subtyping problem becomes decidable. Finally, the above mentioned non-structural nature of the unification problem and the challenges it poses for bounding substitution depth could possibly be compared with challenges posed by the well known open problem of *non-structural subtype entailment*. For this problem a PSPACE lower bound is known [HR98], but despite many attempts over the past 20 years the question of decidability of non-structural subtype entailment remains open (see [Reh06] for more information and further references).

Summarizing the situation with regard to intersection type unification within type theory, it appears to hold an interesting and rather unexplored intermediate position: it is contained in many decision problems associated with intersection type systems, it is known to be expressive enough to capture certain restrictions of the type system, but it is not known whether it is decidable. It is therefore also a problem of importance for advancing our

understanding of restrictions of the intersection type system and computational properties of associated decision problems.

Organization of the paper. The remainder of this paper is organized as follows. Intersection types are introduced in Sec. 3 together with the standard theory of subtyping [BCDC83]. In Sec. 4 we briefly study the matching problem (one-sided unification) as a natural preparation for considering the unification problem. The unification problem is studied in Sec. 5, which contains our main result. We first introduce the unification problem and the equational theory of intersection types (Sec. 5.1) and then turn to the proof of the EXPTIME-lower bound. We introduce tiling games (Sec. 5.2) and prove EXPTIME-completeness of a special form of such ("spiral tiling games"), which is then used (Sec. 5.3) in our reduction to unification and satisfiability. Next we adapt the EXPTIME-lower bound to the ω-free fragment (Sec. 5.4) and inspect a different presentation of the underlying equational theory (Sec. 5.5). We provide some insights into a possible upper bound construction for unification in rank 1 (Sec. 6) and conclude the paper in Sec. 7.

3. INTERSECTION TYPES

Definition 1 ($\mathbb{T}$). *The set $\mathbb{T}$ of intersection types, ranged over by σ, τ, ρ, is given by*

$$\mathbb{T} \ni \sigma, \tau, \rho \ ::= \ a \mid \alpha \mid \omega \mid \sigma \to \tau \mid \sigma \cap \tau$$

where $a, b, c, \dots$ range over type constants drawn from the set $\mathbb{C}$, ω is a special (universal type) constant, and α, β, γ range over type variables drawn from the set $\mathbb{V}$.

As a matter of notational convention, function types associate to the right, and $\cap$ binds stronger than $\to$. A type $\tau \cap \sigma$ is said to have τ and σ as *components*. The *size* of a type τ, denoted by $|\tau|$, is the number of nodes in the syntax tree of τ.

Definition 2 (Subtyping $\leq$). *Subtyping $\leq$ is the least preorder (reflexive and transitive relation) over $\mathbb{T}$ (cf. [BCDC83]) such that*

$$\sigma \leq \omega, \quad \omega \leq \omega \to \omega, \quad \sigma \cap \tau \leq \sigma, \quad \sigma \cap \tau \leq \tau, \quad (\sigma \to \tau_1) \cap (\sigma \to \tau_2) \leq \sigma \to \tau_1 \cap \tau_2,$$

$$\text{if } \sigma \leq \tau_1 \text{ and } \sigma \leq \tau_2 \text{ then } \sigma \leq \tau_1 \cap \tau_2, \quad \text{if } \sigma_2 \leq \sigma_1 \text{ and } \tau_1 \leq \tau_2 \text{ then } \sigma_1 \to \tau_1 \leq \sigma_2 \to \tau_2$$

Type equality, written $\sigma = \tau$, holds when $\sigma \leq \tau$ and $\tau \leq \sigma$, thereby making $\leq$ a partial order over $\mathbb{T}$. We use $\equiv$ for syntactic identity. By the axioms of subtyping, $\cap$ is associative, commutative, idempotent and has the following distributivity properties

$$(\sigma \to \tau_1) \cap (\sigma \to \tau_2) = \sigma \to (\tau_1 \cap \tau_2) \qquad (\sigma_1 \to \tau_1) \cap (\sigma_2 \to \tau_2) \leq (\sigma_1 \cap \sigma_2) \to (\tau_1 \cap \tau_2)$$

We write $\bigcap_{i=1}^{n} \tau_i$ or $\bigcap_{i \in I} \tau_i$ or $\bigcap \{\tau_i \mid i \in I\}$ for an intersection of several components, where the empty intersection is identified with ω.

Observe that in the above subtyping definition type constants are treated the same as type variables. The motivation to distinguish type variables from type constants is twofold. First, practical scenarios often contain constants (e.g. `int` or `string`) that cannot be instantiated. Second, in the setting of E-unification in the sense of [BS01], variables that are subject to instantiation by terms in the underlying term algebra are distinct from any constructors of that algebra (in particular variables in the sense of [BCDC83]).

Using [BCDC83](Lemma 2.4.1) we syntactically define the set $\mathbb{T}^\omega$ of types equal to ω.

Definition 3 ($\mathbb{T}^\omega$). *The set $\mathbb{T}^\omega$ of types in $\mathbb{T}$ equal to ω is given by*

$$\mathbb{T}^\omega \ni \sigma^\omega, \tau^\omega ::= \omega \mid \sigma \to \tau^\omega \mid \sigma^\omega \cap \tau^\omega$$

Lemma 1. *For $\tau \in \mathbb{T}$ we have $\tau \in \mathbb{T}^\omega$ iff $\tau = \omega$.*

Lemma 2 (Beta-Soundness [BCDC83, BDS13]). *Given $\sigma = \bigcap_{i \in I} (\sigma_i \to \tau_i) \cap \bigcap_{j \in J} a_j \cap \bigcap_{k \in K} \alpha_k$,*

we have

    (i) *If $\sigma \le a$ for some $a \in \mathbb{C}$, then $a \equiv a_j$ for some $j \in J$.*
    (ii) *If $\sigma \le \alpha$ for some $\alpha \in \mathbb{V}$, then $\alpha \equiv \alpha_k$ for some $k \in K$.*
    (iii) *If $\sigma \le \sigma' \to \tau' \ne \omega$ for some $\sigma', \tau' \in \mathbb{T}$, then $I' = \{i \in I \mid \sigma' \le \sigma_i\} \ne \emptyset$ and $\bigcap_{i \in I'} \tau_i \le \tau'$.*

Let us briefly consider the problem of deciding the subtyping relation itself:

Problem 1. *(Subtyping) Given $\sigma, \tau \in \mathbb{T}$, does $\sigma \le \tau$ hold?*

The subtyping relation is known to be decidable in polynomial time [RU11]. The algorithm sketched in the proof of the following lemma gives an improved, quadratic upper bound (implemented in the Combinatory Logic Synthesis framework [BDD$^+$14] which is under continuous development).

Lemma 3. *Problem 1 (Subtyping) is decidable in time $\mathcal{O}(n^2)$ where n is the sum of the sizes of the input types σ and τ.*

Proof. For a polynomial time decision algorithm with a quartic upper bound see [RU11]. For a different approach with a quintic upper bound using rewriting see [Sta15]. However, a quadratic upper bound to decide $\sigma \le \tau$ is achievable using Lemmas 1 and 2. First, in linear time, subterms of σ and τ of the shape defined by $\mathbb{T}^\omega$ are replaced by ω. Second, in linear time, nested intersection are flattened using associativity of $\cap$ and components equal to ω are dropped. Third, in quadratic time, Lemma 2 is applied recursively using the additional property $\rho \le \bigcap_{i \in I} \tau_i$ iff $\rho \le \tau_i$ for $i \in I$.

For the explicit algorithmic analysis let $T(\sigma, \tau)$ be the running time to decide $\sigma \le \tau$. We show $T(\sigma, \tau) \le a|\sigma||\tau| - b = \mathcal{O}(|\sigma||\tau|)$ for some constants $a, b \in \mathbb{R}_+$ with $a \ge b$ by induction on the sum of the depths of syntax trees of σ and τ. The most challenging subcase in the algorithmic application of Lemma 2 is to decide whether $\bigcap_{i=1}^n \sigma_i \le \sigma' \to \tau'$ holds. We proceed as follows: let S be an empty linked list. For $i = 1 \ldots n$ such that $\sigma_i \equiv \rho_i \to \tau_i$ and $\sigma' \le \rho_i$ append τ_i to S. Finally, accept iff $\bigcap S \le \tau'$. Our goal is to bound $T(\sigma, \sigma' \to \tau')$ where $\sigma \equiv \bigcap_{i=1}^n \sigma_i$ by $a|\sigma||\sigma' \to \tau'| - b$. Let $\tau \equiv \bigcap(\tau_i \mid \sigma_i \equiv \rho_i \to \tau_i, \sigma' \le \rho_i)$ observing that the depth of the syntax tree of τ is strictly less than the depth of the syntax tree of σ. By the induction hypothesis for some constant u we have

$$
\begin{aligned}
T(\sigma, \sigma' \to \tau') &\le un + \sum_{\substack{i \in \{1, \ldots, n\} \\ \sigma_i \equiv \rho_i \to \tau_i}} T(\sigma', \rho_i) + T(\tau, \tau') \\
&\le un + \sum_{\substack{i \in \{1, \ldots, n\} \\ \sigma_i \equiv \rho_i \to \tau_i}} (a|\sigma'||\rho_i| - b) + a|\tau||\tau'| - b \\
&\le un + a|\sigma'| \sum_{\substack{i \in \{1, \ldots, n\} \\ \sigma_i \equiv \rho_i \to \tau_i}} |\rho_i| + a|\tau||\tau'| - b \\
&\le un + a|\sigma'||\sigma| + a|\sigma||\tau'| - b \\
&\le un + a|\sigma|(|\sigma' \to \tau'| - 1) - b \overset{a \ge u, |\sigma| \ge n}{\le} a|\sigma||\sigma' \to \tau'| - b
\end{aligned}
$$

In the above analysis, it is crucial that linked list concatenation is done in constant time in the construction of τ. Therefore, the invariant that $\cap$ is not nested and does not contain ω as components can be ensured in recursive calls. $\qquad\square$

We recapitulate the notion of paths and organized types introduced in [DMRU12] that coincides with the notion of factors in [Sta15].

Definition 4 (Paths $\mathbb{P}$). *The set $\mathbb{P}$ of paths in $\mathbb{T}$, ranged over by π, is given by*

$$\mathbb{P} \ni \pi ::= a \mid \alpha \mid \tau \to \pi$$

Definition 5 (Organized type). *A type τ is organized, if $\tau \equiv \omega$ or $\tau \equiv \bigcap_{i \in I} \pi_i$ for some paths π_i for $i \in I$.*

A type can be organized (transformed to an equivalent organized type) in polynomial time [DMRU12]. Additionally, an organized type is not necessarily normalized in the sense of [Hin82]. Normalization can lead to an exponential blow-up of type size, whereas organization does not as illustrated by the following examples.

Example 1. *Let $\sigma \equiv ((a \cap b \to a \cap b) \to a \cap b) \to a \cap b$. The equivalent organized type is $\sigma = \big(((a \cap b \to a \cap b) \to a \cap b) \to a\big) \cap \big(((a \cap b \to a \cap b) \to a \cap b) \to b\big)$, whereas the equivalent normalized type in the sense of [Hin82] is*

$$\sigma = \big(((a \cap b \to a) \cap (a \cap b \to b) \to a) \cap ((a \cap b \to a) \cap (a \cap b \to b) \to b) \to a\big)$$
$$\cap \big(((a \cap b \to a) \cap (a \cap b \to b) \to a) \cap ((a \cap b \to a) \cap (a \cap b \to b) \to b) \to b\big)$$

In general, normalization can lead to an exponential blow-up of type size because arguments of arrow types are normalized recursively.

Example 2. *Let $\sigma \equiv a \to \bigcap_{i=0}^{1}(b_i \to \bigcap_{j=0}^{1}(c_j \to \bigcap_{k=0}^{1} d_k))$. The equivalent organized type coincides with the equivalent normalized type $\sigma = \bigcap_{(i,j,k)\in\{0,1\}^3} (a \to b_i \to c_j \to d_k)$. Still, when generalized, this example does not lead to an exponential blow-up of type size by organization because the size of σ is already exponential in the depth of its syntax tree.*

Lemma 4. *Given two organized types $\sigma \equiv \bigcap_{i \in I} \pi_i$ and $\tau \equiv \bigcap_{j \in J} \pi_j$, we have $\sigma \leq \tau$ iff for all $j \in J$ there exists an $i \in I$ with $\pi_i \leq \pi_j$.*

Corollary 1. *Given a path $\pi \in \mathbb{P}$ and types σ, τ, we have $\sigma \cap \tau \leq \pi$ iff $\sigma \leq \pi$ or $\tau \leq \pi$.*

For the sake of completeness, we recall the corresponding type assignment system from [BCDC83], sometimes also called **BCD** in literature. A basis (also called context) is a finite set $\Gamma = \{x_1 : \tau_1, \ldots, x_n : \tau_n\}$, where the variables x_i are pairwise distinct; we set $\mathrm{dom}(\Gamma) = \{x_1, \ldots, x_n\}$ and we write $\Gamma, x : \tau$ for $\Gamma \cup \{x : \tau\}$, where $x \notin \mathrm{dom}(\Gamma)$.

Definition 6 (Type Assignment). **BCD** *type assignment is given by the following rules*

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\,(Ax) \qquad \frac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash \lambda x.e : \sigma \to \tau}\,(\to\!I) \qquad \frac{\Gamma \vdash e : \sigma \to \tau \qquad \Gamma \vdash e' : \sigma}{\Gamma \vdash (e\ e') : \tau}\,(\to\!E)$$

$$\frac{}{\Gamma \vdash e : \omega}\,(\omega) \qquad \frac{\Gamma \vdash e : \sigma \qquad \Gamma \vdash e : \tau}{\Gamma \vdash e : \sigma \cap \tau}\,(\cap I) \qquad \frac{\Gamma \vdash e : \sigma \qquad \sigma \leq \tau}{\Gamma \vdash e : \tau}\,(\leq)$$

4. Intersection type matching

In order to understand the unification problem it is useful first to investigate its restriction to matching (one-sided unification). Intersection type matching occurs naturally during proof search in intersection type systems and is known to be NP-complete [DMR13]. We strengthen this result by showing that the problem remains so even when restricted to the fixed-parameter case where only a single type variable and only a single constant is used in the input.

For $\tau \in \mathbb{T}$ let $\mathrm{Var}(\tau) \subseteq \mathbb{V}$ denote the set of variables occurring in τ.

Problem 2 (Matching). *Given a set of constraints $C = \{\sigma_1 \stackrel{.}{\leq} \tau_1, \ldots, \sigma_n \stackrel{.}{\leq} \tau_n\}$, where for each $i \in \{1, \ldots, n\}$ we have $\mathrm{Var}(\sigma_i) = \emptyset$ or $\mathrm{Var}(\tau_i) = \emptyset$, is there a substitution $S \colon \mathbb{V} \to \mathbb{T}$ such that $S(\sigma_i) \leq S(\tau_i)$ for $1 \leq i \leq n$?*

We say that a substitution S satisfies $\{\sigma_1 \stackrel{.}{\leq} \tau_1, \ldots, \sigma_n \stackrel{.}{\leq} \tau_n\}$ if $S(\sigma_i) \leq S(\tau_i)$ for $1 \leq i \leq n$.

Problem 3 (One-Sided Unification). *Given a set of constraints $C = \{\sigma_1 \stackrel{.}{=} \tau_1, \ldots, \sigma_n \stackrel{.}{=} \tau_n\}$, where for each $i \in \{1, \ldots, n\}$ we have $\mathrm{Var}(\sigma_i) = \emptyset$ or $\mathrm{Var}(\tau_i) = \emptyset$, is there a substitution $S \colon \mathbb{V} \to \mathbb{T}$ such that $S(\sigma_i) = S(\tau_i)$ for $1 \leq i \leq n$?*

Any matching constraint set $C = \{\sigma_1 \stackrel{.}{\leq} \tau_1, \ldots, \sigma_n \stackrel{.}{\leq} \tau_n\}$ can be reduced to a single matching (resp. one-sided unification) constraint $\sigma \stackrel{.}{\leq} \tau$ (resp. $\sigma \cap \tau \stackrel{.}{=} \sigma$) with $\mathrm{Var}(\sigma) = \emptyset$ by fixing a type constant $\bullet \in \mathbb{C}$, defining

$$(\sigma_i', \tau_i') = \begin{cases} (\sigma_i \to \bullet, \tau_i \to \bullet) & \text{if } \mathrm{Var}(\sigma_i) = \emptyset \\ (\tau_i, \sigma_i) & \text{if } \mathrm{Var}(\tau_i) = \emptyset \end{cases} \quad \text{for } 1 \leq i \leq n$$

and taking $\sigma \equiv \sigma_1' \to \ldots \to \sigma_n' \to \bullet$ and $\tau \equiv \tau_1' \to \ldots \to \tau_n' \to \bullet$. By Lemma 2, for any substitution S we have $S(\sigma) \leq S(\tau)$ (resp. $S(\sigma \cap \tau) = S(\sigma)$) iff $S(\sigma_i) \leq S(\tau_i)$ for $1 \leq i \leq n$. Observe that $\mathrm{Var}(\sigma) = \emptyset$. Therefore, matching and one-sided unification coincide and remain NP-complete even when restricted to single constraints.

In [DMR13] the lower bound for matching is shown by reduction from 3-SAT and requires two type variables $\alpha_x, \alpha_{\neg x}$ for each propositional variable x. Since 3-SAT, parameterized by the number of propositional variables, is fixed parameter tractable, we naturally ask whether the same holds for matching (resp. one-sided unification) parameterized by the number of type variables.

Lemma 5. *Problem 2 (Matching) is* NP-*hard even if only a single type variable and a single constant is used in the input.*

Proof. We fix a 3-SAT instance F containing clauses $(L_1 \vee L_2 \vee L_3) \in F$ over propositional variables V where L_i is either x or $\neg x$ for some $x \in V$. We reduce satisfiability of F to matching with one type variable α. First, we fix a set of type constants $B = V \cup \{\neg x \mid x \in V\}$ and the type constant $\bullet$. Let $\sigma_x \equiv \bigcap(B \setminus \{\neg x\})$ and $\sigma_{\neg x} \equiv \bigcap(B \setminus \{x\})$ for $x \in V$. We construct the set C containing the following constraints

for $x \in V$ (consistency) :

$$((\sigma_{\neg x} \to \bullet) \to (\neg x \to \bullet)) \cap ((\sigma_x \to \bullet) \to (x \to \bullet)) \stackrel{.}{\leq} (\alpha \to \bullet) \to (\alpha \to \bullet)$$

for $(L_1 \vee L_2 \vee L_3) \in F$ (validity) :

$$(L_1 \to \bullet) \cap (L_2 \to \bullet) \cap (L_3 \to \bullet) \stackrel{.}{\leq} \alpha \to \bullet$$

If F is satisfied by a valuation v, then the substitution $\alpha \mapsto \bigcap_{v(x)=1} x \cap \bigcap_{v(x)=0} \neg x$ satisfies C.

If C is satisfied by a substitution S, then by Corollary 1 and the consistency constraints we have either $\sigma_{\neg x} \leq S(\alpha) \leq \neg x$ or $\sigma_x \leq S(\alpha) \leq x$ for $x \in V$. A valuation v constructed according to these cases satisfies each clause in F due to Corollary 1 and the validity constraints.

Instead of using constants $\{a_1, \ldots, a_k, \bullet\}$ for an instance of the matching problem, encode $[a_i] = \underbrace{\bullet \to \ldots \to \bullet \to}_{i \text{ times}} \bullet$ for $1 \leq i \leq k$ in the proof. Using this technique, a single type constant $\bullet$ is sufficient. $\square$

Combining Lemma 5 with the reduction in [DMR13] we conclude that neither restricting substitutions to the shape $S : \{\alpha\} \to \mathbb{T}$ nor restricting to the shape $S : \mathbb{V} \to \mathbb{C}$ (atomic substitutions, mapping variables to type constants) reduces the complexity of matching.

5. Intersection type unification

5.1. The unification problem.

Problem 4 (Satisfiability). *Given a set of constraints $C = \{\sigma_1 \dot{\leq} \tau_1, \ldots, \sigma_n \dot{\leq} \tau_n\}$, is there a substitution $S \colon \mathbb{V} \to \mathbb{T}$ such that $S(\sigma_i) \leq S(\tau_i)$ for $1 \leq i \leq n$?*

Problem 5 (Unification). *Given a set of constraints $C = \{\sigma_1 \dot{=} \tau_1, \ldots, \sigma_n \dot{=} \tau_n\}$, is there a substitution $S \colon \mathbb{V} \to \mathbb{T}$ such that $S(\sigma_i) = S(\tau_i)$ for $1 \leq i \leq n$?*

Since for any $\sigma, \tau \in \mathbb{T}$ and any substitution S we have $S(\sigma) \leq S(\tau) \iff S(\sigma) \cap S(\tau) = S(\sigma)$, satisfiability and unification are (trivially) equivalent. Similarly to matching (resp. one-sided unification) restricting satisfiability (resp. unification) to single constraints does not reduce its complexity.

We now provide a number of observations that give some insight into the type-theoretical and combinatorial expressive power of unification.

Consider a combinatory logic with intersection types [HS08, DCH92] with arbitrary basis $\mathfrak{B}$, that is, a finite set of combinator symbols $F, G, \ldots$ with types $\tau_F, \tau_G, \ldots$. Such a system is given by the rules (applicative fragment) $(\to E), (\cap I), (\leq)$ of Definition 6 together with a rule assigning types $S(\tau_F)$ to the combinator symbol F for any substitution S. Write $\mathfrak{B} \vdash E : \tau$ for derivability of the type τ for the combinatory expression E in this system.

Example 3. *Let $\mathfrak{B} = \{F : (\sigma \to \tau) \to a, G : \alpha \to \alpha\}$, where wlog. $\alpha \notin \text{Var}(\sigma) \cup \text{Var}(\tau)$. In this scenario, type-checking $\mathfrak{B} \vdash F\,G : a$ is equivalent to solving the satisfiability problem $\alpha \to \alpha \dot{\leq} \sigma \to \tau$, or equivalently, the unification problem $\sigma \cap \tau \dot{=} \sigma$, because we need to find substitutions $S, S_1, \ldots S_n$ for some $n \in \mathbb{N}$ such that*

$$\bigcap_{i=1}^n S_i(\alpha \to \alpha) \leq S(\sigma \to \tau)$$

$$\overset{Lem. 2}{\iff} S(\sigma) \leq \bigcap_{i \in I} S_i(\alpha) \text{ and } \bigcap_{i \in I} S_i(\alpha) \leq S(\tau) \text{ for some } I \subseteq \{1, \ldots, n\}$$

$$\iff S(\alpha \to \alpha) \leq S(\sigma \to \tau) \text{ setting } S(\alpha) = \bigcap_{i \in I} S_i(\alpha)$$

Write $\mathfrak{B} \vdash^* E : \tau$ if $\mathfrak{B} \vdash E : \tau$ is derivable without the intersection introduction rule ($\cap$I). This restriction occupies an interesting "intermediate" position: generalized to arbitrary bases $\mathfrak{B}$, it is the combinatory logic that subsumes the **BCD**-calculus without intersection introduction [KT95, RU12] and the one-dimensional fragment [DR17] of the intersection type system. For example, using the following basis (cf. [DCH92])

$$\mathfrak{B} = \{S : (\alpha \to \beta \to \gamma) \to (\alpha' \to \beta) \to \alpha \cap \alpha' \to \gamma, K : \alpha \to \beta \to \alpha, I : \alpha \to \alpha\}$$

the system $\vdash^*$ is sufficient to type $S\,I\,I$, i.e., the SKI-combinatory logic equivalent of the λ-term $\lambda x.(x\,x)$, which is not typable in simple types. The full expressive power of $\vdash^*$ is subject to ongoing research as a candidate combinatory logic suitable for software synthesis [Reh13, DMR14].

Example 4. *Typability with respect to $\vdash^*$ is equivalent to unification. Let $\mathfrak{B} = \{F_1 : \tau_1, \ldots F_n : \tau_n\}$, where wlog. $\mathrm{Var}(\tau_i) \cap \mathrm{Var}(\tau_j) = \emptyset$ for $i \neq j$, and let E be a combinatory term over $\mathfrak{B}$. We want to know whether there is a type τ such that $\mathfrak{B} \vdash^* E : \tau$. Conveniently, without arrow introduction or intersection introduction, type derivations are entirely term driven up to $\leq$ and initial substitutions.*

For any combinatory term E' and type τ' we define

$$f(E', \tau') = \begin{cases} \{\tau_i \overset{.}{\leq} \tau'\} & \text{if } E' = F_i \text{ for some } i \in \{1, \ldots, n\} \\ f(E_1, \alpha \to \beta) \cup f(E_2, \alpha) \cup \{\beta \overset{.}{\leq} \tau'\} & \text{if } E' = E_1\,E_2 \text{ and } \alpha, \beta \text{ are fresh} \end{cases}$$

Given E' and τ', by induction on E' we have that the satisfiability problem instance $f(E', \tau')$ has a solution S iff $\mathfrak{B} \vdash^* E' : S(\tau')$. Therefore, $f(E, \alpha)$, where α is fresh, has a solution iff E is typable in the basis $\mathfrak{B}$.

Conversely, given a satisfiability problem $\sigma \overset{.}{\leq} \tau$, we consider typability of $F\,G$ in the basis $\mathfrak{B} = \{F : \tau \to a, G : \sigma\}$.

In addition to the previously illustrated non-structurality of unification, the following examples illustrate further potential difficulties. Example 5 shows that unification is not finitary, i.e. unification instances may have infinitely many most general unifiers.

Example 5. *Consider the unification constraint $a \to a \to (\beta \cap b) \overset{.}{=} \beta \cap \alpha$ and some solution S. We have $S(\beta) = a \to a \to \bigcap_{i \in I} \pi_i$ such that for all $i \in I$ either $\pi_i = b$ or $\pi_i = a \to a \to \pi_j$ for some $j \in I$. Therefore, $S(\beta)$ (and consequently $S(\alpha)$) contains paths that may be arbitrary long, end in the constant b and have an even number of "a"s as arguments. As a result, solutions to the above constraint cannot be built by specialization from a finite set of (most general) unifiers.*

The following Example 6 shows that certain unification instances may require all solutions to be of exponential size.

Example 6. *Consider prime numbers $2, 3$ and the following unification constraints*

$$a \to a \to (\beta_2 \cap b) \overset{.}{=} \beta_2 \cap \alpha, \qquad a \to a \to a \to (\beta_3 \cap b) \overset{.}{=} \beta_3 \cap \alpha$$

Let S be any solution to the above constraints. Similar to Example 5, S necessarily satisfies the following properties

- *$S(\beta_2) \neq \omega$, otherwise we would have $S(\alpha) = a \to a \to b$ which by the second constraint implies $a \to a \to a \to (S(\beta_3) \cap b) \leq a \to a \to b$ and therefore $a \to (S(\beta_3) \cap b) \leq b$, which is a contradiction.*

- $S(\beta_3) \neq \omega$, otherwise we would have $S(\alpha) = a \rightarrow a \rightarrow a \rightarrow b$ which by the first constraint implies $a \rightarrow a \rightarrow (S(\beta_2) \cap b) \leq a \rightarrow a \rightarrow a \rightarrow b$, therefore $S(\beta_2) \leq a \rightarrow b$. Again, by the first constraint we obtain $a \rightarrow a \rightarrow (S(\beta_2) \cap b) \leq a \rightarrow b$, which is a contradiction.
- $S(\alpha) \neq \omega$, otherwise we would have $S(\beta_2) \leq a \rightarrow a \rightarrow b$, therefore $S(\beta_2) \leq a \rightarrow a \rightarrow S(\beta_2) \leq a \rightarrow a \rightarrow a \rightarrow a \rightarrow b$ and, inductively, $S(\beta_2) \leq a \rightarrow \ldots \rightarrow a \rightarrow b$ with an even number of "a"s, which by Lemma 4 would in total imply a solution of infinite size.
- $S(\beta_2) = \bigcap_{i \in I} \pi_i$ such that $\pi_i = a \rightarrow \ldots \rightarrow a \rightarrow b$ with an even number of "a"s similar to Example 5.
- $S(\beta_3) = \bigcap_{j \in J} \pi_j$ such that $\pi_j = a \rightarrow \ldots \rightarrow a \rightarrow b$ with a number of "a"s that is a multiple of 3.
- $S(\alpha) = \bigcap_{k \in K} \pi_k$ such that $\pi_k = a \rightarrow \ldots \rightarrow a \rightarrow b$ with a number of "a"s that is a multiple of 2 and 3.

One possible solution S' to the above constraints is

$$S'(\beta_2) = (a \rightarrow a \rightarrow b) \cap (a \rightarrow a \rightarrow a \rightarrow a \rightarrow b)$$

$$S'(\beta_3) = a \rightarrow a \rightarrow a \rightarrow b$$

$$S'(\alpha) = a \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow b$$

In sum, the size of $S(\alpha)$ in any solution is greater than the product of our initial primes 2 and 3. By adding an additional constraint $a \rightarrow a \rightarrow a \rightarrow a \rightarrow a \rightarrow (\beta_5 \cap b) \doteq \beta_5 \cap \alpha$, the size of $S(\alpha)$ is at least $2 \cdot 3 \cdot 5$ in any solution, growing exponentially with additional constraints. As a side note, similarly to Example 5, the above unification instance does not have a most general unifier that produces all possible solutions by specialization.

An axiomatization of the equational theory of intersection type subtyping (without ω) is derived in [Sta15]. We add two additional axioms **(U)** and **(RE)** in the following Definition 7 to incorporate the universal type ω.

Definition 7 (ACIUD$_l$REAB). *The equational theory* ACIUD$_l$REAB *is given by*

(A) $\sigma \cap (\tau \cap \rho) \sim (\sigma \cap \tau) \cap \rho$
(C) $\sigma \cap \tau \sim \tau \cap \sigma$
(I) $\sigma \cap \sigma \sim \sigma$
(U) $\sigma \cap \omega \sim \sigma$
(D$_l$) $(\sigma \rightarrow \tau) \cap (\sigma \rightarrow \tau') \sim \sigma \rightarrow \tau \cap \tau'$
(RE) $\omega \sim \omega \rightarrow \omega$
(AB) $\sigma \rightarrow \tau \sim (\sigma \rightarrow \tau) \cap (\sigma \cap \sigma' \rightarrow \tau)$

The recursion axiom **(RE)** captures the recursive nature of ω and the absorption axiom **(AB)** captures contra-variance.

Lemma 6. *Given $\sigma, \tau \in \mathbb{T}$ we have $\sigma = \tau$ iff $\sigma \sim \tau$.*

Proof. "$\Longrightarrow$": Induction on the depth of the derivation of $\sigma \leq \tau$ to show $\sigma \cap \tau \sim \sigma$. The most interesting subcase is $\sigma_1 \rightarrow \tau_1 \leq \sigma_2 \rightarrow \tau_2$ under the assumptions $\sigma_2 \leq \sigma_1$ and $\tau_1 \leq \tau_2$.

In the subcase we have

$$
\sigma_1 \to \tau_1 \overset{(AB)}{\sim} (\sigma_1 \to \tau_1) \cap (\sigma_1 \cap \sigma_2 \to \tau_1)
$$
$$
\overset{IH}{\sim} (\sigma_1 \to \tau_1) \cap (\sigma_1 \cap \sigma_2 \to \tau_1 \cap \tau_2)
$$
$$
\overset{(D_l)}{\sim} (\sigma_1 \to \tau_1) \cap (\sigma_1 \cap \sigma_2 \to \tau_1) \cap (\sigma_1 \cap \sigma_2 \to \tau_2)
$$
$$
\overset{IH}{\sim} (\sigma_1 \to \tau_1) \cap (\sigma_1 \cap \sigma_2 \to \tau_1) \cap (\sigma_2 \to \tau_2)
$$
$$
\overset{(AB)}{\sim} (\sigma_1 \to \tau_1) \cap (\sigma_2 \to \tau_2)
$$

Finally, $\sigma \leq \tau$ and $\tau \leq \sigma$ imply $\sigma \sim \sigma \cap \tau \sim \tau$.

"$\Longleftarrow$": Each axiom of $\mathrm{ACIUD}_l\mathrm{ReAb}$ is derivable using subtyping. Again, the most interesting subcase is $\sigma \to \tau \sim (\sigma \to \tau) \cap (\sigma \cap \sigma' \to \tau)$. First, by contravariance we have $\sigma \to \tau \leq \sigma \cap \sigma' \to \tau$, therefore $\sigma \to \tau \leq (\sigma \to \tau) \cap (\sigma \cap \sigma' \to \tau)$. Second, $(\sigma \to \tau) \cap (\sigma \cap \sigma' \to \tau) \leq \sigma \to \tau$. Therefore, $\sigma \to \tau = (\sigma \to \tau) \cap (\sigma \cap \sigma' \to \tau)$. $\qquad\square$

The absorption axiom **(AB)** distinguishes the above theory $\mathrm{ACIUD}_l\mathrm{ReAb}$ from theories studied in literature. As discussed in the introduction, the closest equational theory ACIUD_l of [ANR04], which assumes $\omega \to \sigma \sim \omega \sim \sigma \to \omega$ and has no equivalent of the absorption axiom **(AB)**, is Exptime-complete. Unfortunately, the absorption axiom prevents the approaches presented in [ANR04, ANR03] as shown by the following examples.

Example 7. *Consider $\alpha \cap (\alpha \to a) \doteq \alpha$ (or equivalently $\alpha \overset{\cdot}{\leq} \alpha \to a$). A DAG-based (or "occurs-check"-based) approach cannot stratify such a constraint since any solution $S(\alpha)$ contains at least one subterm $S(\alpha) \to a$ and therefore a circular dependency. Interestingly, using absorption there is a solution $S(\alpha) = \omega \to a$.*

Example 8. *Consider $\alpha \cap (((\alpha \to c) \cap b) \to a) \doteq \alpha$ (or equivalently $\alpha \overset{\cdot}{\leq} ((\alpha \to c) \cap b) \to a$). In contrast to the previous example, all occurrences of α are positive. Again, we have a circular dependency. Using absorption there is a solution $S(\alpha) = b \to a$.*

The constraint $\alpha \cap (\alpha \to a) \doteq \alpha$ (or equivalently $\alpha \overset{\cdot}{\leq} \alpha \to a$) in Example 7 has several interesting properties. For any σ such that $\sigma \leq \sigma \to a$ we may type the (Church-style annotated) lambda-term $\lambda x^\sigma.(x^{\sigma \to a} \; x^\sigma)$ using subtyping. Additionally, there are several structurally different solution for the above constraint:

- $S_1(\alpha) = \omega \to a$ (requires **(AB)**, **(U)**, **(C)**)
- $S_2(\alpha) = a \cap (a \to a)$ (requires **(AB)**)
- $S_3(\alpha) = ((a \cap (a \to a)) \to a) \to a$ (requires **(AB)**, **(D$_l$)**, **(A)**, **(C)**)

S_1 requires the absorption axiom as well as the unit axiom, whereas S_2 and S_3 do not require any axiom containing ω. Therefore, ω may play a crucial role in solving circular constraints but is not strictly necessary. The main difference between S_2 and S_3 is that S_3 uses distributivity and, more importantly, only one path (or factor in the sense of [Sta15]) instead of an intersection of two paths. This is especially interesting in the light of Lemma 4. In particular, restricting substitutions to paths does not avoid satisfiable circular constraints.

5.2. **Tiling games.** In this section we introduce a special kind of domino tiling game, referred to as two-player corridor tiling games, for which the problem of existence of winning strategies is Exptime-complete [Chl86]. We then show that Exptime-completeness is preserved when tilings are restricted to a particular "spiral" shape, which will be used to prove our Exptime-lower bound for intersection type unification in Sec. 5.3.

Definition 8 (Tiling System)**.** *A tiling system is a tuple $(D, H, V, \bar{b}, \bar{t}, n)$, where*
- *D is a finite set of tiles (also called dominoes)*
- *$H, V \subseteq D \times D$ are horizontal and vertical constraints*
- *$\bar{b}, \bar{t} \in D^n$ are n-tuples of tiles*
- *n is a unary encoded natural number*

Definition 9 (Corridor Tiling)**.** *Given a tiling system $(D, H, V, \bar{b}, \bar{t}, n)$, a* corridor tiling *is a mapping $\lambda : \{1, \ldots, l\} \times \{1, \ldots, n\} \to D$ for some $l \in \mathbb{N}$ such that*
- *$\bar{b} = (\lambda(1,1), \ldots, \lambda(1,n))$ (correct bottom row)*
- *$\bar{t} = (\lambda(l,1), \ldots, \lambda(l,n))$ (correct top row)*
- *for $i \in \{1, \ldots, l\}$ and $j \in \{1, \ldots, n-1\}$ we have $(\lambda(i,j), \lambda(i,j+1)) \in H$, i.e. the horizontal constraints are satisfied*
- *for $i \in \{1, \ldots, l-1\}$ and $j \in \{1, \ldots, n\}$ we have $(\lambda(i,j), \lambda(i+1,j)) \in V$, i.e. the vertical constraints are satisfied*

Given a tiling system $(D, H, V, \bar{b}, \bar{t}, n)$, a *Two-Player Corridor Tiling* game consists of two players (*Constructor* and *Spoiler*). The game is played on an $\mathbb{N} \times \{1, \ldots, n\}$ board and starts with the bottom row $\bar{b}$. Each player places copies of tiles in turn starting with Constructor. While Constructor tries to construct a corridor tiling, Spoiler tries to prevent it. Constructor wins if Spoiler makes an illegal move (with respect to H or V), or when a correct corridor tiling is completed. We say Constructor has winning strategy, if he can win no matter what Spoiler does.

Lemma 7 (Chlebus [Chl86])**.** *The decision problem whether Constructor has a winning strategy in a given two-player corridor tiling game is* EXPTIME-*complete.*

Instead of directly encoding a Two-Player Corridor Tiling into intersection type satisfiability, we introduce a slightly different game that is played out as sequences instead of corridors. The main goal is to get rid of several structural constraints of corridors for a more accessible construction of a spiral where each new tile has a neighboring previous tile.

Definition 10 (Spiral Tiling)**.** *Given a tiling system $(D, H, V, \bar{b}, \bar{t}, n)$, a* spiral tiling *is a sequence $d_1 \ldots d_m \in D^m$ for some $m \in \mathbb{N}$ such that*
- *$\bar{b} = d_1 \ldots d_n$ (correct prefix)*
- *$\bar{t} = d_{m-n+1} \ldots d_m$ (correct suffix)*
- *$(d_i, d_{i+1}) \in H$ for $1 \leq i \leq m-1$ (horizontal constraints)*
- *$(d_i, d_{i+n}) \in V$ for $1 \leq i \leq m-n$ (vertical constraints)*

Given a tiling system $(D, H, V, \bar{b}, \bar{t}, n)$ a *Two-Player Spiral Tiling* game, played by Constructor and Spoiler, starts with the sequence $\bar{b}$. Each player adds a copy of a tile to the end of the current sequence taking turns starting with Constructor. While Constructor tries to construct a spiral tiling, Spoiler tries to prevent it. Constructor wins if Spoiler makes an illegal move (with respect to H or V), or when a correct spiral tiling is completed. Again, we are interested in whether Constructor has a winning strategy.

The main difference between a corridor tiling and a spiral tiling is the lack of individual rows. While a tile at the beginning of the new row of a corridor is not constrained by the previously placed tile, in a spiral each new tile is constrained by the previous one. Additionally, any corridor tiling contains $l \cdot n$ tiles for some l; a spiral tiling does not obey this restriction. To clarify essential aspects of spiral tilings, consider the following examples.

Example 9. *Consider the tiling system with $D = \{a, b\}$, $H = \{(a, b), (b, a), (b, b)\}$, $V = D^2$, $\bar{b} = aaa$, $\bar{t} = bbb$ and $n = 3$. Constructor does not have a winning strategy in the corresponding Two-Player Spiral Tiling game. During the game on Spoiler's turn there are two possibilities. In case the current sequence ends in a, Spoiler is forced/allowed to append b, which does not result in the suffix bbb. Regardless of Constructor's next tile the suffix is not bbb. In case the current sequence ends in b, Spoiler is allowed to append a and, similarly, Constructor is not able to produce a spiral tiling.*

Example 10. *Consider the tiling system with $D = \{a, b\}$, $H = D^2$, $V = D^2 \setminus \{(b, a)\}$, $\bar{b} = aaaaa$, $\bar{t} = bbbbb$ and $n = 5$. Constructor has a winning strategy in the corresponding Two-Player Spiral Tiling game by always appending b. Due to V, if the current sequence has the tile b at position i, then it will have the tile b at any later position $i + 5j$ for $j \in \mathbb{N}$. Therefore, after the first nine turns of the game all positions $1 + 5j$, $3 + 5j$, $5 + 5j$, $7 + 5j$ and $9 + 5j$ for $j \in \mathbb{N}$ will have the tile b regardless of Spoiler's moves. Since those positions will form a suffix $bbbbb$ after 9 turns, Constructor is able to produce a spiral tiling.*

Lemma 8. *The decision problem whether Constructor has a winning strategy in a given two-player spiral tiling game is* EXPTIME-*complete.*

Proof.
Lower Bound: Given a tiling system $T = (D, H, V, (b_1, \ldots, b_n), (t_1, \ldots, t_n), n)$, let

$$D' = D \,\dot\cup\, \{\#\}$$
$$H' = H \,\dot\cup\, \{(d, \#) \mid d \in D'\} \cup \{(\#, d) \mid d \in D'\}$$
$$V' = V \,\dot\cup\, \{(\#, \#)\}$$
$$T' = (D', H', V', (b_1, \ldots, b_n, \#, \#), (t_1, \ldots, t_n, \#, \#), n + 2)$$

We show that Constructor has a winning strategy for Two-Player Corridor Tiling in T iff he has a winning strategy for Two-Player Spiral Tiling in T'.

By construction, both players are allowed to and have to place the tile $\#$ at exactly the turns $i(n + 2) - 1$ and $i(n + 2)$ for $i \geq 1$. Therefore, a winning strategy does not branch nor end at those turns. Additionally, a correct spiral tiling ends in two consecutive $\#$ tiles, therefore necessarily contains $i(n + 2)$ tiles.

From any correct corridor tiling $\lambda : \{1, \ldots, l\} \times \{1, \ldots, n\}$ for T we construct a spiral tiling $d_1 \ldots d_{l(n+2)}$ for T' by

$$d_k = \begin{cases} \lambda(i, j) & \text{if } k = (i - 1)(n + 2) + j \text{ and } i \geq 1 \text{ and } 1 \leq j \leq n \\ \# & \text{if } k = (i - 1)(n + 2) + j \text{ and } i \geq 1 \text{ and either } j = 0 \text{ or } j = n + 1 \end{cases}$$

From a correct spiral tiling $d_1 \ldots d_{l(n+2)}$ for T' we construct a corridor tiling $\lambda : \{1, \ldots, l\} \times \{1, \ldots, n\}$ for T by $\lambda(i, j) = d_{(i-1)(n+2)+j}$.

In particular, Constructor's winning strategy (skipping/adding the forced $\#$ turns) is exactly the same for both games.

Upper Bound: Computation in APSPACE = EXPTIME (similar to Two-Player Corridor Tiling). To continue, only the n previously placed tiles have to be considered. $\square$

5.3. **Exptime lower bound.** We now prove our main result, that the intersection type unification problem is EXPTIME-hard. The proof will be by reduction from spiral tiling games (Lemma 8) to the intersection type satisfiability problem.

Let $T = (D, H, V, \bar{b} = b_1 \ldots b_n, \bar{t} = t_1 \ldots t_n, n)$ be a tiling system. Wlog. $(b_i, b_{i+1}) \in H$ and $(t_i, t_{i+1}) \in H$ for $1 \leq 1 < n$, otherwise the given prefix $\bar{b}$ or the given suffix $\bar{t}$ would already violate constraints on consecutive tiles and Constructor would have no chance to construct a spiral tiling. We fix the set of type constants $\mathbb{C} = D \,\dot\cup\, \{\bullet\}$ and variables $\mathbb{V} = \{\alpha\} \cup \{\beta_d \mid d \in D\}$ and construct the following set of constraints $\mathscr{C}_T$:

(i) $\sigma_\perp^H \cap \sigma_\perp^V \cap \sigma_t \cap \bigcap_{d \in D} \beta_d \,\dot\leq\, \sigma_b \cap \bigcap_{d' \in D} \bigcap_{d \in D} (d' \to d \to \beta_d)$ \hfill (Game moves)

(ii) $\bigcap_{(d', d) \in H} (d \to d' \to \alpha) \,\dot\leq\, \bigcap_{d \in D} (d \to \beta_d)$ \hfill (d respects H)

(iii) $\bigcap_{(d', d) \in V} (d \to \underbrace{\omega \to \ldots \to \omega}_{n-1 \text{ times}} \to d' \to \alpha) \,\dot\leq\, \bigcap_{d \in D} (d \to \beta_d)$ \hfill (d respects V)

where

$$\sigma_b \equiv b_n \to \ldots \to b_1 \to \bullet \qquad\qquad \text{(Initial state)}$$

$$\sigma_t \equiv (t_n \to \ldots \to t_1 \to \alpha) \cap (\omega \to t_n \to \ldots \to t_1 \to \alpha) \qquad \text{(Final states)}$$

$$\sigma_\perp^H \equiv \bigcap_{(d, d') \in D \times D \backslash H} (d' \to d \to \alpha) \qquad\qquad \text{(d' violates H)}$$

$$\sigma_\perp^V \equiv \bigcap_{(d, d') \in D \times D \backslash V} (d' \to \underbrace{\omega \to \ldots \to \omega}_{n-1 \text{ times}} \to d \to \alpha) \qquad \text{(d' violates V)}$$

Intuitively, we use Lemma 4 to realize alternation. The rhs of (i) represents an intersection of all board positions which Constructor may face. For all such positions he needs to find a suitable move by choosing a path on the lhs of (i). He can either state that the Spoiler's last move violates H (resp. V) choosing $\sigma_\perp^H$ (resp. $\sigma_\perp^V$), or that the game is finished choosing σ_t, or he can pick his next move $d \in D$ choosing β_d. Intuitively, β_d captures all board positions in which Constructor decides to place d next. Note that on the rhs of (i) in the type $d' \to d \to \beta_d$ the tile d' is not constrained (representing all possible Spoiler's moves) while the tile d is constrained to the index of β_d, i.e. Constructor's previous choice. Therefore, by picking a move d Constructor faces board positions that arise from the previous position extended by d and each possible d'. Constraints (ii) and (iii) ensure that whenever Constructor picks his next move $d \in D$ choosing β_d he has to respect H and V.

We show that Constructor has a winning strategy for two-player spiral tiling in T iff the constraint system $\mathscr{C}_T$ is satisfiable. To represent game positions as types, we define the mapping $[\cdot] : D^* \to \mathbb{T}$ such that $[\epsilon] = \bullet$ and $[\bar{s}d] = d \to [\bar{s}]$ for $\bar{s} \in D^*$ and $d \in D$. To improve readability, we use the notation $\sigma \overset{\phi}{\leq} \tau$, where ϕ is a hint why the inequality holds.

Lemma 9. *Let T be a tiling system. If Constructor has a winning strategy in a two-player spiral tiling game in T, then the constraint system $\mathscr{C}_T$ is satisfiable.*

Proof. Assume that Constructor has a winning strategy that is represented by a labeled tree $f : \text{dom}(f) \to \{C, S\}$ where

- $\text{dom}(f) \subseteq D^*$ is finite and prefix-closed, i.e. $\bar{u}\bar{v} \in \text{dom}(f)$ implies $\bar{u} \in \text{dom}(f)$.
- $\text{depth}(f) = \max\{k \mid d_1 \ldots d_k \in \text{dom}(f)\}$.
- For $\bar{s} = d_1 \ldots d_k \in \text{dom}(f)$ we have $f(\bar{s}) = C$ if k is even and $f(\bar{s}) = S$ if k is odd, i.e. C (resp. S) places a tile after an even (resp. odd) number of turns.

- For $\bar{s} \in \mathrm{dom}(f)$ such that $f(\bar{s}) = S$ we have $\bar{s}d' \in \mathrm{dom}(f)$ for all $d' \in D$, i.e. the strategy has to consider all (possibly illegal) Spoiler's moves.
- For $\bar{s} \in \mathrm{dom}(f)$ such that $f(\bar{s}) = C$ we have either
  - There exists exactly one $d \in D$ such that $\bar{s}d \in \mathrm{dom}(f)$ and $\bar{b}\bar{s} = \bar{u}d_1 \ldots d_n$ for some $\bar{u} \in D^*$ and $d_1, \ldots, d_n \in D$ with $(d_n, d) \in H$ and $(d_1, d) \in V$, i.e. Constructor's next move is d which respects H and V.
  - $\bar{s}d \notin \mathrm{dom}(f)$ for all $d \in D$ and either
    * $\bar{b}\bar{s} = \bar{u}\bar{t}$ for some $\bar{u} \in D^*$, i.e. Constructor states that the game is finished.
    * $\bar{b}\bar{s} = \bar{u}\bar{t}d'$ for some $\bar{u} \in D^*$ and $d' \in D$, i.e. Constructor states that Spoiler's last move d' is illegal because the game already ended.
    * $\bar{b}\bar{s} = \bar{u}dd'$ for some $\bar{u} \in D^*$, $d, d' \in D$ such that $(d, d') \notin H$, i.e. Constructor states that Spoiler's last move d' violates H.
    * $\bar{b}\bar{s} = \bar{u}d\bar{v}d'$ for some $\bar{u} \in D^*$, $\bar{v} \in D^{n-1}$, $d, d' \in D$ such that $(d, d') \notin V$, i.e. Constructor states that Spoiler's last move d' violates V.

We construct the following substitution S

$$S(\alpha) = \bigcap_{\substack{d_1 \ldots d_k = \bar{s} \in D^* \\ k \le \mathrm{depth}(f)+n}} [\bar{s}] \quad \text{and} \quad S(\beta_d) = \bigcap_{\substack{\bar{s} \in f^{-1}(C) \\ \bar{s}d \in \mathrm{dom}(f)}} [\bar{b}\bar{s}] \quad \text{for } d \in D$$

We verify that the individual inequalities hold.

- $S(\sigma_\perp^H \cap \sigma_\perp^V \cap \sigma_t \cap \bigcap_{d \in D} \beta_d) \le \sigma_b$:

  if $\bar{b} = \bar{t}$, then $S(\sigma_t) \le \sigma_b$. Otherwise, according to f, there exists a $d \in D$ such that $d \in \mathrm{dom}(f)$. Therefore, $S(\beta_d) \le [\bar{b}] \equiv \sigma_b$.
- $S(\sigma_\perp^H \cap \sigma_\perp^V \cap \sigma_t \cap \bigcap_{d \in D} \beta_d) \le S(d' \to d \to \beta_d)$ for all $d, d' \in D$:

  we show that for any $\pi = d_k \to \ldots \to d_1 \to \sigma_b = [\bar{b}\bar{s}]$ such that $d_1 \ldots d_k = \bar{s} \in f^{-1}(C)$ and $\bar{s}d \in \mathrm{dom}(f)$ we have $S(\sigma_\perp^H \cap \sigma_\perp^V \cap \sigma_t \cap \bigcap_{d \in D} \beta_d) \le d' \to d \to \pi \equiv [\bar{b}\bar{s}dd']$. Since $\bar{s}d \in \mathrm{dom}(f)$ and $f(\bar{s}) = C$ we have $\bar{s}dd' \in \mathrm{dom}(f)$ and $f(\bar{s}dd') = C$. According to f we have either
  - $\bar{s}dd'd'' \in \mathrm{dom}(f)$ for some $d'' \in D$. Therefore, $S(\beta_{d''}) \le d' \to d \to \pi \equiv [\bar{b}\bar{s}dd']$
  - or
    * $\bar{b}\bar{s}dd' = \bar{u}\bar{t}$ for some $\bar{u} \in D^*$, then $S(\sigma_t) \le [\bar{u}\bar{t}] \equiv [\bar{b}\bar{s}dd']$.
    * $\bar{b}\bar{s}dd' = \bar{u}\bar{t}d'$ for some $\bar{u} \in D^*$, then $S(\sigma_t) \le [\bar{u}\bar{t}d'] \equiv [\bar{b}\bar{s}dd']$.
    * $(d, d') \notin H$, then $S(\sigma_\perp^H) \le [\bar{b}\bar{s}dd']$.
    * $\bar{b}\bar{s}dd' = \bar{u}d''\bar{v}d'$ for some $\bar{u} \in D^*$, $\bar{v} \in D^{n-1}$, $d'' \in D$ such that $(d'', d') \notin V$, then $S(\sigma_\perp^V) \le [\bar{u}d''\bar{v}d'] \equiv [\bar{b}\bar{s}dd']$.
- $S(\bigcap_{(d',d) \in H} (d \to d' \to \alpha)) \le S(d \to \beta_d)$ for all $d \in D$:

  fix any $\pi = d_k \to \ldots \to d_1 \to \sigma_b = [\bar{b}\bar{s}]$ such that $d_1 \ldots d_k = \bar{s} \in f^{-1}(C)$ and $\bar{s}d \in \mathrm{dom}(f)$. Since $f(\bar{s}) = C$ and $\bar{s}d \in \mathrm{dom}(f)$, we have $\bar{b}\bar{s}d = \bar{u}d'd$ for some $\bar{u} \in D^*$ and $d' \in D$ such that $(d', d) \in H$. We have

$$S(\bigcap_{(d',d) \in H} (d \to d' \to \alpha)) \overset{S(\alpha) \le [\bar{u}]}{\le} [\bar{u}d'd] \equiv [\bar{b}\bar{s}d] \equiv d \to \pi$$

- $S(\bigcap_{(d',d) \in V} (d \to \underbrace{\omega \to \ldots \to \omega}_{n-1 \text{ times}} \to d' \to \alpha)) \le S(d \to \beta_d)$ for all $d \in D$:

  fix any $\pi = d_k \to \ldots \to d_1 \to \sigma_b = [\bar{b}\bar{s}]$ such that $d_1 \ldots d_k = \bar{s} \in f^{-1}(C)$ and $\bar{s}d \in \mathrm{dom}(f)$.

Since $f(\bar{s}) = C$ and $\bar{s}d \in \mathrm{dom}(f)$, we have $\bar{b}\bar{s}d = \bar{u}d'\bar{v}d$ for some $\bar{u} \in D^*$, $\bar{v} \in D^{n-1}$ and $d' \in D$ such that $(d', d) \in V$. We have

$$S(\bigcap_{(d',d)\in V} (d \to \underbrace{\omega \to \ldots \to \omega \to}_{n-1 \text{ times}} d' \to \alpha)) \overset{S(\alpha)\leq[\bar{u}]}{\leq} [\bar{u}d'\bar{v}d] \equiv [\bar{b}\bar{s}d] \equiv d \to \pi \qquad \square$$

Lemma 10. *Let T be a tiling system. If the constraint system $\mathscr{C}_T$ is satisfiable, then Constructor has a winning strategy in a two-player spiral tiling game in T.*

Proof. Assume that there exists a substitution S that satisfies the constraints $\mathscr{C}_T$ and wlog. uses only organized types. Constructor wins the game regardless of Spoiler's moves as follows: Let $\tau \equiv S(\sigma_b \cap \bigcap_{d'\in D} \bigcap_{d\in D} (d' \to d \to \beta_d))$, i.e. the rhs of (i). The initial game position is $\bar{b}$ and we have $\tau \leq \sigma_b \equiv [\bar{b}]$. We consider a single turn of Constructor from any game position $\bar{s} \in D^*$ that he may face.

Assume $(\star)$ that the current game position $\bar{s}$ satisfies $\tau \leq [\bar{s}]$. Due to (i) and Corollary 1 we have the following cases

- If $S(\sigma_\perp^H) \leq [\bar{s}]$, then there exist $d, d' \in D$ such that $(d, d') \notin H$ and for some path π we have $d' \to d \to \pi \leq [\bar{s}]$. Therefore, $\bar{s} = \bar{u}dd'$ for some $\bar{u} \in D^*$ and Constructor wins because Spoiler's last move d' violates H. As a side note, his is not possible for $\bar{s} = \bar{b}$ since $\bar{b}$ is horizontally consistent.
- If $S(\sigma_\perp^V) \leq [\bar{s}]$, then there exist $d, d' \in D$ such that $(d, d') \notin V$ and for some path π we have $d' \to \underbrace{\omega \to \ldots \to \omega \to}_{n-1 \text{ times}} d \to \pi \leq [\bar{s}]$. Therefore, $\bar{s} = \bar{u}d\bar{v}d'$ for some $\bar{u} \in D^*$, $\bar{v} \in D^{n-1}$
  and Constructor wins because Spoiler's last move d' violates V. As a side note, this is not possible for $\bar{s} = \bar{b}$ since $\bar{b}$ is too short.
- If $S(\sigma_t) \leq [\bar{s}]$, then Constructor wins because $t_n \to \ldots \to t_1 \to \pi \leq [\bar{s}]$ for some path π implies the winning condition $\bar{s} = \bar{u}\bar{t}$ for some $\bar{u} \in D^*$. Alternatively $\omega \to t_n \to \ldots \to t_1 \to \pi \leq [\bar{s}]$ for some path π implies $\bar{s} = \bar{u}\bar{t}d'$ for some $\bar{u} \in D^*$ and $d' \in D$, therefore Spoiler's last move d' was illegal because the game already ended.
- If $S(\beta_d) \leq [\bar{s}]$ for some $d \in D$, then Constructor may safely place d as the next tile. We verify consistency wrt. H and V. First, due to (ii) there exists a path $d \to d' \to \pi$ for some $d' \in D$ with $(d', d) \in H$ such that

$$d \to d' \to \pi \overset{(ii)}{\leq} S(d \to \beta_d) \overset{S(\beta_d)\leq[\bar{s}]}{\leq} [\bar{s}d]$$

  Therefore, $\bar{s}d = \bar{u}d'd$ for some $\bar{u} \in D^*$ and placing d does not violate H. Second, due to (iii) there exists a path $d \to \underbrace{\omega \to \ldots \to \omega \to}_{n-1 \text{ times}} d' \to \pi'$ for some $d' \in D$ with $(d', d) \in V$
  such that

$$d \to \underbrace{\omega \to \ldots \to \omega \to}_{n-1 \text{ times}} d' \to \pi' \overset{(iii)}{\leq} S(d \to \beta_d) \overset{S(\beta_d)\leq[\bar{s}]}{\leq} [\bar{s}d]$$

  Therefore, $\bar{s}d = \bar{u}d'\bar{v}d$ for some $\bar{u} \in D^*$, $\bar{v} \in D^{n-1}$ and placing d does not violate V.

In neither case Constructor loses. If Constructor placed the tile $d \in D$, in which case we have $S(\beta_d) \leq [\bar{s}]$, Spoiler may place any tile $d' \in D$. The new game position is $\bar{s}dd'$ and our initial assumption $(\star)$ is inductively satisfied

$$\tau \leq S(d' \to d \to \beta_d) \overset{S(\beta_d)\leq[\bar{s}]}{\leq} [\bar{s}dd']$$

Therefore, we may apply our argument inductively. Additionally, the game necessarily ends after a finite number of turns: if $\tau = \bigcap_{i \in I}(\sigma_1^i \to \ldots \sigma_{l_i}^i \to c_i)$ (for some index set I, integers $l_i \geq 0$ for $i \in I$, types σ_j^i for $i \in I$ and $1 \leq j \leq l_i$ and type constants c_i for $i \in I$), then $(\star)$, i.e. $\tau \leq [\bar{s}]$, cannot be satisfied by any $\bar{s} \in D^k$ with $k > \max\{l_i \mid i \in I\}$. $\qquad\square$

Theorem 1. *The intersection type satisfiability problem and the intersection type unification problem are* EXPTIME-*hard.*

Proof. Immediate from Lemma 8, Lemma 9 and Lemma 10, since all reduction steps are evidently computable in polynomial time. Moreover, satisfiability is polynomial time equivalent to unification. $\qquad\square$

Corollary 2. *Satisfiability and unification are* EXPTIME-*hard even in the presence of only one constant.*

Proof. Encode d_i by $[d_i] = \underbrace{\bullet \to \ldots \to \bullet \to}_{i \text{ times}} \bullet$ for $1 \leq i \leq k$ in the original proofs. $\qquad\square$

Corollary 3. *Satisfiability and unification are* EXPTIME-*hard even if the codomain of substitutions is restricted to types of rank 1, i.e. intersections of simple types.*

Proof. In the proof of Lemma 9 each variable is substituted by an intersection of simple types, i.e. a rank 1 intersection type. $\qquad\square$

Interestingly, the axiom **(RE)** $\omega \sim \omega \to \omega$ (resp. $\omega \leq \omega \to \omega$) is not necessary for the EXPTIME lower bound proof, while the axioms **(U)** and **(AB)** (resp. $\sigma \leq \omega$ and $\omega \to \tau \leq \sigma \to \tau$ derived from contravariance) play a crucial role in the construction of $\sigma_\perp^V$ to capture an exponential number of cases. Naturally, we would like to know whether the special properties of ω are necessary for the EXPTIME lower bound, which we will consider in the following Sec. 5.4.

5.4. **Exptime lower bound without** ω. In AC unification theories [BS01] it is often the case that presence of unit does not change unification complexity. This holds true for the EXPTIME lower bound of satisfiability in the ω-free fragment of intersection types and the corresponding ACID_lAB unification. We observe that in the construction of constraints $\mathscr{C}_T$ in Section 5.3 it suffices to consider substitutions that map variables to intersections of types of the shape $d_1 \to \ldots \to d_k \to \bullet$ where $d_1, \ldots, d_k \in D$. Therefore, we may replace the use of ω in $\mathscr{C}_T$ by iterated distributivity. Generally, this is possible because the codomain of substitutions necessary for the lower bound construction may be restricted to a fixed rank.

Given a tiling system $T = (D, H, V, \bar{b} = b_1 \ldots b_n, \bar{t} = t_1 \ldots t_n, n)$ we fix the set of type constants $\mathbb{C} = D \,\dot{\cup}\, \{\bullet\}$ and variables $\mathbb{V} = \{\alpha\} \cup \{\beta_d \mid d \in D\} \cup \{\gamma_d^i \mid d \in D, i \in \{1, \ldots, n\}\}$ and construct the following set of constraints $\mathscr{C}_T'$:

(i) $\sigma_\perp^H \cap \sigma_\perp^V \cap \sigma_t \cap \bigcap_{d \in D} \beta_d \,\dot{\leq}\, \sigma_b \cap \bigcap_{d' \in D} \bigcap_{d \in D} (d' \to d \to \beta_d)$ $\qquad\qquad$ (Game moves)

(ii) $\sigma_\top^H \,\dot{\leq}\, \bigcap_{d \in D} (d \to \beta_d)$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad$ (d respects H)

(iii) $\sigma_\top^V \,\dot{\leq}\, \bigcap_{d \in D} (d \to \beta_d)$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad$ (d respects V)

(iv) $\gamma_d^1 \,\dot{=}\, d \to \alpha$ $\qquad\qquad\qquad\qquad\qquad\qquad\qquad$ (for each $d \in D$, d is last)

(v) $\gamma_d^i \,\dot{=}\, \bigcap_{d_i \in V} (d_i \to \gamma_d^{i-1})$ $\qquad\quad$ (for each $d \in D$ and each $i = 2 \ldots n$, d is i-th to last)

$$\text{where} \qquad \sigma_b \equiv b_n \to \ldots \to b_1 \to \bullet \qquad\qquad\qquad\qquad \text{(Initial state)}$$

$$\sigma_t \equiv (t_n \to \ldots \to t_1 \to \alpha) \cap \bigcap_{d' \in D} (d' \to t_n \to \ldots \to t_1 \to \alpha) \qquad \text{(Final states)}$$

$$\sigma_\perp^H \equiv \bigcap_{(d,d') \in D \times D \setminus H} (d' \to d \to \alpha) \qquad\qquad\qquad (d' \text{ violates } H)$$

$$\sigma_\perp^V \equiv \bigcap_{(d,d') \in D \times D \setminus V} (d' \to \gamma_d^n) \qquad\qquad\qquad (d' \text{ violates } V)$$

$$\sigma_\top^H \equiv \bigcap_{(d',d) \in H} (d \to d' \to \alpha) \qquad\qquad\qquad (d \text{ respects } H)$$

$$\sigma_\top^V \equiv \bigcap_{(d',d) \in V} (d \to \gamma_{d'}^n) \qquad\qquad\qquad (d \text{ respects } V)$$

The above constraints $\mathscr{C}_T'$ differ from $\mathscr{C}_T$ in two aspects. First, final states in σ_t are described explicitly for each possible last tile $d' \in D$, instead of using ω as a wildcard. Second, in order to describe a sequence in which d' is last and d is $n-1$ tiles before d, i.e., d' is vertically adjacent to d in a tiling, instead of using a type such as $d' \to \omega \to \ldots \omega \to d \to \alpha$ we use $d' \to \gamma_d^n$. By a simple inductive proof we have that for substitution S satisfying constraints (iv) and (v) if $S(d' \to \gamma_d^i) \leq d_1 \to \ldots \to d_k \to \bullet$, then $d_1 = d'$ and $d_{n+1} = d$. Additionally, intersections in γ_d^i appear either on the top level or in arrow target position. Therefore, using distributivity, it still suffices to consider only substitutions that map variables to intersections of types of the shape $d_1 \to \ldots \to d_k \to \bullet$ where $d_1, \ldots, d_k \in D$.

Theorem 2. *The intersection type satisfiability problem in the ω-free fragment and the intersection type unification problem in the ω-free fragment are* EXPTIME-*hard.*

Proof. Same as the proof of Theorem 1 using the set of constraints $\mathscr{C}_T'$. $\qquad\square$

5.5. **Alternative Presentation.** The most peculiarly shaped axiom in ACIUD$_l$REAB is absorption. Therefore, we would like to explore a different presentation of ACIUD$_l$REAB unification that does not rely on such oddly shaped axiom. For this purpose, we introduce the least upper bound $\vee$ (cf. [KT95, Theorem 2.9]) as a syntactic meta-operator to intersection types. In particular, we can define $\vee : \mathbb{T} \times \mathbb{T} \to \mathbb{T}$ such that for all $\sigma, \tau, \rho \in \mathbb{T}$ we have $\sigma \leq \rho$ and $\tau \leq \rho$ iff $\sigma \vee \tau \leq \rho$. As a result, the axiom (AB) is subsumed by lattice-theoretic operations. Note that using the meta-operator $\vee$ is different from using union types $\cup$, which are strictly more expressive.

Considering the way the least upper bound operation is defined for intersection types, we split the original absorption axiom into lattice-theoretic absorption (AB$_\cap$) and contravariant right-distributivity (D$_r^-$) in the following Definition 11.

Definition 11 (ACIUD$^-$REAB$_\cap$). *The equational theory* ACIUD$^-$REAB$_\cap$ *is given by*
(A) $\sigma \cap (\tau \cap \rho) \approx (\sigma \cap \tau) \cap \rho$
(C) $\sigma \cap \tau \approx \tau \cap \sigma$
(I) $\sigma \cap \sigma \approx \sigma$
(U) $\sigma \cap \omega \approx \sigma$
(D$_l$) $(\sigma \to \tau) \cap (\sigma \to \tau') \approx \sigma \to \tau \cap \tau'$
(RE) $\omega \approx \omega \to \omega$

$(\mathbf{AB}_\cap)$ $\sigma \approx \sigma \cap (\sigma \vee \tau)$
$(\mathbf{D}_r^-)$ $(\sigma \to \tau) \vee (\sigma' \to \tau) \approx (\sigma \cap \sigma') \to \tau$

The theory $\mathrm{ACIUD}^-\mathrm{REAB}_\cap$ interacts with $\vee$ syntactically. Since we are interested in $\vee$-free intersection types, we need to verify that $\approx$ is identical to $\sim$ on $\mathbb{T}$.

Lemma 11. *Given $\sigma, \tau \in \mathbb{T}$ we have $\sigma \sim \tau$ in $\mathrm{ACIUD}_l\mathrm{REAB}$ iff $\sigma \approx \tau$ in $\mathrm{ACIUD}^-\mathrm{REAB}_\cap$.*

Proof. "$\Longrightarrow$": The theory $\mathrm{ACIUD}^-\mathrm{REAB}_\cap$ splits the original absorption axiom into lattice-theoretic absorption $(\mathrm{AB}_\cap)$ and contravariant right-distributivity (D_r^-) as follows

$$\sigma \to \tau \overset{\mathbf{AB}_\cap}{\approx} (\sigma \to \tau) \cap ((\sigma \to \tau) \vee (\sigma' \to \tau)) \overset{\mathbf{D_r^-}}{\approx} (\sigma \to \tau) \cap (\sigma \cap \sigma' \to \tau)$$

Therefore, $\sigma \sim \tau$ implies $\sigma \approx \tau$ by replacing $(\mathbf{AB})$ as shown above.

"$\Longleftarrow$": First, we show that the additional axioms $(\mathbf{AB}_\cap)$ and $(\mathbf{D}_r^-)$ are derivable using subtyping and the least upper bound property of $\vee$.

$(\mathbf{AB}_\cap)$ First, $\sigma \leq \sigma$ and $\sigma \leq \sigma \vee \tau$ implies $\sigma \leq \sigma \cap (\sigma \vee \tau)$. Second, $\sigma \cap (\sigma \vee \tau) \leq \sigma$. Therefore, $\sigma = \sigma \cap (\sigma \vee \tau)$.

$(\mathbf{D}_r^-)$ Due to [KT95, Theorem 2.9] $\sigma \cap \sigma' \to \tau$ is the least upper bound of $\sigma \to \tau$ and $\sigma' \to \tau$. By least upper bound uniqueness, we have

$$(\sigma \to \tau) \vee (\sigma' \to \tau) = (\sigma \cap \sigma') \to \tau$$

As a result, $\sigma \approx \tau$ implies $\sigma = \tau$, therefore $\sigma \sim \tau$ by Lemma 6. $\qquad\square$

Corollary 4. *Given $\sigma, \tau \in \mathbb{T}$ we have $\sigma = \tau$ iff $\sigma \approx \tau$.*

Remark 1. *Since $\cap$ is greatest lower bound wrt. $\leq$ and $\vee$ is least upper bound wrt. $\leq$, we may add lattice theoretic properties of $\vee$ (such as ACI and mutual distributivity of $\cap$ and $\vee$) to $\mathrm{ACIUD}^-\mathrm{REAB}_\cap$ without changing its expressiveness.*

It remains open whether one can exploit additional lattice theoretic properties introduced by $\vee$ to bound unification complexity similar to the type tallying problem [CNXA15].

6. On Decidability of Rank 1 Unification

Since problems in the intersection type system are often sensitive to rank [Urz09], it is interesting to inspect decidability of rank restricted unification.

By *rank 1 unification* (resp. *rank 1 satisfiability*) we denote unification (resp. satisfiability) where the substitution codomain is restricted to rank 1 types. To be precise, each variable can be mapped to a (possibly empty, i.e. ω) intersection of simple types (not containing intersections or ω).

In this section we sketch the reduction of rank 1 unification to set constraints with projections in finite sets with cardinalities. Unfortunately, only the complexity of set constraints with projections in infinite sets is known to be NEXPTIME [CP94] and complexity in finite sets is unknown. For an overview of properties of set constraints with projections see [CP94]. The purpose of the sketched reduction is to provide a toolbox to translate the oddities of intersection type unification in a uniform set theoretic setting, raising the question regarding its decidability. Although intersection types have a set-theoretic interpretation in a λ-calculus model [BCDC83] where subtyping coincides with set inclusion at the level of λ-*terms*, it is not evident how the the shape of legal solutions (possibly infinite sets of λ-terms of an arbitrary type) can be expressed in the syntax of set constraints with

projections in the sense of [CP94]. The fundamental observation that distinguishes rank 1 unification is that rank 1 subtyping degenerates to set inclusion (cf. Lemma 13) at the level of *types*.

Given a set $\mathcal{C}$ of constraints, we non-deterministically transform $\mathcal{C}$ to a set of set constrains with projections $\mathcal{D}$ such that $\mathcal{C}$ has a solution iff $\mathcal{D}$ has a solution. The signature of the resulting set constraints is that of simple types, where src (resp. tgt) is the first (resp. second) projection of $\rightarrow$ lifted to sets. The cardinality constraint $|\alpha| = 1$ implies that any solution requires the set substituted for α to contain exactly one element.

To construct $\mathcal{D}$, first choose a set V of variables and replace each occurrence of each variable in V by ω. Second, normalize (recursively organize) all occurring types. Third, iteratively apply the following rules until $\mathcal{C}$ is empty (given by highest to lowest priority)

(1) Remove $\sigma \overset{.}{\leq} \tau$ from $\mathcal{C}$ if $\sigma \leq \tau$.
(2) Remove $\varphi \overset{.}{\leq} \alpha$ from $\mathcal{C}$ if φ is a simple type and add $\alpha = \{\varphi\}$ to $\mathcal{D}$ (cf. Lemma 12).
(3) Remove $\alpha \overset{.}{\leq} \varphi$ from $\mathcal{C}$ if φ is a simple type and add $\{\varphi\} \subseteq \alpha$ to $\mathcal{D}$ (cf. Lemma 13).
(4) Remove $\alpha \overset{.}{\leq} \beta$ from $\mathcal{C}$ and add $\beta \subseteq \alpha$ to $\mathcal{D}$ (cf. Lemma 13).
(5) If $\omega \overset{.}{\leq} \tau$ in $\mathcal{C}$ and $\tau \notin \mathbb{T}^\omega$ abort with "no solution" (cf Lemma 1).
(6) Replace $\sigma \overset{.}{\leq} \bigcap_{i \in I} \tau_i$ in $\mathcal{C}$ by $\{\sigma \overset{.}{\leq} \tau_i \mid i \in I\}$.
(7) If $a \overset{.}{\leq} \sigma \rightarrow \tau$, $\sigma \rightarrow \tau \overset{.}{\leq} a$ or $a \overset{.}{\leq} b$ in $\mathcal{C}$ then abort with "no solution".
(8) Replace $\bigcap_{i \in I} \tau_i \overset{.}{\leq} \sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow a$ in $\mathcal{C}$ by $\tau_i \overset{.}{\leq} \sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow a$ where $i \in I$ is chosen non-deterministically (cf. Corollary 1).
(9) Replace $\bigcap_{i \in I} \tau_i \overset{.}{\leq} \sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \alpha$ in $\mathcal{C}$ by $\{\tau_i \overset{.}{\leq} \sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \alpha_i \mid i \in I'\}$ and add $\alpha = \bigcup_{i \in I'} \alpha_i$ to $\mathcal{D}$ where $\emptyset \neq I' \subseteq I$ is chosen non-deterministically and α_i for $i \in I'$ are fresh (cf. Lemma 4).
(10) Replace $\sigma_1 \rightarrow \tau_1 \overset{.}{\leq} \sigma_2 \rightarrow \tau_2$ in $\mathcal{C}$ by $\{\sigma_2 \overset{.}{\leq} \sigma_1, \tau_1 \overset{.}{\leq} \tau_2\}$.
(11) Replace $\alpha \overset{.}{\leq} \sigma \rightarrow \tau$ in $\mathcal{C}$ by $\{\sigma \overset{.}{\leq} \beta, \gamma \overset{.}{\leq} \tau\}$ and add $\{\delta \subseteq \alpha, \beta = \mathrm{src}(\delta), \gamma = \mathrm{tgt}(\delta)\}$ to $\mathcal{D}$ where β, γ, δ are fresh (cf. Lemma 14).
(12) Replace $\omega \rightarrow \tau \overset{.}{\leq} \alpha$ in $\mathcal{C}$ by $\tau \overset{.}{\leq} \beta$ and add $\alpha \subseteq \gamma \rightarrow \beta$ to $\mathcal{D}$ where β, γ are fresh.
(13) Replace $\bigcap_{i \in I} \sigma_i \rightarrow \tau \overset{.}{\leq} \alpha$ in $\mathcal{C}$ by $\{\sigma_i \rightarrow \tau \overset{.}{\leq} \alpha \mid i \in I\}$ (cf. [KT95, Theorem 2.9]).
(14) Replace $(\sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow a) \rightarrow \tau \overset{.}{\leq} \alpha$ in $\mathcal{C}$ by $\{\sigma_i \overset{.}{\leq} \beta_i \mid i = 1 \ldots n\} \cup \{\tau \overset{.}{\leq} \gamma\}$ and add $\alpha \subseteq (\beta_1 \rightarrow \ldots \rightarrow \beta_n \rightarrow a) \rightarrow \gamma$ to $\mathcal{D}$ where β_i, γ are fresh for $i = 1 \ldots n$ (cf. Lemma 15).
(15) Replace $(\sigma_1 \rightarrow \ldots \rightarrow \sigma_n \rightarrow \delta) \rightarrow \tau \overset{.}{\leq} \alpha$ in $\mathcal{C}$ by $\{\sigma_i \overset{.}{\leq} \beta_i \mid i = 1 \ldots n\} \cup \{\tau \overset{.}{\leq} \gamma\}$ and add $\{\alpha \subseteq (\beta_1 \rightarrow \ldots \rightarrow \beta_n \rightarrow \delta) \rightarrow \gamma, |\delta| = 1\}$ to $\mathcal{D}$ where β_i, γ are fresh for $i = 1 \ldots n$ (cf. Lemma 15).

The above sketched construction contains justifications for its soundness and completeness. A crucial aspect for rank 1 unification is that subtyping restricted to simple types degenerates to equality.

Lemma 12. *For $\varphi, \psi \in \mathbb{T}$ such that φ and ψ are simple types we have $\varphi \leq \psi$ iff $\varphi \equiv \psi$.*

Proof. By induction on the sum of depths of φ and ψ using Lemma 2. $\square$

Since a simple type is also a path (cf. Definition 4), subtyping restricted to intersections of simple types degenerates to set inclusion.

Lemma 13. *For $\sigma, \tau \in \mathbb{T}$ such that $\sigma \equiv \bigcap_{i \in I} \varphi_i$ and $\tau \equiv \bigcap_{j \in J} \psi_j$ are intersections of simple types we have $\sigma \leq \tau$ iff $\{\psi_j \mid j \in J\} \subseteq \{\varphi_i \mid i \in I\}$.*

Proof. Consequence of Lemma 12 and Lemma 4. $\square$

An other crucial property of intersection type subtyping that allows to strictly strengthen inequalities is expressed by the following Lemma 14.

Lemma 14. *We have $\bigcap_{i \in I}(\sigma_i \to \tau_i) \le \sigma' \to \tau'$ iff there exists an index set $I' \subseteq I$ such that $\bigcap_{i \in I'} \sigma_i \to \bigcap_{i \in I'} \tau_i \le \sigma' \to \tau'$.*

Proof. Immediate consequence of Lemma 2 *(iii)*. $\qquad \square$

Finally, the most important case (15) in the above construction relies on a more involved argument. In particular, we need to restrict the cardinality of solutions due to the following Lemma 15.

Lemma 15. *We have $(\sigma_1 \to \ldots \to \sigma_n \to \bigcap_{i \in I} \varphi_i) \to \tau \le \bigcap_{j \in J} \psi_j$ where φ_i, ψ_j are simple types for $i \in I \ne \emptyset$, $j \in J \ne \emptyset$ iff for each $j \in J$ we have $\psi_j \equiv (\psi_j^1 \to \ldots \to \psi_j^n \to \psi_j^{n+1}) \to \psi_j^{n+2}$ for some $\psi_j^1, \ldots, \psi_j^{n+2}$ and the following conditions hold*

   (i) $\tau \le \bigcap_{j \in J} \psi_j^{n+2}$
   (ii) $\sigma_k \le \bigcap_{j \in J} \psi_j^k$ *for* $k = 1 \ldots n$
   (iii) $\phi_i \equiv \psi_j^{n+1}$ *for all* $i \in I$ *and* $j \in J$

Proof. "$\Longrightarrow$": For each $j \in J$ the shape $\psi_j \equiv (\psi_j^1 \to \ldots \to \psi_j^n \to \psi_j^{n+1}) \to \psi_j^{n+2}$ for some $\psi_j^1, \ldots, \psi_j^{n+2}$ is due to Lemma 2 with the resulting conditions

(1) for all $j \in J$ we have $\tau \le \psi_j^{n+2}$

(2) for all $j \in J$ and $k = 1 \ldots n$ we have $\sigma_k \le \psi_j^k$

(3) for all $j \in J$ we have $\psi_j^{n+1} \le \bigcap_{i \in I} \varphi_i$

Conditions (1) and (2) are equivalent to conditions *(i)* and *(ii)*. By Lemma 13 we have $\{\varphi_i \mid i \in I\} \subseteq \{\psi_j^{n+1}\}$ for each $j \in J$ which implies condition *(iii)*.

    "$\Longleftarrow$": Immediately by definition of $\le$. $\qquad \square$

7. Conclusion and future work

We have positioned the algebraic intersection type unification problem as a natural object of study within unification theory and type theory, and we have provided the first nontrivial lower bound in the standard and the ω-free fragment showing that the problem is of high complexity. Our EXPTIME-lower bound uses game-theoretic methods which may be useful for making further progress on the main open question for future work, that of decidability. Next steps include exploring variants and restrictions. Variants of intersection type subtyping theories (see [BDS13]) give rise to a *family* of intersection type unification problems yet to be studied. We conjecture an NEXPTIME-upper bound for rank 1 restricted unification, in which variables are substituted by intersections of simple types. Since organized rank 1 subtyping corresponds to set inclusion, one can reduce a rank 1 unification problem to satisfiability of set constraints with projections [CP94] in finite sets. Unfortunately, standard set constraint interpretations may contain infinite sets, which is why this approach needs further investigation.

Acknowledgement

References

[ANR03] S. Anantharaman, P. Narendran, and M. Rusinowitch. Acid-unification is NEXPTIME-decidable. In *Mathematical Foundations of Computer Science 2003*, pages 169–178. Springer, 2003.

[ANR04] S. Anantharaman, P. Narendran, and M. Rusinowitch. Unification Modulo ACUI Plus Distributivity Axioms. *Journal of Automated Reasoning*, 33(1):1–28, 2004.

[BCDC83] H. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A Filter Lambda Model and the Completeness of Type Assignment. *Journal of Symbolic Logic*, 48(4):931–940, 1983.

[BDD$^+$14] J. Bessai, A. Dudenhefner, B. Düdder, M. Martens, and J. Rehof. Combinatory Logic Synthesizer. In Tiziana Margaria and Bernhard Steffen, editors, *ISoLA'14*, volume 8802, pages 26–40, 2014.

[BDS13] H. Barendregt, W. Dekkers, and R. Statman. *Lambda Calculus with Types*. Perspectives in Logic, Cambridge University Press, 2013.

[BN01] F. Baader and P. Narendran. Unification of Concept Terms in Description Logics. *Journal of Symbolic Computation*, 31:277–305, 2001.

[BS01] F. Baader and W. Snyder. *Handbook of Automated Reasoning*, chapter Chapter 8, Unification Theory, pages 439–526. Elsevier, 2001.

[BZ05] G. Boudol and P. Zimmer. On type inference in the intersection type discipline. *Electr. Notes Theor. Comput. Sci.*, 136:23–42, 2005.

[CG95] M. Coppo and P. Giannini. Principal types and unification for a simple intersection type system. *Inf. Comput.*, 122(1):70–96, 1995.

[Chl86] B. S. Chlebus. Domino-tiling games. *Journal of Computer and System Sciences*, 32(3):374–392, 1986.

[CNXA15] G. Castagna, K. Nguyen, Z. Xu, and P. Abate. Polymorphic functions with set-theoretic types: Part 2: Local type inference and type reconstruction. In *Principles of Programming Languages POPL 2015*, pages 289–302. ACM, 2015.

[CP94] W. Charatonik and L. Pacholski. Set constraints with projections are in NEXPTIME. In *35th Annual Symposium on Foundations of Computer Science*, pages 642–653. IEEE, 1994.

[DCH92] M. Dezani-Ciancaglini and J. R. Hindley. Intersection Types for Combinatory Logic. *Theoretical Computer Science*, 100(2):303–324, 1992.

[DMR13] B. Düdder, M. Martens, and J. Rehof. Intersection Type Matching with Subtyping. In *Proceedings of TLCA'13*, Springer LNCS, 2013.

[DMR14] B. Düdder, M. Martens, and J. Rehof. Staged Composition Synthesis. In *Proceedings of ESOP'14*, volume 8410 of *LNCS*, pages 67–86. Springer, 2014.

[DMRU12] B. Düdder, M. Martens, J. Rehof, and P. Urzyczyn. Bounded Combinatory Logic. In *Proceedings of CSL'12*, volume 16 of *LIPIcs*, pages 243–258, 2012.

[DR17] A. Dudenhefner and J. Rehof. Intersection Type Calculi of Bounded Dimension. In *POPL 2017, Proceedings of the 44th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, Paris, France, January 2017*, 2017.

[FP91] T. Freeman and F. Pfenning. Refinement Types for ML. In *Proceedings of PLDI'91*, pages 268–277. ACM, 1991.

[Hin82] J. R. Hindley. The Simple Semantics for Coppo-Dezani-Sallé Types. In *International Symposium on Programming*, volume 137 of *LNCS*, pages 212–226. Springer, 1982.

[Hin08] J. R. Hindley. *Basic Simple Type Theory*. Cambridge Tracts in Theoretical Computer Science, vol. 42, 2008.

[HR98] F. Henglein and J. Rehof. Constraint automata and the complexity of recursive subtype entailment. In *Automata, Languages and Programming, 25th International Colloquium, ICALP'98, Aalborg, Denmark, July 13-17, 1998, Proceedings*, pages 616–627, 1998.

[HS08]   J. R. Hindley and J. P. Seldin. *Lambda-calculus and Combinators, an Introduction*. Cambridge University Press, 2008.

[Kfo99]   A. J. Kfoury. Beta-reduction as unification. *Banach Center Publications*, 46(1):137–158, 1999.

[KT95]   T. Kurata and M. Takahashi. Decidable properties of intersection type systems. In *TLCA*, volume 902 of *LNCS*, pages 297–311. Springer, 1995.

[KTU93]   A. J. Kfoury, Jerzy Tiuryn, and Pawel Urzyczyn. The undecidability of the semi-unification problem. *Inf. Comput.*, 102(1):83–101, 1993.

[KW04]   A. J. Kfoury and J. B. Wells. Principality and Type Inference for Intersection Types Using Expansion Variables. *Theor. Comput. Sci.*, 311(1-3):1–70, 2004.

[Reh98]   Jakob Rehof. *The Complexity of Simple Subtyping Systems*. PhD thesis, DIKU, Department of Computer Science, 1998.

[Reh06]   J. Rehof. Problem #16: Is the subtype entailment problem decidable? *R. Hasegawa, L. Paolini, and P. Urzyczyn (Eds.): TLCA (Typed Lambda Calculus and Applications) Open Problem List, http://tlca.di.unito.it/opltlca/*, 2006.

[Reh13]   J. Rehof. Towards Combinatory Logic Synthesis. In *BEAT'13, 1st International Workshop on Behavioural Types*. ACM, January 22 2013.

[Rey96]   J.C. Reynolds. Design of the Programming Language Forsythe. Technical Report CMU-CS-96-146, CMU, 1996.

[Ron88]   S. Ronchi Della Rocca. Principal Type Scheme and Unification for Intersection Type Discipline. *Theor. Comput. Sci.*, 59:181–209, 1988.

[RU11]   J. Rehof and P. Urzyczyn. Finite Combinatory Logic with Intersection Types. In *Proceedings of TLCA'11*, volume 6690 of *LNCS*, pages 169–183. Springer, 2011.

[RU12]   J. Rehof and P. Urzyczyn. The Complexity of Inhabitation with Explicit Intersection. In *Kozen Festschrift*, volume 7230 of *LNCS*, pages 256–270. Springer, 2012.

[Sta15]   R. Statman. A finite model property for intersection types. In *Proceedings Seventh Workshop on Intersection Types and Related Systems, ITRS 2014, Vienna, Austria, 18 July 2014.*, pages 1–9, 2015.

[TU96]   J. Tiuryn and P. Urzyczyn. The subtyping problem for second-order types is undecidable. In *lics*, page 74. IEEE, 1996.

[Urz09]   P. Urzyczyn. Inhabitation of Low-Rank Intersection Types. In *Proceedings of TLCA'09*, volume 5608 of *LNCS*, pages 356–370. Springer, 2009.

[vB11]   Steffen van Bakel. Strict Intersection Types for the Lambda Calculus. *ACM Computing Surveys*, 43(3), 2011.