A Short Note on Type-Inhabitation: Formula-Trees vs. Game Semantics [1]

S. Alves, S. Broda

CRACS/INESCTEC & CMUP, DCC-FC, University of Porto

Abstract

This short note compares two different methods for exploring type-inhabitation in the simply typed lambda-calculus, highlighting their similarities.

Keywords. Lambda calculus; Game semantics; Type-inhabitation.

1. Introduction

In the simply typed λ-calculus, the problem of associating to a type a term that inhabits it, known as type inhabitation, has been a major focus of research over the years. Through Curry-Howard's isomorphism the problem is equivalent to provability in the implicational fragment of propositional logic. As such, algorithms for type-inhabitation can be used to indirectly decide provability.

In [5, 7] a new formal method for exploring type inhabitation, called *Formula-Tree Method*, has been presented, which proved to be effective in establishing new results as well as simplifying existing proofs [6, 7, 8]. More recently, another method for studying type inhabitation was given through the use of game semantics [4], where inhabitants are seen as the interpretation of winning strategies in the arena given by a typing. The aim of this note is to clarify the close relationship between the two methods. This will be done by consistently instantiating the used concepts with a fixed type θ, given in Example 1. A formal exposition of all introduced notions and detailed proofs of the correspondence between the methods, stated in Section 5, can be found in [1].

2. Preliminaries

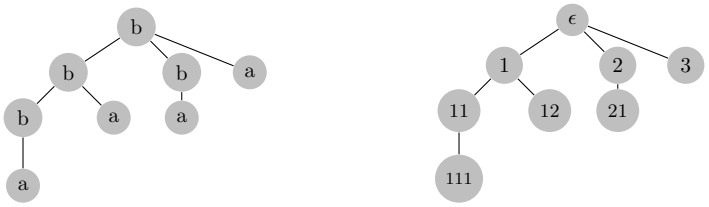
We assume familiarity with basic results on the simply typed λ-calculus, c.f. [9, 2], denoting type-variables by $a, b, c, \ldots$ and types by $\alpha, \beta, \gamma \ldots$ A β-normal inhabitant M of a type γ is called a *long* normal inhabitant of γ iff

[1] Partially funded by FCT, Portuguese Foundation for Science and Technology within project UID/EEA/50014/2013 and CMUP (UID/MAT/00144/2013), which is funded by FCT with national (MEC) and European structural funds through the programs FEDER, under the partnership agreement PT2020.

every variable-occurrence z in M is followed by the longest sequence of arguments, allowed by its type. Ben-Yelles [3, 9], showed that when studying normal inhabitants of a type, one might focus on the set of its long normal inhabitants from which all normal inhabitants can be obtained by η-reduction.

Every type $\gamma = \gamma_1 \to \cdots \to \gamma_n \to a$, with $n \geq 0$, can be uniquely represented by an ordered tree $\mathbf{t}_\gamma$, such that a labels the root, which has n subtrees corresponding to the trees of $\gamma_1, \ldots, \gamma_n$, respectively. Then, one can associate to each node in the tree of γ a unique position $s \in \mathbb{N}_+^\star$, i.e. a finite sequence of elements of $\mathbb{N}_+ = \{1, 2, \ldots\}$, as follows: ϵ is the position of the root, and the root of the ith subtree of another node whose position is s, has position $s \cdot i$.

Example 1 Let $\theta = ((a \to b) \to a \to b) \to (a \to b) \to a \to b$. Its tree $\mathbf{t}_\theta$ as well as the association of positions to nodes can be depicted as follows.



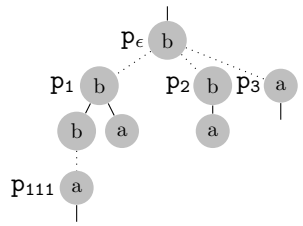
The Böhm tree of a β-normal form $M = \lambda x_1 \ldots x_n.yN_1 \ldots N_m$, $n, m \geq 0$, denoted by $BT(M)$, is the tree with root $\lambda x_1 \ldots x_n.y$, and $m \geq 0$ subtrees $BT(N_1), \ldots, BT(N_m)$. Given a β-normal inhabitant M of a type γ, there is exactly one deduction assigning type γ to M, and in this unique deduction every variable and subterm is given a type. Furthermore, in this deduction all occurrences of a variable x correspond to one occurrence of a subtype γ_x in γ. This particular occurrence of γ_x as well as the corresponding node s_x in the tree $\mathbf{t}_\gamma$ of γ can be easily determined, c.f. [7].

Example 2 The term $M = \lambda xyz.x(\lambda u.x(\lambda v.yv)u)z$ is a long normal inhabitant of type θ from Example 1. For x, y, u and v one has, $\gamma_x = (a \to b) \to a \to b$ and $s_x = 1$, $\gamma_y = a \to b$ and $s_y = 2$, $\gamma_u = \gamma_v = a$ and $s_u = s_v = 111$.

3. The Formula-Tree Method

In the formula-tree method types are represented by formula-trees as follows. The *formula-tree* of a type γ, denoted by FT_γ, is obtained from $\mathbf{t}_\gamma$ by splitting it into *primitive parts*, which are formed by the nodes whose positions are of odd length and their direct descendants. If the root node of a primitive part has position s in $\mathbf{t}_\gamma$, then we associate to it the identifier p_s. Additionally, we consider a primitive part with identifier p_ϵ, consisting of the root node of FT_γ.

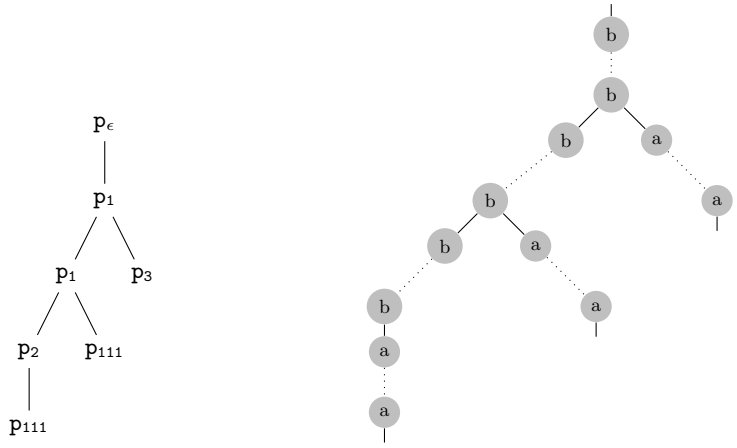
Example 3 The formula-tree FT_γ of θ from Example 1 is the following.



A *proof-tree* for a type γ is a tree PT, labelled with primitive parts of FT_γ, that satisfies the following conditions:

- p_ϵ labels the root of PT and every occurrence of a primitive part p_s in PT, with $n \geq 0$ leafs $a_1, \ldots, a_n$, has exactly n direct descendants $p_{s_1}, \ldots, p_{s_n}$ whose roots are respectively $a_1, \ldots, a_n$.
- Every node in PT with label p_s, such that $s = s' \cdot i \cdot j$, occurs in the ith subtree of some ancestor node in PT with label $p_{s'}$.

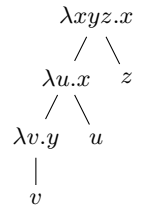
Example 4 The following proof-tree is constructed from FT_θ of Example 3.



From long normal inhabitants to proof-trees and back. Given a long normal inhabitant M of a type γ, we can easily construct a corresponding proof-tree $\mathbf{t}_M$, by replacing every label $\lambda x_1 \ldots x_n.y$ in the Böhm tree of M by p_{s_y}, where s_y is the position corresponding to y in $\mathbf{t}_\gamma$, and adding an extra node with label p_ϵ at the root of this tree.

Conversely, given a proof-tree $\mathbf{t}$ of γ, we compute a finite set of long normal inhabitants $\texttt{Terms}(\mathbf{t})$ as follows. First let $N_\mathbf{t}$ be the *term-scheme* whose Böhm tree is obtained from $\mathbf{t}$ by: first substituting every node labelled with p_s, and descending from the ith leaf of another primitive part $p_{s'}$ in $\mathbf{t}$, by $\lambda x_{s' \cdot i \cdot 1} \ldots x_{s' \cdot i \cdot l}.x_s$, where $l \geq 0$ is the biggest value such that $s' \cdot i \cdot l$ is a position in $\mathbf{t}_\gamma$; then, removing the top node labelled with p_ϵ. Finally, we obtain the finite set $\texttt{Terms}(\mathbf{t})$ from $N_\mathbf{t}$ by renaming all variables in abstraction sequences with identical names, and renaming free occurrences of these variables in the scope of these abstraction sequences in all possible ways, c.f. [7].

Example 5 The proof-tree $\mathbf{t}_M$ corresponding to M from Example 2 can be obtained from its Böhm tree below and is exactly the one depicted in Example 4. For this, just recall that $s_x = 1$, $s_y = 2$, $s_z = 3$ and $s_u = s_v = 111$.



Conversely, we have $N_{\mathbf{t}_M} = \lambda x_1 x_2 x_3.x_1(\lambda x_{111}.x_1(\lambda x_{111}.x_2 x_{111})x_{111})x_3$, and

$$\begin{array}{rcl} \texttt{Terms}(\mathbf{t}_M) & = & \{\lambda x_1 x_2 x_3.x_1(\lambda x_{111}.x_1(\lambda x'_{111}.x_2 x_{111})x_{111})x_3, \\ & & \ \lambda x_1 x_2 x_3.x_1(\lambda x_{111}.x_1(\lambda x'_{111}.x_2 x'_{111})x_{111})x_3\} \\ & =_\alpha & \{\lambda xyz.x(\lambda u.x(\lambda v.yu)u)z,\ \lambda xyz.x(\lambda u.x(\lambda v.yv)u)z\ \}. \end{array}$$

4. Type Inhabitation through Game Semantics

In the method for studying type-inhabitation based on game-semantics, types and positions are respectively represented by arenas and moves.

Let γ be a simple type. The *arena associated to* γ is $A_\gamma = (M_\gamma, \tau_\gamma)$, where $M_\gamma = \{\ s \mid s \text{ is a position of a node in } \mathbf{t}_\gamma\ \}$ is called a set of *moves* and $\tau_\gamma : M_\gamma \longrightarrow \mathcal{A}$ is a typing function mapping each move s to the type-variable at position s in $\mathbf{t}_\gamma$. A move $s \in M_\gamma$ is called a *player move* (P-move) if s is of odd length, and an *opponent move* (O-move), otherwise.

Example 6 The arena $A_\theta = (M_\theta, \tau_\theta)$ of θ from Example 1 is given by $M_\theta = \{\epsilon, 1, 11, 12, 111, 2, 21, 3\}$ and $\tau_\theta(\epsilon) = \tau_\theta(1) = \tau_\theta(11) = \tau_\theta(2) = b$ and $\tau_\theta(12) = \tau_\theta(111) = \tau_\theta(21) = \tau_\theta(3) = a$.

Then, proofs are represented by winning strategies, each winning strategy consisting of a finite set of legal positions. Unlike [4], we only consider sequences of even length to be legal, without any influence on the remaining exposition/results. Consider an arena $A = (M, \tau)$. A *legal position* in A is a non-empty sequence of pairs $S \in (M \times \mathbb{N}_+)^\star$ of the form

$$(s_0, 0) \cdot (s'_0, i_0) \cdot (s_1, 1) \cdot (s'_1, i_1) \cdots (s_n, n) \cdot (s'_n, i_n),$$

where $n \geq 0$, $s_0 = \epsilon$ and for each $k \in \{0, \ldots, n\}$, s_k is an O-move, s'_k is a P-move, $i_k \in \{0, \ldots, k\}$, and $s'_k = s_{i_k} \cdot j$ for some $j \in \mathbb{N}_+$. Furthermore, $k < n$ implies that $s_{k+1} = s'_k \cdot j$, for some $j \in \mathbb{N}_+$.

A finite non-empty set Σ of legal positions in A, closed w.r.t. prefixes of even length, is a *typing strategy* iff $S \cdot (s_k, k) \cdot (s'_{k_1}, i_{k_1}), S \cdot (s_k, k) \cdot (s'_{k_2}, i_{k_2}) \in \Sigma$, with $S \in (M \times \mathbb{N}_+)^\star$, imply that $s'_{k_1} = s'_{k_2}$, $i_{k_1} = i_{k_2}$, and $\tau(s_k) = \tau(s'_{k_1})$.

Let $\mathsf{max}(\Sigma) \subseteq \Sigma$ be the set of sequences in Σ which are maximal with respect to prefixes. A typing strategy Σ in $A = (M, \tau)$, is a called a *winning strategy* iff Σ satisfies the following conditions.

- For all $S \cdot (s_n, n) \cdot (s'_n, i_n) \in \mathsf{max}(\Sigma)$ the position s'_n is maximal in M, i.e. there is no $j \in \mathbb{N}_+$ such that $s'_n \cdot j \in M$.
- If $S \cdot (s_k, k) \cdot (s'_k, i_k) \in \Sigma$ and there is some position $s'_k \cdot j \in M$, then there is some $S \cdot (s_k, k) \cdot (s'_k, i_k) \cdot (s'_k \cdot j, k+1) \cdot (s'_{k+1}, i_{k+1}) \in \Sigma$.

Every winning strategy Σ is completely determined by $\mathsf{max}(\Sigma)$, since it is closed w.r.t. prefixes of even length.

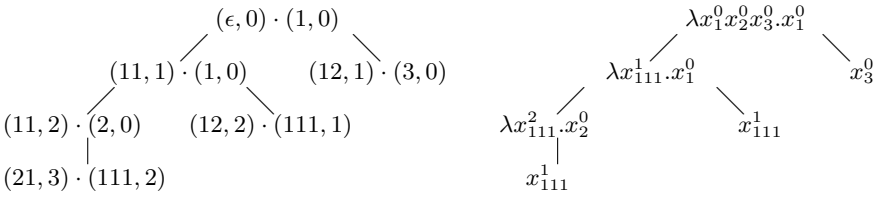
Example 7 Consider again type θ as before and its arena given in Example 6. A winning strategy Σ for θ is given by

$$\begin{aligned}\mathsf{max}(\Sigma) = \{\ & (\epsilon,0)\cdot(1,0)\cdot(11,1)\cdot(1,0)\cdot(11,2)\cdot(2,0)\cdot(21,3)\cdot(111,2),\\ & (\epsilon,0)\cdot(1,0)\cdot(11,1)\cdot(1,0)\cdot(12,2)\cdot(111,1),\\ & (\epsilon,0)\cdot(1,0)\cdot(12,1)\cdot(3,0)\}.\end{aligned}$$

From winning strategies to long normal inhabitants and back. The *arborescent reading* $\mathbb{T}_\Sigma$, of a winning strategy Σ in $A = (M, \tau)$ is a tree obtained from $\mathsf{max}(\Sigma)$ as follows. For every $(s_0, 0) \cdot (s'_0, i_0) \cdots (s_n, n) \cdot (s'_n, i_n) \in \mathsf{max}(\Sigma)$ there is a branch in $\mathbb{T}_\Sigma$ with $n+1$ nodes, labelled respectively by $(s_0, 0) \cdot (s'_0, i_0)$, $\ldots$,$(s_n, n) \cdot (s'_n, i_n)$. Also, if $S \cdot (s_k, k) \cdot (s'_k, i_k) \cdot (s'_k \cdot j, k+1) \cdot (s'_{k+1}, i_{k+1}) \cdot S' \in \mathsf{max}(\Sigma)$, then the node with label $(s'_k \cdot j, k+1) \cdot (s'_{k+1}, i_{k+1})$ is the root of the jth subtree of the node labelled with $(s_k, k) \cdot (s'_k, i_k)$.

Finally, the *interpretation* $[\![\Sigma]\!]$ of Σ is the term whose Böhm tree is obtained from $\mathbb{T}_\Sigma$ by substituting every label $(s_k, k) \cdot (s'_k, i_k)$ by a new label $\lambda x^k_{s_k \cdot 1} \ldots x^k_{s_k \cdot l_k} . x^{i_k}_{s'_k}$, where $l_k \geq 0$ is the biggest value such that $s_k \cdot l_k \in M$.

Example 8 For Σ from Example 7, $\mathbb{T}_\Sigma$ and the Böhm tree of $[\![\Sigma]\!]$ are depicted below. Note that we have precisely $[\![\Sigma]\!] =_\alpha M$, for M from Example 2.



It was proved in [4] that given a winning strategy Σ on an arena A_γ, the term $[\![\Sigma]\!]$ is a long inhabitant of γ. Conversely, given a long inhabitant M a winning strategy Σ can be defined such that $[\![\Sigma]\!] =_{\beta\eta} M$, c.f. [4].

5. Winning Strategies and Proof Trees

In this section we establish the close relationship that exists between winning strategies and proof-trees. In fact, it is straightforward to define transformation algorithms from one approach to the other and back.

Let γ be a type and Σ a winning strategy in $A_\gamma = (M_\gamma, \tau_\gamma)$. The tree $\mathtt{PT}_\Sigma$ is obtained by substituting in the arborescent reading $\mathbb{T}_\Sigma$ every label $(\_, \_) \cdot (\mathbf{s}, \_)$ by $p_{\mathbf{s}}$ and adding an additional root-node with label p_ϵ.

Proposition 9 If Σ is a winning strategy in A_γ, then $\mathtt{PT}_\Sigma$ is a proof-tree for γ.

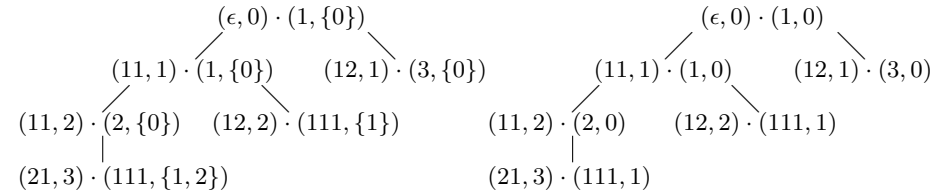
We now present the inverse transformation that given a proof-tree $\mathtt{PT}$ for γ constructs a finite set of winning strategies in the arena $A_\gamma = (M_\gamma, \tau_\gamma)$. In the first step we compute a tree $\mathbb{T}_{\mathtt{PT}}$ such that each node is labelled with two pairs $(s^O, d) \cdot (s^P, \{d_1, \ldots, d_k\})$, where s^O and s^P are respectively an O-move and a P-move, d is the reference of this node, and $\{d_1, \ldots, d_k\}$ is the set of all references to nodes that can enable the P-move s^P in this path.

The construction of $\mathbb{T}_{\mathtt{PT}}$ is done top-down, creating for each node $p_s \neq p_\epsilon$ in $\mathtt{PT}$ that descends from the i-th leaf of another primitive part $p_{s'}$ in $\mathtt{PT}$, a node labelled by $(s' \cdot i, \mathsf{depth}) \cdot (s, \mathsf{Ref})$, where depth is the depth of this node in $\mathbb{T}_{\mathtt{PT}}$ and Ref is the set containing all references d, such that there is some ancestor of this node with label $(s^-, \mathsf{d}), (\_, \_)$ with $s^- \cdot j = s$, for some $j \in \mathbb{N}_+$, as well as the reference depth, whenever $s' \cdot i \cdot j = s$, for some $j \in \mathbb{N}_+$.

Then, the set of winning strategies $\mathtt{WS}(\mathtt{PT})$ is the set of strategies Σ with arborescent reading $\mathbb{T}_\Sigma$ consistent with $\mathbb{T}_{\mathtt{PT}}$, meaning that $\mathbb{T}_\Sigma$ has the same structure as $\mathbb{T}_{\mathtt{PT}}$ and such that each node labelled with $(s^O, d) \cdot (s^P, \{d_1, \ldots, d_k\})$ in $\mathbb{T}_{\mathtt{PT}}$ is labelled with $(s^O, d) \cdot (s^P, d_i)$ in $\mathbb{T}_\Sigma$, where $i \in \{1, \ldots, k\}$.

Proposition 10 If $\mathtt{PT}$ is a proof-tree for γ, then
$\mathtt{WS}(\mathtt{PT}) = \{\ \Sigma \mid \Sigma \text{ is a winning strategy in } A_\gamma \text{ such that } \mathtt{PT}_\Sigma = \mathtt{PT}\ \}$.

Example 11 It is easy to see that the proof-tree $\mathtt{PT}_\Sigma$ corresponding to Σ from Example 7 and its arborescent reading $\mathbb{T}_\Sigma$ in Example 8, is the proof-tree $\mathbf{t}_M$ given in Example 4. On the other hand, the tree $\mathbb{T}_{\mathtt{PT}}$, corresponding to $\mathtt{PT} = \mathbf{t}_{\mathtt{M}}$ is the tree below on the left. Thus $\mathtt{WS}(\mathtt{PT}) = \{\Sigma_1, \Sigma_2\}$, where $\Sigma_1 = \Sigma$ is the winning strategy from Example 7 and the arborescent reading of Σ_2 is the tree below on the right. These two strategies represent precisely the two λ-terms computed in Example 4.



Some considerations on the expressiveness of both methods. We saw that the notions of proof-tree and winning strategy are essentially the same, but for references to preceding primitive parts/O-moves, which are present in winning strategies and missing in proof-trees. In fact, a winning strategy represents exactly one long normal inhabitant and consequently a finite family of normal inhabitants of a type. On the other hand, a proof-tree represents a finite set of long normal inhabitants, corresponding to a possibly bigger finite family of normal inhabitants, which share important properties such as principality, etc. (c.f. [5]). As such, it is natural that in the past both methods have been used for similar purposes. For instance, the formula-tree method was used in

2000, c.f. [5], to characterize principal typings of β-normal terms. An equivalent characterization in terms of game-semantics was given in 2011, c.f. [4]. Also, both methods have been used (respectively in 2002 and 2011) to present a concise proof of Aoto's theorem, which states that negatively non-duplicating types have at most one normal inhabitant (c.f. [6] and [4]). In fact, it seems as if most notions in the game-semantics approach translate easily to the formula-tree approach. On the other hand, results that depend on the absence/presence of references to enabling variables cannot be transferred directly from one world to the other. As an example, it was shown in [10] that it is possible to describe the set of normal inhabitants of a type using an infinitary extension of the concept of context-free grammar, which allows for an infinite number of non-terminal symbols as well as production rules. Later, using the formula-tree approach, it has been shown, c.f. [7], that for every type γ there is in fact a finite context-free grammar G_γ from which all normal inhabitants of γ can be obtained. The existence of this grammar relies on the absence of references and it seems that there is no straightforward counterpart to the construction of a finite grammar in terms of game-semantics. In fact, and in spite of the fact that the two approaches are equivalent, the two methods are not completely identical and different problems may benefit from the features of one or the other.

6. Bibliography

References

[1] S. Alves and S. Broda. Type-Inhabitation: Formula-Trees vs. Game Semantics, (http://www.dcc.fc.up.pt/Pubs/treports.html). Technical Report DCC-2014-08, DCC-FCUP, December 2014.

[2] H.P. Barendregt. Lambda Calculi with Types. In *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Clar. Press, Oxford, 1992.

[3] Ch. Ben-Yelles. *Type Assignment in the Lambda-Calculus: Syntax and Semantics*. PhD thesis, University College of Swansea, September 1979.

[4] P. Bourreau and S. Salvati. Game semantics and uniqueness of type inhabitance in the simply-typed λ-calculus. In *TLCA'11*, LNCS 6690:61-75,2011.

[5] S. Broda and L. Damas. On the structure of normal λ-terms having a certain type. In *Proc. 7th WoLLIC'2000*, pages 33–43, 2000.

[6] S. Broda and L. Damas. Studying provability in implicational intuitionistic logic: the formula tree approach. *ENTCS*, 67:131–147, 2002.

[7] S. Broda and L. Damas. On long normal inhabitants of a type. *J. Log. and Comput.*, 15:353–390, June 2005.

[8] S. Broda, L. Damas, M. Finger, and P. Silva e Silva. The decidability of a fragment of BB'IW-logic. *Theor. Comput. Sci.*, 318(3):373–408, 2004.

[9] J.R. Hindley. *Basic Simple Type Theory*, volume 42 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1997.

[10] M. Takahashi, Y. Akama, and S. Hirokawa. Normal proofs and their grammar. *Information and Computation*, 125(2):144–153, 1996.