Fundamenta Informaticae 170 (2019) 93–110
DOI 10.3233/FI-2019-1856
IOS Press

Undecidability of Intersection Type Inhabitation at Rank 3 and its Formalization

Andrej Dudenhefner*, **Jakob Rehof**
Department of Computer Science
TU Dortmund University
Dortmund, Germany
{andrej.dudenhefner, jakob.rehof}@cs.tu-dortmund.de

Abstract. We revisit the undecidability result of rank 3 intersection type inhabitation (Urzyczyn 2009) in pursuit of two goals. First, we simplify the existing proof, reducing simple semi-Thue systems to intersection type inhabitation in the original Coppo-Dezani type assignment system. Additionally, we outline a direct reduction from the Turing machine halting problem to intersection type inhabitation. Second, we formalize soundness and completeness of the reduction in the Coq proof assistant under the banner of "type theory inside type theory".

1. Introduction

Intersection types [1, 2] capture deep semantic properties of λ-terms such as normalization and have been subject to extensive study for decades [3]. A decision problem of fundamental interest for a type system is the *type inhabitation problem* (or, *type emptiness problem*): given a type, does there exist a term having the type? In this paper we are concerned with the inhabitation problem for intersection types. More specifically, we are interested in the borderline between decidability and undecidability and the fine structure of this problem. In pursuit of this goal, in this paper we develop a simplified undecidability proof by reduction from semi-Thue systems, and we formalize soundness and completeness of the reduction in the Coq proof assistant. Our understanding of this topic has to a large extent been influenced by the work of Paweł Urzyczyn, as we will proceed to outline in the remainder of this introduction.

*Address for correspondence: Department of Computer Science, TU Dortmund University, Dortmund, Otto-Hahn-Str. 14, 44227 Dortmund , Germany

1.1. Decidability, complexity, and rank

Whereas undecidability of the dual problem of *typability* (given a term, does it have a type?) follows immediately from the normalization properties of the intersection type system, the question of decidability of the inhabitation problem for intersection types remained open for many years, until Urzyczyn proved in 1994 that the problem is undecidable [4, 5]. The proof is by reduction from the halting problem for queue automata and was carried out for the general intersection type system of [6]. Recently, in [7, 8], the inhabitation problem has also been shown to be equivalent to the problem of λ-definability [9], for which Loader proved undecidability in 1993 (but first published in 2001) [10], thereby further improving our understanding of the expressiveness of intersection types (an overview of the situation can be found in [3, 8]).

Later, in 2009, the fine structure of the problem was clarified in terms of the *rank hierarchy* (referring to Leivant's notion of rank [11]). By a fascinating, technically deep analysis of the problem Urzyczyn [12] determined the exact position of the problem in terms of decidability and complexity under bounded rank. In 2007 it had been shown in a paper by his student Kuśmierek [13] that the problem is decidable and EXPTIME-hard in rank 2. Since the reductions showing undecidability in [5] were carried out within rank 4, the analysis of the inhabitation problem in [12] essentally pivots around the status of the problem in rank 2 and and rank 3 as determined by the questions: what is the exact complexity of the problem in rank 2, and is the problem decidable in rank 3? Urzyczyn closed both gaps by showing in [12] that the problem is EXPSPACE-complete in rank 2 and undecidable in all ranks k for $k \geq 3$.

Exponential space hardness for the rank 2-bounded problem is proven by simulation of alternating [14] exponential time Turing machines. The simulation uses a special intermediary device suited for representing the rank 2 inhabitation problem called *bus machines*. Such machines are alternating devices operating on a fixed length word with the property that instructions may be dynamically extended during computation. Incidentally, bus machines were useful again later, in [15] for showing EXPSPACE-completeness of inhabitation in the fragment of the intersection type system of [6] without intersection introduction studied in [16], and in [17, 18] for showing EXPSPACE-completeness of provability in the class Σ_1 of the Mints hierarchy of intuitionistic first-order logic.

The exponential space upper bound follows by a modified and extended version of a Wajsberg/Ben Yelles-style algorithm for simple types [19, 20]. First, the upper bound argument in [12] is based on an alternating decision procedure. The idea of introducing alternation into inhabitation algorithms was first realized by Urzyczyn in 1997 [21], leading (among other results) to a major simplification of the proof of PSPACE-completeness of inhabitation in simple types (Statman's theorem [22]). Second, the algorithm operates on systems of *simultaneous* ("parallel") inhabitation constraints representing obligations to find a single inhabitant satisfying all the constraints, which is characteristic of the intersection type system. The rank 2 restriction can then be used to induce a linear bound (in the size of the input type) on the number of simultaneous constraints that can arise, which in turn leads to an exponential upper bound on non-repeating sequences of configurations during runs of the algorithm. One can view the upper bound argument as providing a linear bound on the degree of parallelism of the alternating search trees defined by runs of the algorithm, thereby bounding search in alternating exponential time and hence in EXPSPACE.

One can take several routes in order to show undecidability of intersection type inhabitation (abbreviated by IHP, resp. IHP3 for rank 3). In [3] the following reduction is performed: EQA $\leq$ ETW $\leq$ WTG $\leq$ IHP, where EQA is the emptiness problem for queue automata, ETW is the emptiness problem for typewriter automata and WTG is the problem of determining whether one can win a tree game. A different route taken in [12] performs the following reduction: ELBA $\leq$ SSTS1 $\leq$ HETM $\leq$ IHP3, where ELBA is the emptiness problem for linear bounded automata, SSTS1 is the problem of deciding whether there is a word that can be rewritten to 1s in a simple semi-Thue system and HETM is the halting problem for expanding tape machines. Alternatively, following the route of [8] via semantics, the following reduction can be performed: WSTS $\leq$ LDF $\leq$ IHP3, where WSTS is the word problem in semi-Thue systems and LDF is the λ-definability problem. Each of these routes introduces its own machinery that may distract from the initial question. As a result, it is challenging to even give examples of particularly hard inhabitation problem instances, or distinguish necessary properties on a finer scale than the rank restriction.

In this work, we reduce a word rewriting problem for simple semi-Thue systems (Section 3) to intersection type inhabitation in the Coppo-Dezani type assignment system (Section 2). Soundness and completeness of the reduction are shown in Section 4. Since simple semi-Thue systems are closely related to Turing machines, the halting problem can be simulated directly by proof search (Section 5). Key definitions and results are accompanied by references to their corresponding formalization[1] in the Coq proof assistant. In particular, the soundness and completeness results are fully formalized. While this work is self-contained, the proof techniques used are based on previous unpublished work by the authors (see [23] for a technical report) improving upon it in the following aspects. First, the proofs are done with respect to the original Coppo-Dezani type assignment system, while previous work [12, 23] uses an intermediate presentation. Second, the reduction focuses on simplicity (for a more accessible formalization) using simple semi-Thue systems as a starting point instead of Turing machines.

1.2. Synthesis and dimensional calculi

Our interest in understanding the borderline between decidability and undecidability at rank 3 of the inhabitation problem for λ-calculus with intersection types was partly motivated from working on type-theoretic foundations for synthesis and, in particular, is most closely connected with our recent development of a theory of dimensional intersection type calculi [24, 25, 26]. In [24] we were able to generalize decidability of rank 2 inhabitation for intersection types substantially, based on the notion of dimension. In the course of developing these results we were naturally led to a close study of [12], and we made intensive use both of methods and of results of that paper. In the remainder of this introduction we will briefly sketch this context in some more detail.

Around 2010 the second author became interested in the problem of inhabitation for intersection types, motivated from the idea of using intersection types as a foundation for type-based synthesis, initially using combinatory logic as a foundation for component-oriented synthesis as outlined in a statement of the programme in [27] and later in [28]. From this standpoint one is naturally interested in computational aspects and decidable fragments of the problem. It was therefore natural to turn

[1] `https://github.com/mrhaandi/lambda-cap`

to Paweł Urzyczyn to discuss the status of the problem[2], and a number of joint papers followed on inhabitation in bounded combinatory logic with intersection types [29, 30] and on the aforementioned restricted problem in λ-calculus [15].

In combinatory logic the bounding principle was based on "levels" (depth of instantiation, which can be understood in terms of order and rank). A technical highlight was the exponential time hierarchy for relativized inhabitation in bounded combinatory logic with intersection types [30], generalizing the EXPTIME-completeness result of [29], the lower bounds being obtained by simulating exponential space alternating Turing machines involving an encoding of numerals in intersection types. Relativized inhabitation in combinatory logic is the decision problem: given a combinatory theory $\mathscr{C}$ and a type τ, is there a combinatory expression in the theory $\mathscr{C}$ with type τ? Since the problem is relativized to an arbitrary given combinatory theory (a Hilbert-style propositional theory), it is undecidable (when considered without bounding restrictions) already in simple types (cf. Linial-Post theorems [31], see also [32] for recent results). In applications to synthesis, the input combinatory theory is used to model a repository of components relative to which synthesis is performed (see, e.g., [33, 34, 35, 36]).

For λ-calculus with intersection types we recently developed a theory of *dimension* [24, 25, 26], which can be used as a bounding principle orthogonal to the notion of rank. It was shown in [24] that inhabitation under dimensional bound strictly generalizes decidability of inhabitation for the rank 2-fragment [12]. But, whereas rank-bounded inhabitation has a strict borderline at rank 2 (becoming undecidable above rank 2), inhabitation under dimensional bound remains decidable for every fixed bound[3] and approximates inhabitation in the unbounded system arbitrarily (hence also across all ranks) under increasing bound, since every inhabitant can be found in sufficiently large dimension. The basic idea is to introduce a *norm* on derivations by instrumenting the intersection type system with *elaborations*, $\mathbf{P}$, which are proof theoretic decorations of pure λ-terms M in which the usage of the intersection introduction rule ($\cap$I) is made explicit by recording the sets (called decoration sets) of types which are assigned to subterm occurrences of the term. The norm on elaborations, $\|\mathbf{P}\|$, measures the maximal size of decorating sets in $\mathbf{P}$, thereby yielding a measure of the amount of intersection introduction performed to type the term. A central component of the measurement system based on the norm on elaborations is to interpret intersection introduction by a form of addition on elaborations such that, whenever $\mathbf{P}$ and $\mathbf{Q}$ are elaborations of the same term at two types, then $\mathbf{P} \sqcup \mathbf{Q}$ is an elaboration

[2]The second author may be forgiven for relating briefly, on a personal note, how this came about. I met Paweł Urzyczyn the first time in Warsaw at LICS 1997. Since then we had occasion to interact several times, not least during a longer stay of Paweł in Copenhagen, in the group working with Fritz Henglein. During this time I once had occasion to help him out, when he was in a tight spot having locked himself out of his apartment in Copenhagen. In addition to our common interest in type theory, this incident laid the foundation for a lasting friendship. Later, from 1998 until 2010, we had little contact. Then, in 2010, for reasons mentioned above, I contacted Paweł for the first time in about 10 years. I used the tragic plane crash of the Polish president as a context in which to contact him again so suddenly after so many years. Here is the initial mail exchange: "JR to PU on 10.04.2010. Subject: Greetings. Dear Pawel, Sorry to hear about the disaster in Russia. [One sentence left out]. I hope you are doing well. I have been looking at inhabitation quite a bit lately, was very impressed with your results for intersection types. All the best, Jakob." His answer: "PU to JR on 10.04.2010. Hi, Jakob! Nice to hear from you. Why are you interested in the inhabitation? Best, Pawel." I think one can describe this exchange as being to the point. As a Scandinavian I appreciate the laconic quality, and the line was immediatley hot. Several joint papers ensued in short order, including [29, 30, 15].

[3]This holds under a *multiset* interpretation of dimension, as explained below.

of the term at the intersection of the types. Here, the operation $\sqcup$ recursively unions decoration sets at corresponding positions in $\mathbf{P}$ and $\mathbf{Q}$ and satisfies the triangle inequality $\|\mathbf{P} \sqcup \mathbf{Q}\| \leq \|\mathbf{P}\| + \|\mathbf{Q}\|$. The *set-theoretic dimension* of M (at a type) can now be defined as the smallest d such that the type is derivable for M by some elaboration $\mathbf{P}$ with $\|\mathbf{P}\| = d$. We define the *multiset dimension* of a typing analogously, by considering multisets of decoration types and by defining the norm accordingly and addition on elaborations by multiset union. It turns out that β-reduction is not norm-increasing (in either measurement system), and therefore each dimensional fragment constitutes a meaningful type system enjoying subject reduction under dimensional bound, both in the set-theoretic system and in the multiset system. A central result of [24] is that, whereas inhabitation is undecidable under bounded set-theoretic dimension (this is [24, Theorem 28]), inhabitation is decidable and EXPSPACE-complete under bounded multiset dimension (this is [24, Theorem 34]). Moreover, it turns out that the rank 2-fragment is contained in linear multiset dimension (this is [24, Proposition 23]), from which it follows that decidability in EXPSPACE of rank 2-inhabitation is strictly subsumed by the aforementioned result for bounded multiset dimension.

While providing a substantial generalization of the exponential space decidability result for rank 2-inhabitation of [12], the development of the results in [24] would not have been possible without [12]. A key insight behind those results is that the notion of multiset dimension corresponds to the maximal width of systems of parallel constraints in the sense of [12] which are needed in an alternating search procedure for inhabitants with norm bounded by the dimensional parameter. In addition, the EXPSPACE-lower bound is obtained by reduction from the rank 2-inhabitation problem via the results of [12]. In this sense it is safe to say that [12] was a *sine qua non* for [24]. The results presented in the remainder of this paper were obtained in the course of our continued work on understanding the fine structure of the inhabitation problem at rank 3.

2. Coppo-Dezani intersection type assignment system

The Coppo-Dezani type assignment system [37, Sec. 2] was introduced as an extension to the simply typed system allowing to have sets of types as premises. This initiated the line of work known today as the intersection type discipline (for a historic survey and further developments see [38]).

In this section we assemble the necessary prerequisites in order to discuss inhabitation in the Coppo-Dezani type system in the presentation of [38, Sec. 3.2]. We denote λ-*terms* (Definition 2.1) by M, N, where *term variables* are denoted by x, y.

Definition 2.1. (λ-Terms)

$$M, N ::= x \mid (\lambda x.M) \mid (M\ N)$$

We denote *intersection types* (Definition 2.2) by σ, τ, and *strict types* (subset of intersection types where the outermost constructor is not an intersection) by A, B, where *type atoms* (also called *type variables* in literature) are denoted by a, b, c and drawn from the denumerable set $\mathbb{A}$. Conventionally, the intersection operator $\cap$ binds more strongly than $\rightarrow$.

Definition 2.2. (Intersection Types)

$$A, B ::= a \mid \sigma \to A$$
$$\sigma, \tau ::= A_1 \cap \cdots \cap A_n \text{ where } n \geq 1$$

Historically, strict types (a term coined in [39]) are called "types" and intersection types are treated as sets of types in [37]. Although we adopt the modern presentation by van Bakel [38], the above syntax coincides with the original one.

A *type environment*, denoted by Γ, is a finite set of *type assumptions* having the shape $x : \sigma$ for distinct term variables. We define the *domain* dom, the *codomain* ran and the *extension* of Γ.

Definition 2.3. (Type Environment)

$$\Gamma ::= \{x_1 : \sigma_1, \ldots, x_n : \sigma_n\} \text{ where } x_i \neq x_j \text{ for } i \neq j$$
$$\mathrm{dom}(\Gamma) = \{x_1, \ldots, x_n\}$$
$$\mathrm{ran}(\Gamma) = \{\sigma_1, \ldots, \sigma_n\}$$
$$\Gamma, x : \sigma = \Gamma \cup \{x : \sigma\} \text{ where } x \notin \mathrm{dom}(\Gamma)$$

The rules of the Coppo-Dezani type assignment system in the presentation of [38, Sec. 3.2] with *judgements* of shape $\Gamma \vdash M : \sigma$ are given below. This system types any term in β-normal form, and enjoys subject reduction and strong normalization properties.

Definition 2.4. (Coppo-Dezani Type Assignment System)

$$\frac{1 \leq i \leq n}{\Gamma, x : A_1 \cap \cdots \cap A_n \vdash x : A_i}\ (\cap\mathrm{E}) \qquad \frac{\Gamma \vdash M : \sigma \to A \qquad \Gamma \vdash N : \sigma}{\Gamma \vdash M\ N : A}\ (\to\mathrm{E})$$

$$\frac{\Gamma, x : \sigma \vdash M : A}{\Gamma \vdash \lambda x.M : \sigma \to A}\ (\to\mathrm{I}) \qquad \frac{\Gamma \vdash M : A_1 \qquad \cdots \qquad \Gamma \vdash M : A_n}{\Gamma \vdash M : A_1 \cap \cdots \cap A_n}\ (\cap\mathrm{I})$$

Observe that in [12] a slightly different typing system is considered, which is more along the lines of the Barendregt-Coppo-Dezani type assignment system [6], albeit without subtyping. Although we could have chosen any conventional flavor of intersection type assignment (the universal type ω can also be included freely[4]), our choice of the Coppo-Dezani type assignment system is made in honor of the original authors.

The above type assignment is formalized in `CD_Derivation.v` as the inductive type

```
Inductive cd_derivation : environment → term → formula' → Prop
```

having as type constructors exactly the above typing rules including well-formedness conditions on the type environment (having distinct term variables) and terms (having no unbound De Bruijn indices). The formalization uses lists as type environments instead of sets, which is immaterial because the

[4] A corresponding formalization is found at `https://github.com/mrhaandi/lambda-cap/tree/cdv`

typing system enjoys weakening (and permutation) of type environments as shown in

```
Theorem cd_weakening : ∀ (Γ Γ': environment) (M: term) (t: formula),
  cd_derivation Γ M t → well_formed_environment Γ' →
    (∀ (p : label * formula'), In p Γ → In p Γ') → cd_derivation Γ' M t.
```

After Leivant [11] we define the *rank* of a type.

Definition 2.5. (Rank)

$$\begin{aligned} \mathrm{rank}(\tau) &= 0 \text{ if } \tau \text{ is a simple type, i.e. containing no } \cap \\ \mathrm{rank}(\sigma \to A) &= \max\{1 + \mathrm{rank}(\sigma), \mathrm{rank}(A)\} \\ \mathrm{rank}(A_1 \cap \cdots \cap A_n) &= \max\{1, \mathrm{rank}(A_1), \ldots, \mathrm{rank}(A_n)\} \end{aligned}$$

Example 2.5. Let $\tau = \bigcap_{i=1}^{n}(((d_i \to c_i) \to b_i) \to a_i) \to a$. We have $\mathrm{rank}(\tau) = 2$, although the functional order (i.e. nesting of $\to$ to the left) is higher. Note that types that are similar to τ are used in [12] to encode expanding tape machine switches.

One core decision problem for any typing system is *inhabitation* (Problem 1), which is the dual of typability.

Problem 1. (Inhabitation $\Gamma \vdash ? : \tau$)
Given a type environment Γ and a type τ, is there a λ-term M such that $\Gamma \vdash M : \tau$ is derivable?

Clearly, the above problem is equivalent to inhabitation in the empty type environment by abstracting all assumptions. While inhabitation in the simply typed λ-calculus is PSPACE-complete [22], it is undecidable in traditional intersection typed systems [12].

3. Simple semi-Thue systems

In this section we recapitulate the notion of *simple semi-Thue systems* which are closely related to Turing machines, albeit much simpler to handle. Additionally, we give details on the undecidability of rewriting zeroes into ones in such a system. A similar problem is used in [12] as a starting point to establish undecidability of intersection type inhabitation.

Definition 3.1. (Semi-Thue System)
A *semi-Thue system* $\mathcal{R}$ over an alphabet Σ is a finite set of rules of shape $v \to w$ where $v, w \in \Sigma^*$. The relation $\to$ is extended to $\to_{\mathcal{R}} \subseteq \Sigma^* \times \Sigma^*$ by $tvu \to_{\mathcal{R}} twu$, if $v \to w$ where $t, u \in \Sigma^*$. The relation $\twoheadrightarrow_{\mathcal{R}}$ is the reflexive, transitive closure of $\to_{\mathcal{R}}$.

Definition 3.2. (Simple Semi-Thue System)
A semi-Thue system $\mathcal{R}$ over an alphabet Σ, where each rule has the form $ab \to_{\mathcal{R}} cd$ for some $a, b, c, d \in \Sigma$, is called a *simple semi-Thue system*.

The simple semi-Thue System rewriting relation is formalized in `SSTS.v` as the inductive type

```
Inductive rewrites_to (rs : list rule) : list nat → list nat → Prop
```

where words are represented by lists of natural numbers. Let us abbreviate $a \cdots a \in \Sigma^n$ by a^n.

Problem 2. ($0^? \twoheadrightarrow_{\mathcal{R}} 1^?$)
Given a simple semi-Thue system $\mathcal{R}$ over an alphabet Σ such that $0, 1 \in \Sigma$, is there an $n \in \mathbb{N}$ with $n > 0$ such that $0^n \twoheadrightarrow_{\mathcal{R}} 1^n$?

Accordingly, the above decision problem is formalized in `SSTS.v` as

```
∃ (m : nat), rewrites_to rs (repeat 0 (1+m)) (repeat 1 (1+m))
```

Lemma 3.3. Problem 2 is undecidable.

Proof:
Reduction from the empty word halting problem for a given Turing machine $\mathcal{M}$.

A configuration (q, i, v) of a $\mathcal{M}$, where

- $q \in Q$ is the current state
- $i \in \mathbb{N}$ is the current head position
- $v \in \Theta^*$ is the current tape content over the alphabet Θ

can be encoded as a word $w \in ((Q \times \Theta) \cup \Theta)^*$ identical to v except having at i-th position the symbol (q, v_i). A transition $\delta(q, a) = (q', a', \text{left})$ (resp. (q', a', right)) can be encoded as the set of rules $\{c(q, a) \to (q', c)a' \mid c \in \Theta^*\}$ (resp. $\{(q, a)c \to a'(q', c) \mid c \in \Theta^*\}$).

Without loss of generality $0, 1 \notin \Theta$ and $\mathcal{M}$ halts only if the head is at the leftmost (starting) position and the contexts of the tape is empty. Let $q_0, q_f \in Q$ be the initial and final states, and $\text{␣} \in \Theta$ the (default) empty tape symbol.

Define the simple semi-Thue system $\mathcal{R}$ over the alphabet $\Sigma = \{0, 1\} \cup (Q \times \Theta) \cup \Theta$ having as rules the above encoding of the transition function together with $00 \to_{\mathcal{R}} 1(q_0\text{␣})$, $00 \to_{\mathcal{R}} \text{␣␣}$, $(q_f, \text{␣})\text{␣} \to_{\mathcal{R}} 1(q_f, \text{␣})$, and $(q_f, \text{␣})\text{␣} \to_{\mathcal{R}} 11$.

If $\mathcal{M}$ accepts the empty word using at most $n > 0$ tape cells, then we have

$$\Sigma^{n+1} \ni 0 \cdots 0 \twoheadrightarrow_{\mathcal{R}} 1(q_0, \text{␣})\text{␣} \cdots \text{␣} \twoheadrightarrow_{\mathcal{R}} 1(q_f, \text{␣})\text{␣} \cdots \text{␣} \twoheadrightarrow_{\mathcal{R}} 1 \cdots 1 \in \Sigma^{n+1}$$

Conversely, if for some $n > 0$ we have $0^n \twoheadrightarrow_{\mathcal{R}} 1^n$, then

$$0^n \twoheadrightarrow_{\mathcal{R}} 1(q_0, \text{␣})\text{␣} \cdots \text{␣}w \twoheadrightarrow_{\mathcal{R}} 1(q_f, \text{␣})\text{␣} \cdots \text{␣}w \twoheadrightarrow_{\mathcal{R}} 1^n \text{ such that } w \in \Sigma^0 \text{ or } w = 1w'$$

Since $0, 1 \notin \Theta$ and the rules not including $0, 1$ exactly correspond to the valid transitions in $\mathcal{M}$, we can reconstruct an accepting run of $\mathcal{M}$. □

4. Reduction from rewriting to inhabitation

In this section we reduce simple semi-Thue system rewriting $0^? \twoheadrightarrow_{\mathcal{R}} 1^?$ (Problem 2) to intersection type inhabitation $\Gamma \vdash ? : \tau$ (Problem 1). The used tools immediately (although, with more bookkeeping) give a direct reduction from the halting problem to intersection type inhabitation.

For the remainder of this section, let us fix a simple semi-Thue system $\mathcal{R}$ over the alphabet $\Sigma \subseteq \mathbb{A}$ such that $0, 1 \in \Sigma$ and $\blacktriangle, \bullet, *, \#, \$, l, r \in \mathbb{A} \setminus \Sigma$.

We define the following types of rank at most 2 and the type environment $\Gamma_{\blacktriangle}$, providing some intuition for their intended use.

$$\sigma_{\blacktriangle} = (\# \cap \$) \to \blacktriangle$$

initializes the first word to $\#\$$

$$\sigma_{*} = \Big((\bullet \to *) \to *\Big) \cap \Big((l \to *) \to \#\Big) \cap \Big(\big((r \to \#) \cap (\bullet \to \$)\big) \to \$\Big)$$

expands the word $* \cdots * \#\$$ to $* \cdots * *\#\$$

$$\sigma_{0} = (0 \to *) \cap (0 \to \#) \cap (1 \to \$)$$

ends the expansion phase by rewriting $* \cdots * \#\$$ to $0 \cdots 001$

$$\sigma_{1} = 1$$

accepts the word $1 \cdots 1$

$$\sigma_{ab \to cd} = (l \to c \to a) \cap (r \to d \to b) \cap \bigcap_{e \in \Sigma} (\bullet \to e \to e) \text{ for each } ab \to cd \in \mathcal{R}$$

rewrites the word $vabw$ to $vcdw$

$$\Gamma_{\blacktriangle} = \{x_{\blacktriangle} : \sigma_{\blacktriangle}, x_{*} : \sigma_{*}, x_{0} : \sigma_{0}, x_{1} : \sigma_{1}\} \cup \{x_{ab \to cd} : \sigma_{ab \to cd} \mid ab \to cd \in \mathcal{R}\}$$

The above definitions are formalized in `Encoding.v`. In particular, $\Gamma_{\blacktriangle}$ corresponds to the environment `Γ_init ++ Γ_step rs`.

We will show that $\Gamma_{\blacktriangle} \vdash ? : \blacktriangle$ is equivalent to $0^{?} \twoheadrightarrow_{\mathcal{R}} 1^{?}$, which is formalized [40] in `MainResult.v` as

```
Theorem correctness : ∀ (rs : list rule),
  (∃ (m : nat), rewrites_to rs (repeat 0 (1+m)) (repeat 1 (1+m))) ↔
  (∃ (N : term), normal_form N ∧
    cd_derivation (Γ_init ++ Γ_step rs) N (singleton (atom triangle))).
```

As observed by Urzyczyn in [12], inhabitation amounts to solving a set of "parallel" judgement constraints $\Gamma_1 \vdash M : a_1, \ldots, \Gamma_n \vdash M : a_n$ with $\mathrm{dom}(\Gamma_i) = \mathrm{dom}(\Gamma_j)$ for $1 \leq i, j \leq n$. If we are able to establish a linear order on such parallel constraints, we can represent a word $v = a_1 \cdots a_n$ by derived type atoms $a_1, \ldots, a_n$ and transition rules by assumptions in $\Gamma_1, \ldots, \Gamma_n$ (leading to the idea of *bus machines* in [12]). In presence of intersection introduction, we are able to extend the current word (leading to the idea of *expanding tape machines* in [12]).

The Coppo-Dezani type assignment system turns out to be somewhat unwieldy for the purpose of formalization, therefore the core soundness and completeness proofs are realized with respect to the auxiliary inductive type

```
Inductive derivation : nat → environment → term → formula → Prop
```

defined in `Derivation.v` focusing on strict types (similarly to [39]), relaxing well-formedness conditions, and providing in the first parameter an upper bound on the depth of the derivation. It is important to note that the `correctness` theorem is proven with respect to `cd_derivation`, and `derivation` only serves an intermediary role. Therefore, the formalization of the main `correctness` result covers exactly the Coppo-Dezani type assignment system.

4.1. Soundness

In this paragraph, we show that $\Gamma_{\blacktriangle} \vdash ? : \blacktriangle$ implies $0^? \twoheadrightarrow_{\mathcal{R}} 1^?$.

As previously discussed, we need to establish a linear order on parallel judgements to be able to encode a word. For this purpose (and differently from [12]), we define the following types of rank at most 2 and the family of type environments Γ_i^n

$$\sigma_i^j = \begin{cases} l & \text{if } i = j \\ r & \text{if } i = j+1 \\ \bullet & \text{otherwise} \end{cases} \qquad \Gamma_i^n = \{y_j : \sigma_i^j \mid 1 \leq j < n\} \text{ for } 1 \leq i$$

By construction, the following Lemmas 4.1, 4.2 enable us to identify "neighboring" judgements.

Lemma 4.1. For $n \geq 1$ we have
$\Gamma_{n+1}^n \vdash y_j : \bullet$ iff $\Gamma_j^n \vdash y_j : l$ and $\Gamma_{j+1}^n \vdash y_j : r$ and $\Gamma_i^n \vdash y_j : \bullet$ for $i < j$ or $i > j+1$.

Lemma 4.2. For $n \geq 1$ we have

$$\Gamma_i^{n+1} = \Gamma_i^n, y_n : \bullet \text{ for } 1 \leq i < n \qquad \Gamma_{n+2}^{n+1} = \Gamma_{n+1}^n, y_n : \bullet$$
$$\Gamma_n^{n+1} = \Gamma_n^n, y_n : l \qquad \Gamma_{n+1}^{n+1} = \Gamma_{n+1}^n, y_n : r = \Gamma_{n+2}^n, y_n : r$$

The following Lemma 4.3 shows that, once particular parallel constraints are established for a word v, we obtain $v \twoheadrightarrow_{\mathcal{R}} 1 \cdots 1$.

Lemma 4.3. Let $v = a_1 \cdots a_n \in \Sigma^n$ where $n > 0$, and let $\Gamma_i = \Gamma_{\blacktriangle} \cup \Gamma_i^n$ for $1 \leq i$. If there exists a term N such that $\Gamma_i \vdash N : a_i$ for $1 \leq i \leq n$ and $\Gamma_{n+1} \vdash N : 1$, then $v \twoheadrightarrow_{\mathcal{R}} 1^n$.

Proof:
Without loss of generality N is in β-normal form. We proceed by induction on the size of N. Consider $\Gamma_{n+1} \vdash N : 1$. We have two possible cases:

Case $N = x_1$**:** We have $a_1 = \cdots = a_n = 1$, therefore $v = 1^n$.

Case $N = x_{ab \to cd}\ y_j\ M$ **for some** $1 \leq j < n$**:** We have $\Gamma_{n+1} \vdash y_j : \bullet$ and $\Gamma_{n+1} \vdash M : 1$. By inversion we have $(y_j : \bullet) \in \Gamma_{n+1}^n$, therefore $\Gamma_{n+1}^n \vdash y_j : \bullet$. By Lemma 4.1 we obtain

$$\begin{aligned} \Gamma_j \vdash y_j : l &\implies a_j = a \text{ and } \Gamma_j \vdash M : c \\ \Gamma_{j+1} \vdash y_j : r &\implies a_{j+1} = b \text{ and } \Gamma_{j+1} \vdash M : d \\ \Gamma_i \vdash y_j : \bullet &\implies \Gamma_i \vdash M : a_i \text{ for } i < j \text{ or } i > j+1 \end{aligned}$$

In sum, $v = tabu$ for some $t, u \in \Sigma^*$, and by the induction hypothesis $v \twoheadrightarrow_{\mathcal{R}} tcdu \twoheadrightarrow_{\mathcal{R}} 1^n$. □

The above argumentation is formalized in `Soundness.v` as `Lemma soundness_step`.

It remains to show that the above parallel constraints are necessarily met. The following Lemma 4.4 shows that parallel constraints required in the above Lemma 4.3 are necessarily established.

Lemma 4.4. If there exists $n > 0$ and a term N such that $\Gamma_{\blacktriangle} \cup \Gamma_i^n \vdash N : *$ for $1 \leq i < n$, $\Gamma_{\blacktriangle} \cup \Gamma_n^n \vdash N : \#$, and $\Gamma_{\blacktriangle} \cup \Gamma_{n+1}^n \vdash N : \$$, then there exists an $n' > 0$ and N' such that $\Gamma_{\blacktriangle} \cup \Gamma_i^{n'} \vdash N' : 0$ for $i = 1 \ldots n'$, and $\Gamma_{\blacktriangle} \cup \Gamma_{n'+1}^{n'} \vdash N' : 1$.

Proof:
Without loss of generality N is in β-normal form. We proceed by induction on the size of N. Consider $\Gamma_{\blacktriangle} \cup \Gamma_{n+1}^n \vdash N : \$$. We have two possible cases:

Case $N = x_0\ M$**:** For $n' = n$ we obtain $\Gamma_{\blacktriangle} \cup \Gamma_i^{n'} \vdash M : 0$ for $i = 1 \ldots n'$, and $\Gamma_{\blacktriangle} \cup \Gamma_{n'+1}^{n'} \vdash M : 1$.

Case $N = x_*\ (\lambda y_n.M)$**:** We have by Lemma 4.2

$$\begin{aligned}
\Gamma_{\blacktriangle} \cup \Gamma_i^n \vdash \lambda y_n.M : \bullet \to * &\implies \Gamma_{\blacktriangle} \cup \Gamma_i^{n+1} \vdash M : * \text{ for } 1 \leq i < n \\
\Gamma_{\blacktriangle} \cup \Gamma_n^n \vdash \lambda y_n.M : l \to * &\implies \Gamma_{\blacktriangle} \cup \Gamma_n^{n+1} \vdash M : * \\
\Gamma_{\blacktriangle} \cup \Gamma_{n+1}^n \vdash \lambda y_n.M : (r \to \#) \cap (\bullet \to \$) &\implies \Gamma_{\blacktriangle} \cup \Gamma_{n+1}^{n+1} \vdash M : \# \\
&\quad \text{and } \Gamma_{\blacktriangle} \cup \Gamma_{n+2}^{n+1} \vdash M : \$
\end{aligned}$$

In sum, the claim immediately follows by the induction hypothesis. □

The above argumentation is formalized in `Soundness.v` as `Lemma soundness_expand`.

Finally, we can show soundness of the described reduction in the following Theorem 4.5.

Theorem 4.5. If $\Gamma_{\blacktriangle} \vdash\ N : \blacktriangle$, then there exists an $n > 0$ such that $0^n \twoheadrightarrow_{\mathcal{R}} 1^n$.

Proof:
Without loss of generality N is in β-normal form. Since $\blacktriangle$ appears only in $\sigma_{\blacktriangle}$ in the type environment, we have $N = x_{\blacktriangle}\ M$ such that $\Gamma_{\blacktriangle} \vdash\ M : \#$ and $\Gamma_{\blacktriangle} \vdash\ M : \$$. Since $\Gamma_1^1 = \Gamma_2^1 = \emptyset$, we have $\Gamma_{\blacktriangle} \cup \Gamma_1^1 \vdash\ M : \#$ and $\Gamma_{\blacktriangle} \cup \Gamma_2^1 \vdash\ M : \$$. Therefore, by Lemma 4.4, there exists an $n' > 0$ and N' such that $\Gamma_{\blacktriangle} \cup \Gamma_i^{n'} \vdash N' : 0$ for $i = 1 \ldots n'$, and $\Gamma_{\blacktriangle} \cup \Gamma_{n'+1}^{n'} \vdash N' : 1$. Therefore, by Lemma 4.3, we obtain $0^{n'} \twoheadrightarrow_{\mathcal{R}} 1^{n'}$. □

The above argumentation is formalized in `Soundness.v` as `Lemma soundness`.

4.2. Completeness

In this paragraph, we show that $0^? \twoheadrightarrow_{\mathcal{R}} 1^?$ implies $\Gamma_{\blacktriangle} \vdash ? : \blacktriangle$.

Similarly to the proof of soundness, we first show in the following Lemma 4.6 that, once particular parallel constraints are established, we can represent simple semi-Thue rewriting.

Lemma 4.6. Let $v = a_1 \cdots a_n \in \Sigma^n$ where $n > 0$, and let $\Gamma_i = \Gamma_{\blacktriangle} \cup \Gamma_i^n$ for $1 \leq i$. If $v \twoheadrightarrow_{\mathcal{R}} 1^n$, then there exists a term N such that $\Gamma_i \vdash N : a_i$ for $1 \leq i \leq n$ and $\Gamma_{n+1} \vdash N : 1$.

Proof:
Induction on the number of rewriting steps.

Case $v = 1^n$**:** The term $N = x_1$ immediately shows the claim.

Case $v = tabu \to_{\mathcal{R}} tcdu \twoheadrightarrow_{\mathcal{R}} 1^n$ where $t \in \Sigma^{n'}$ and $u \in \Sigma^*$ for some $n' \geq 0$**:** Let $a_1 \cdots a_n = tabu$ and $j = n' + 1$. By the induction hypothesis there exists a term M such that

$$\Gamma_i \vdash M : a_i \text{ for } 1 \leq i < j \implies \Gamma_i \vdash x_{ab\to cd}\ y_j\ M : a_i \text{ for } 1 \leq i < j$$
$$\text{where } \Gamma_i \vdash x_{ab\to cd} : \bullet \to a_i \to a_i \text{ and } \Gamma_i \vdash y_j : \bullet \text{ for } 1 \leq i < j$$
$$\Gamma_j \vdash M : c \implies \Gamma_j \vdash x_{ab\to cd}\ y_j\ M : a$$
$$\text{where } \Gamma_j \vdash x_{ab\to cd} : l \to c \to a \text{ and } \Gamma_j \vdash y_j : l$$
$$\Gamma_{j+1} \vdash M : d \implies \Gamma_{j+1} \vdash x_{ab\to cd}\ y_j\ M : b$$
$$\text{where } \Gamma_{j+1} \vdash x_{ab\to cd} : r \to d \to b \text{ and } \Gamma_{j+1} \vdash y_j : r$$
$$\Gamma_i \vdash M : a_i \text{ for } 1 + j < i \leq n \implies \Gamma_i \vdash x_{ab\to cd}\ y_j\ M : a_i \text{ for } 1 + j < i \leq n$$
$$\text{where } \Gamma_i \vdash x_{ab\to cd} : \bullet \to a_i \to a_i \text{ and } \Gamma_i \vdash y_j : \bullet \text{ for } 1 + j < i \leq n$$
$$\Gamma_{n+1} \vdash M : 1 \implies \Gamma_{n+1} \vdash x_{ab\to cd}\ y_j\ M : 1$$
$$\text{where } \Gamma_{n+1} \vdash x_{ab\to cd} : \bullet \to 1 \to 1 \text{ and } \Gamma_{n+1} \vdash y_j : \bullet$$

In sum, the claim is shown for the term $x_{ab\to cd}\ y_j\ M$. □

The above argumentation is formalized in `Completeness.v` as `Lemma completeness_step`.

Corollary 4.7. Let $n > 0$. If $0^n \twoheadrightarrow_{\mathcal{R}} 1^n$, then there exists a term N such that $\Gamma_\blacktriangle \cup \Gamma_i^n \vdash (x_0\ N) : *$ for $1 \leq i < n$, $\Gamma_\blacktriangle \cup \Gamma_n^n \vdash (x_0\ N) : \#$, and $\Gamma_\blacktriangle \cup \Gamma_{n+1}^n \vdash (x_0\ N) : \$$.

In the following Lemma 4.8 we establish the parallel constraints required for Corollary 4.7.

Lemma 4.8. If $\Gamma_\blacktriangle \cup \Gamma_i^n \vdash N : *$ for $1 \leq i < n$, $\Gamma_\blacktriangle \cup \Gamma_n^n \vdash N : \#$, and $\Gamma_\blacktriangle \cup \Gamma_{n+1}^n \vdash N : \$$ for some $n > 0$ and a term N, then there exists a term N' such that $\Gamma_\blacktriangle \vdash N' : \#$ and $\Gamma_\blacktriangle \vdash N' : \$$.

Proof:
Induction on n.

Basis Step ($n = 1$)**:** Since $\Gamma_1^1 = \Gamma_2^1 = \emptyset$ we immediately obtain the claim for $N' = N$.

Inductive Step ($n > 1$)**:** We have by Lemma 4.2

$$\begin{aligned}
\Gamma_\blacktriangle \cup \Gamma_i^{n+1} \vdash N : * \text{ for } 1 \leq i < n &\implies \Gamma_\blacktriangle \cup \Gamma_i^n \vdash \lambda y_n.N : \bullet \to * \text{ for } 1 \leq i < n \\
&\implies \Gamma_\blacktriangle \cup \Gamma_i^n \vdash x_*\ (\lambda y_n.N) : * \text{ for } 1 \leq i < n \\
\Gamma_\blacktriangle \cup \Gamma_n^{n+1} \vdash N : * &\implies \Gamma_\blacktriangle \cup \Gamma_n^n \vdash \lambda y_n.N : l \to * \\
&\implies \Gamma_\blacktriangle \cup \Gamma_n^n \vdash x_*\ (\lambda y_n.N) : \# \\
\Gamma_\blacktriangle \cup \Gamma_{n+1}^{n+1} \vdash N : \# &\implies \Gamma_\blacktriangle \cup \Gamma_{n+1}^n \vdash \lambda y_n.N : r \to \# \\
\Gamma_\blacktriangle \cup \Gamma_{n+2}^{n+1} \vdash N : \$ &\implies \Gamma_\blacktriangle \cup \Gamma_{n+1}^n \vdash \lambda y_n.N : \bullet \to \$ \\
&\implies \Gamma_\blacktriangle \cup \Gamma_{n+1}^n \vdash x_*\ (\lambda y_n.N) : \$
\end{aligned}$$

In sum, the claim immediately follows by the induction hypothesis. □

The above argumentation is formalized in `Completeness.v` as `Lemma completeness_expand`.

Finally, we can show completeness of the described reduction in the following Theorem 4.9.

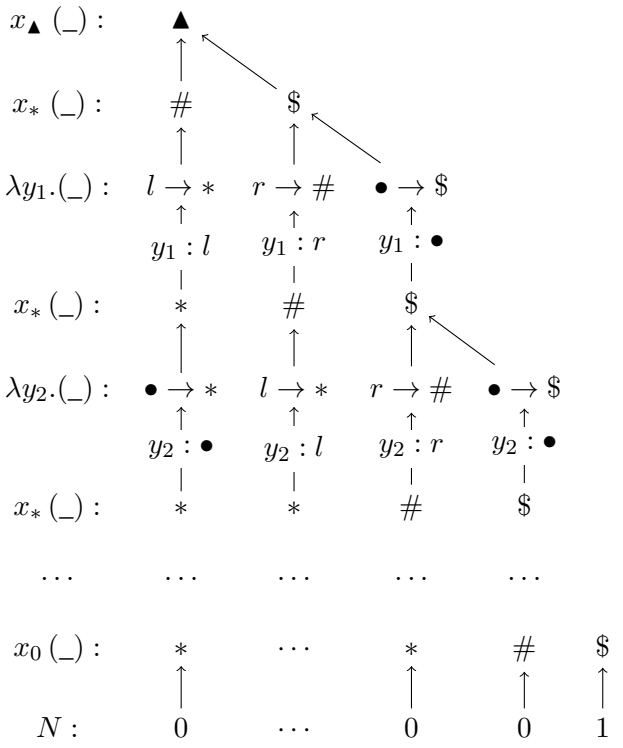
Theorem 4.9. If there exists an $n > 0$ such that $0^n \twoheadrightarrow_{\mathcal{R}} 1^n$, then there exists a term N such that $\Gamma_{\blacktriangle} \vdash N : \blacktriangle$.

Proof:
By Corollary 4.7 there exists a term M such that $\Gamma_{\blacktriangle} \cup \Gamma_i^n \vdash (x_0\ M) : *$ for $1 \leq i < n$, $\Gamma_{\blacktriangle} \cup \Gamma_n^n \vdash (x_0\ M) : \#$, and $\Gamma_{\blacktriangle} \cup \Gamma_{n+1}^n \vdash (x_0\ M) : \$$. Therefore, by Lemma 4.8, there exists a term M' such that $\Gamma_{\blacktriangle} \vdash M' : \#$ and $\Gamma_{\blacktriangle} \vdash M' : \$$. Finally, we obtain $\Gamma_{\blacktriangle} \vdash x_{\blacktriangle}\ M' : \blacktriangle$. □

The above argumentation is formalized in `Completeness.v` as `Lemma completeness`.

Observe that types in the type environment $\Gamma_{\blacktriangle}$ are of at most rank 2 and $\mathrm{rank}(\blacktriangle) = 0$. Therefore, the reduction described by Theorem 4.9 relies on intersection type inhabitation at rank 3.

Lastly, we would like to visualize the construction diagrammatically. The expansion phase introduces neighbor information as types of y_i splitting the last constraint containing $\$$. Afterwards the word representation is initialized as $0 \cdots 0$. The last constraint containing the symbol 1 does not have neighbor information, remaining unchanged.



Second, once expanded, the rewriting happens as follows ending in the representation of the word $1 \cdots 1$ typed by x_1.

$$
\begin{array}{rcccccccccc}
x_{ab\to cd}\ y_j\ (\_): & a_1 & \cdots & a_{j-1} & a & b & a_{j+1} & \cdots & a_n & 1 \\
 & \uparrow & & \uparrow & \uparrow & \uparrow & \uparrow & & \uparrow & \uparrow \\
N[x_1]: & a_1 & \cdots & a_{j-1} & c & d & a_{j+1} & \cdots & a_n & 1 \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
x_1: & 1 & \cdots & 1 & 1 & 1 & 1 & \cdots & 1 & 1
\end{array}
$$

5. Direct reduction from the Halting problem

Composing the reduction form the empty word halting problem for Turing machines to simple semi-Thue rewriting and the reduction from simple semi-Thue rewriting to inhabitation, undecidability of inhabitation is immediate. In this section, we outline a *direct* reduction from the empty word halting problem for Turing machines to inhabitation, as discussed in [23]. Since the underlying type system is slightly different, the reduction given in [23] cannot be used immediately for the Coppo-Dezani type assignment system, and is adjusted accordingly. A direct reduction may be of practical importance so that the described reduction can be embedded into a larger framework of computational reductions in Coq [41] without additional dependencies.

Compared to simple semi-Thue systems, there are several aspects that need to be addressed in order to represent a Turing machine $\mathcal{M} = (\Theta, Q, q_0, q_f, \delta)$, where Θ is the finite, non-empty set of tape symbols, Q is the finite, non-empty set of states, $q_0 \in Q$ is the initial state, $q_f \in Q$ is the final state, and $\delta : (Q \setminus \{q_f\}) \times \Theta \to Q \times \Theta \times \{\text{left}, \text{right}\}$ is the transition function.

First, we need to consider the current state $q \in Q$ and the current head position. For that reason we introduce additional symbols $\{\langle q, a\rangle \mid q \in Q, a \in \Theta\}$.

Second, we need to remember the "left-most" constraint to be initialized to $\langle q_0, \text{␣}\rangle$, i.e. the representation of the initial configuration. This is done using an additional symbol $\circ$ and the types

$$
\begin{aligned}
\sigma_{\blacktriangle} &= \big((l \to \circ) \cap (r \to \#) \cap (\bullet \to \$)\big) \to \blacktriangle \\
\sigma_{*} &= ((\bullet \to \circ) \to \circ) \cap ((\bullet \to *) \to *) \cap ((l \to *) \to \#) \cap ((r \to \#) \cap (\bullet \to \$) \to \$) \\
\sigma_{0} &= (\langle q_0, \text{␣}\rangle \to \circ) \cap (\text{␣} \to *) \cap (\text{␣} \to \#) \cap (\text{␣} \to \$)
\end{aligned}
$$

Third, the accepting configuration is distinguished by q_f. Therefore, the corresponding type representing an accepting configuration is $\sigma_f = \bigcap_{a \in \Theta} \langle q_f, a\rangle \cap \bigcap_{a \in \Theta} a$.

Fourth, the transition function δ may move the head to the left or the right resulting in the following representation

$$\sigma_t = \bigcap_{a \in \Theta} (\bullet \to a \to a) \cap (l \to c' \to \langle q, c \rangle) \cap \bigcap_{a \in \Theta} (r \to \langle q', a \rangle \to a)$$
$$\text{for each } t = ((q, c) \mapsto (q', c', \text{right})) \in \delta$$
$$\sigma_t = \bigcap_{a \in \Theta} (\bullet \to a \to a) \cap (r \to c' \to \langle q, c \rangle) \cap \bigcap_{a \in \Theta} (l \to \langle q', a \rangle \to a)$$
$$\text{for each } t = ((q, c) \mapsto (q', c', \text{left})) \in \delta$$

In sum, we have the following Proposition 5.1

Proposition 5.1. $\mathcal{M}$ accepts the empty word iff there exists a term N such that

$$\{x_\blacktriangle : \sigma_\blacktriangle, x_* : \sigma_*, x_0 : \sigma_0, x_f : \sigma_f\} \cup \{x_t : \sigma_t \mid t \in \delta\} \vdash N : \blacktriangle$$

Overall, except for a far more complicated case analysis, the soundness and completeness proofs of the above proposition are mostly the same as for Theorems 4.5, 4.9 (for more details see [23]).

6. Conclusion

In this work we formalized the (as of yet) simplest version of Urzyczyns undecidability proof [12] for rank 3 inhabitation in a type system encompassing intersection types. Even in its simpler form the rigorous proof requires a heavily branching, repetitive case analysis on the structure of type derivations (around 1000 lines of code for soundness and 500 lines of code for completeness spanning a total of around 70 man hours). Therefore, we believe that a semi-automated (by use of proof tactics) formalization greatly helps to verify the reduction to the very last detail. Among the `omega` tactic (Presburger arithmetic) and `firstorder` tactic (first-order reasoning) a user-defined `list_inclusion` tactics (used for weakening) were most helpful. The used proof structure relies on `ssreflect` tactics for proof management trying to use as few explicit assumption names as possible, however the `reflect` predicate was not used.

The next step is to embed the formalization into the larger framework of *computational reductions in Coq* [41] already containing a collection of formalized reductions that are used in undecidability results. Since the framework of [41] does not yet contain statements on simple semi-Thue systems, formalizing the direct reduction from the halting problem for Turing machines seems appropriate.

References

[1] Coppo M, Dezani-Ciancaglini M. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 1980. **21**(4):685–693. doi:10.1305/ndjfl/1093883253.

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

[3] Barendregt HP, Dekkers W, Statman R. Lambda Calculus with Types. Perspectives in Logic, Cambridge University Press, 2013. ISBN: 978-0-521-76614-2.

[4] Urzyczyn P. The Emptiness Problem for Intersection Types. In: Proceedings of the Ninth Annual Symposium on Logic in Computer Science (LICS '94), Paris, France, July 4-7, 1994. 1994 pp. 300–309. doi:10.1109/LICS.1994.316059.

[5] Urzyczyn P. The Emptiness Problem for Intersection Types. *Journal of Symbolic Logic*, 1999. **64**(3):1195–1215. doi:10.2307/2586625.

[6] Barendregt HP, Coppo M, Dezani-Ciancaglini M. A Filter Lambda Model and the Completeness of Type Assignment. *Journal of Symbolic Logic*, 1983. **48**(4):931–940. doi:10.2307/2273659.

[7] Salvati S. Recognizability in the Simply Typed Lambda-Calculus. In: Ono H, Kanazawa M, de Queiroz RJGB (eds.), WoLLIC 2009, Proceedings of Workshop on Logic, Language, Information and Computation, volume 5514 of *LNCS*. Springer, 2009 pp. 48–60. doi:10.1007/978-3-642-02261-6_5.

[8] Salvati S, Manzonetto G, Gehrke M, Barendregt HP. Urzyczyn and Loader are logically related. In: ICALP 2012, Proceedings of Automata, Languages, and Programming - 39th International Colloquium, volume 7392 of *LNCS*. Springer, 2012 pp. 364–376. doi:10.1007/978-3-642-31585-5_34.

[9] Plotkin G. Lambda definability and logical relations. Technical Report Memorandum SAI-RM-4, School of Artificial Intelligence, University of Edinburgh, 1973.

[10] Loader R. The undecidability of lambda definability. *Logic, Meaning and Computation: Essays in Memory of Alonzo Church*, 2001. pp. 331–342.

[11] Leivant D. Polymorphic Type Inference. In: POPL 1983, Proceedings of the 10th ACM Symposium on Principles of Programming Languages. 1983 pp. 88–98. doi:10.1145/567067.567077.

[12] Urzyczyn P. Inhabitation of Low-Rank Intersection Types. In: TLCA 2009, Proceedings of Typed Lambda Calculus and Applications, volume 5608 of *LNCS*. Springer, 2009 pp. 356–370. doi:10.1007/978-3-642-02273-9_26.

[13] Kuśmierek D. The Inhabitation Problem for Rank Two Intersection Types. In: TLCA 2007, Proceedings of Typed Lambda Calculus and Applications, volume 4583 of *LNCS*. Springer, 2007 pp. 240–254. doi:10.1007/978-3-540-73228-0_18.

[14] Chandra AK, Kozen DC, Stockmeyer LJ. Alternation. *Journal of the ACM*, 1981. **28**(1):114–133. doi:10.1145/322234.322243.

[15] Rehof J, Urzyczyn P. The Complexity of Inhabitation with Explicit Intersection. In: Constable RL, Silva A (eds.), Logic and Program Semantics - Essays Dedicated to Dexter Kozen on the Occasion of His 60th Birthday, volume 7230 of *LNCS*. Springer, 2012 pp. 256–270. doi:10.1007/978-3-642-29485-3_16.

[16] Kurata T, Takahashi M. Decidable properties of intersection type systems. In: TLCA 1995, Proceedings of Typed Lambda Calculus and Applications, volume 902 of *LNCS*. Springer, 1995 pp. 297–311. doi:10.1007/BFb0014060.

[17] Schubert A, Urzyczyn P, Zdanowski K. On the Mints Hierarchy in First-Order Intuitionistic Logic. *Logical Methods in Computer Science*, 2016. **12**(4). doi:10.2168/LMCS-12(4:11)2016.

[18] Schubert A, Urzyczyn P, Zdanowski K. On the Mints Hierarchy in First-Order Intuitionistic Logic. In: Foundations of Software Science and Computation Structures - 18th International Conference, FoSSaCS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings. 2015 pp. 451–465. doi:10.1007/978-3-662-46678-0_29.

[19] Hindley JR. Basic Simple Type Theory. Cambridge Tracts in Theoretical Computer Science, vol. 42, Cambridge University Press, 2008.

[20] Sørensen M, Urzyczyn P. Lectures on the Curry-Howard Isomorphism, volume 149 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, 2006. ISBN:0444520775.

[21] Urzyczyn P. Inhabitation in Typed Lambda-Calculi (A Syntactic Approach). In: TLCA'97, Typed Lambda Calculi and Applications, Proceedings, volume 1210 of *LNCS*. Springer, 1997 pp. 373–389. doi:10.1007/3-540-62688-3_47.

[22] Statman R. Intuitionistic Propositional Logic is Polynomial-space Complete. *Theoretical Computer Science*, 1979. **9**:67–72. doi:10.1016/0304-3975(79)90006-9.

[23] Dudenhefner A, Rehof J. Rank 3 Inhabitation of Intersection Types Revisited (Extended Version). *CoRR*, 2017. **abs/1705.06070**. 1705.06070, URL http://arxiv.org/abs/1705.06070.

[24] Dudenhefner A, Rehof J. 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. doi:10.1145/3009837.3009862.

[25] Dudenhefner A, Rehof J. Typability in Bounded Dimension. In: LICS 2017, Proceedings of the 32nd ACM/IEEE Symposium on Logic in Computer Science, Reykjavik, Iceland, June. 2017. doi:10.1109/LICS.2017.8005127.

[26] Dudenhefner A, Rehof J. Principality and approximation under dimensional bound. *PACMPL*, 2019. **3**(POPL):8:1–8:29. URL https://dl.acm.org/citation.cfm?id=3290321.

[27] Rehof J. Towards Combinatory Logic Synthesis. In: BEAT 2013, 1st International Workshop on Behavioural Types. ACM, 2013 .

[28] Düdder B, Martens M, Rehof J. Staged Composition Synthesis. In: ESOP 2014, Proceedings of European Symposium on Programming, Grenoble, France 2014, volume 8410 of *LNCS*. Springer, 2014 pp. 67–86. doi:10.1007/978-3-642-54833-8_5.

[29] Rehof J, Urzyczyn P. Finite Combinatory Logic with Intersection Types. In: TLCA 2011, Proceedings of Typed Lambda Calculus and Applications, volume 6690 of *LNCS*. Springer, 2011 pp. 169–183. doi:10.1007/978-3-642-21691-6_15.

[30] Düdder B, Martens M, Rehof J, Urzyczyn P. Bounded Combinatory Logic. In: CSL 2012, Proceedings of Computer Science Logic, volume 16 of *LIPIcs*. Schloss Dagstuhl, 2012 pp. 243–258. doi:10.4230/LIPIcs.CSL.2012.243.

[31] Linial S, Post EL. Recursive Unsolvability of the Deducibility, Tarski's Completeness and Independence of Axioms Problems of Propositional Calculus. *Bulletin of the American Mathematical Society*, 1949. **55**:50. MR 10514.

[32] Dudenhefner A, Rehof J. Lower End of the Linial-Post Spectrum. In: TYPES 2017, 23rd International Conference on Types for Proofs and Programs, Budapest, Hungary, 29 May - 1 June 2017. 2017 doi:10.4230/LIPIcs.TYPES.2017.2.

[33] Düdder B, Martens M, Rehof J. Staged Composition Synthesis. In: ESOP 2014, Proceedings. 2014 pp. 67–86. doi:10.1007/978-3-642-54833-8_5.

[34] Heineman GT, Hoxha A, Düdder B, Rehof J. Towards Migrating Object-Oriented Frameworks to Enable Synthesis of Product Line Members. In: Proceedings of the 19th International Conference on Software Product Line, SPLC 2015, Nashville, TN, USA, July 20-24, 2015. 2015 pp. 56–60. doi:10.1145/2791060.2791076.

[35] Bessai J, Dudenhefner A, Düdder B, Martens M, Rehof J. Combinatory Process Synthesis. In: Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques - 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part I. 2016 pp. 266–281. doi:10.1007/978-3-319-47166-2_19.

[36] Bessai J, Chen T, Dudenhefner A, Düdder B, de'Liguoro U, Rehof J. Mixin Composition Synthesis based on Intersection Types. *Logical Methods in Computer Science*, 2018. **14**(1). doi:10.23638/LMCS-14(1:18)2018.

[37] Coppo M, Dezani-Ciancaglini M. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic*, 1980. **21**(4):685–693. doi:10.1305/ndjfl/1093883253.

[38] van Bakel S. Strict Intersection Types for the Lambda Calculus. *ACM Computing Surveys*, 2011. **43**(3). doi:10.1145/1922649.1922657.

[39] van Bakel S. Complete Restrictions of the Intersection Type Discipline. *Theor. Comput. Sci.*, 1992. **102**(1):135–163. doi:10.1016/0304-3975(92)90297-S.

[40] Dudenhefner A. Reduction from simple semi-Thue system rewriting to inhabitation in the Coppo-Dezani type assignment system. `https://github.com/mrhaandi/lambda-cap`. Accessed: 2018-09-13.

[41] Forster Y, Heiter E, Smolka G. Verification of PCP-Related Computational Reductions in Coq. In: Interactive Theorem Proving - 9th International Conference, ITP 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9-12, 2018, Proceedings. 2018 pp. 253–269. doi:10.1007/978-3-319-94821-8_15.