Rank 3 Inhabitation of Intersection Types Revisited

Jan Bessai[1], Andrej Dudenhefner[1], Boris Düdder[1], and Jakob Rehof[1]

Technical University of Dortmund, Dortmund, Germany
{jan.bessai, boris.duedder, andrej.dudenhefner, jakob.rehof}@cs.tu-dortmund.de

Abstract

Intersection types capture deep semantic properties of λ-terms such as normalization and were subject to extensive study for decades [2]. Due to their expressive power, intersection type inhabitation (given a type, does there exist a term having the type?) is undecidable in the standard intersection type system **BCD** [1].

Urzyczyn showed relatively recently [6] that restricted to rank 2 intersection type inhabitation becomes exponential space complete and is undecidable for rank 3 and up. Here, $\mathrm{rank}(\tau) = 0$ if τ is a simple type, $\mathrm{rank}(\sigma \to \tau) = \max(\mathrm{rank}(\sigma) + 1, \mathrm{rank}(\tau))$ and $\mathrm{rank}(\sigma \cap \tau) = \max(1, \mathrm{rank}(\sigma), \mathrm{rank}(\tau))$ otherwise. Later, Salvati et al. discovered the equivalence of intersection type inhabitation and λ-definability [5], which further improved our understanding of the expressiveness of intersection types.

One can take several routes in order to track the reason for undecidability of intersection type inhabitation (abbreviated by IHP, resp. IHP3 for rank 3). In [2] 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 [6] 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 [5] 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. We pinpoint the reason for undecidability by performing a direct reduction from the halting problem for Turing machines to intersection type inhabitation. In particular, we show how arbitrary Turing machine computations can be directly simulated using proof search. The main benefit of the presented approach is its accessibility and simplicity. It can be used to inspect properties of intersection type inhabitation or related systems on a more fine-grained scale than that of rank. Additionally, simulating Turing machine computations by proof search may provide new insights into the line of work, where proof search is regarded as the execution of a logic programming language used for code synthesis [4]. In order to simulate a Turing machine computation, we construct an intersection type $\tau_\star$ that exactly captures individual properties of a Turing machine. Due to the structure of $\tau_\star$, the execution of a proof search algorithm [3] necessarily consists of two phases. During the first phase the simultaneous set of type judgments (used by the algorithm) is expanded to provide room for the simulation of a Turing machine computation. During the second phase the simultaneous set of type judgments is transformed according to the transition function of the Turing machine until the final state is reached. As a result, $\tau_\star$ is inhabitable iff the simulated Turing machine halts. Since our construction is more concise than existing approaches taking no detours, we believe that it is valuable for a better understanding of the expressiveness of intersection type inhabitation.

For the sake of completeness, we outline the construction of $\tau_\star$. Let $M = (\Sigma, Q, q_0, q_f, \delta)$ be a TM. Fix the set of type constants $\mathbb{C} = \Sigma \dot{\cup} \{l, r, \bullet\} \dot{\cup} \{\langle q, a\rangle \mid q \in Q, a \in \Sigma\} \dot{\cup} \{\circ, *, \#, \$\}$. Note that $\llcorner\!\lrcorner \in \Sigma$ is the space symbol. We define the following types:

$$\sigma_f = \bigcap_{c \in \Sigma} (c \cap \langle q_f, c\rangle)$$

for each $t = ((q, c) \mapsto (q', c', +1)) \in \delta$

$$\sigma_t = \bigcap_{a \in \Sigma} (\bullet \to a \to a) \cap (l \to c' \to \langle q, c\rangle) \cap \bigcap_{a \in \Sigma} (r \to \langle q', a\rangle \to a)$$

for each $t = ((q, c) \mapsto (q', c', -1)) \in \delta$

$$\sigma_t = \bigcap_{a \in \Sigma} (\bullet \to a \to a) \cap (r \to c' \to \langle q, c\rangle) \cap \bigcap_{a \in \Sigma} (l \to \langle q', a\rangle \to a)$$

$$\sigma_* = ((\bullet \to \circ) \to \circ) \cap ((\bullet \to *) \to *) \cap ((l \to *) \to \#) \cap ((r \to \#) \cap (\bullet \to \$) \to \$)$$

$$\sigma_0 = ((\bullet \to \langle q_0, \llcorner\!\lrcorner\rangle) \to \circ) \cap ((\bullet \to \llcorner\!\lrcorner) \to *) \cap ((l \to \llcorner\!\lrcorner) \to \#) \cap ((r \to \llcorner\!\lrcorner) \to \$)$$

$$\tau_\star = \sigma_0 \to \sigma_* \to \sigma_f \to \sigma_{t_1} \to \ldots \to \sigma_{t_k} \to (l \to \circ) \cap (r \to \#) \cap (\bullet \to \$)$$

where $\delta = \{t_1, \ldots, t_k\}$

Each of the above types represents one particular feature of M. To provide some intuition:

- l and r link neighboring tape cells.
- $\circ$, $\$$ and $\#$ indicate the first, last and next to the last tape cell during tape expansion.
- $\langle q, a\rangle$ represents that M is at the position reading a in state q.
- σ_f recognizes whether an accepting state is reached.
- σ_t transforms the state according to δ.
- σ_* represents tape expansion.
- σ_0 initializes M to the state q_0 and the empty tape.
- $\tau_\star$ is inhabited iff M accepts the empty word.

References

[1] 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.

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

[3] Martin W. Bunder. The inhabitation problem for intersection types. In James Harland and Prabhu Manyem, editors, *Theory of Computing 2008. Proc. Fourteenth Computing: The Australasian Theory Symposium (CATS 2008), Wollongong, NSW, Australia, January 22-25, 2008. Proceedings*, volume 77 of *CRPIT*, pages 7–14. Australian Computer Society, 2008.

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

[5] S. Salvati, G. Manzonetto, M. Gehrke, and H.P. Barendregt. Urzyczyn and Loader are logically related. In *Proceedings of ICALP 2012*, volume 7392 of *LNCS*, pages 364–376. Springer, 2012.

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