Invisible pebbles and the geometry of affine higher-order tree transducers

Lê Thành Dũng (Tito) Nguyễn
LIP, École Normale Supérieure de Lyon, France

Gabriele Vanoni
IRIF, Université Paris Cité, France

Paper draft: `https://nguyentito.eu/tmp/invisible-pebbles.pdf`

This work investigates the expressive power of various kinds of *tree transducers*: automata computing tree-to-tree functions. This is a topic with a long history, and many equivalences between machine models are already known. For instance, the class of monadic second-order transductions (MSOTs), whose name refers to a definition by logic, is also captured by various tree transducer models; it is closed under composition, and includes functions such as:

mirror and add a d above each b:	$b(a(c, a(b(c), c))) \mapsto d(b(a(a(c, d(b(c))), c)))$
relabel each c by parity of its depth:	$b(a(c, a(b(c), c))) \mapsto b(a(0, a(b(0), 1)))$
count number of non-a nodes in unary:	$b(a(c, a(b(c), c))) \mapsto S(S(S(S(S(0)))))$

Some machines for MSOTs, such as the register tree transducers of [2, Section 4], involve *tree contexts* as data structures used in their computation. A tree context is a tree with "holes" at some leaves, e.g. $b(a(c, a(\square, c)))$, and these holes are meant to be substituted by other trees. Thus, it represents a simple *function* on trees. Generalizing this, transducer models using *higher-order data* and inspired by recursion schemes were introduced in the 1980s [6].

In this vein, Gallot, Lemay and Salvati's [7] more recent tree transducer model for MSOTs uses the *linear λ-calculus* to represent higher-order data – linearity corresponds to the "single use" restriction of the register tree transducers of [2]. Independently and around the same time, Nguyễn and Pradic gave another closely related characterization of tree-to-tree MSOTs using linear λ-terms [10, Theorem 1.2.3], in the style of implicit complexity. In this work, to avoid some uninteresting pathologies (cf. [10, Theorem 7.0.2]), we prefer to use an affine λ-calculus; and we consider a simple device in this vein, the "affine λ-transducer".

▶ **Example 1.** The following λ-transducer with memory type $o \multimap o$ takes input trees with binary a-labeled nodes, unary b-labeled nodes and c-labeled leaves, and is specified by:

$$t_a = \lambda lr.\ (\lambda x.\ l\ (r\ x)) \qquad t_b = \lambda f.\ (\lambda x.\ S\ (f\ x)) \qquad t_c = S \qquad u = \lambda f.\ f\ 0$$

where $S : o \multimap o$ and $0 : o$ are constants from the output alphabet. The input $b(a(c, a(b(c), c)))$ is then mapped to $u\ (t_b\ (t_a\ t_c\ (t_a\ (t_b\ t_c)\ t_c)))$ which evaluates to $S\ (S\ (S\ (S\ (S\ 0))))$. So this λ-transducer computes the aforementioned "number of non-a nodes" function.

The issue with this transducer model is its lack of expressiveness, as shown by the following consequence of our results, which settles an equivalent conjecture on "implicit automata" that had been put forth by Nguyễn and Pradic in [12, Section 5.3].

▶ **Corollary 2** (of Theorem 3 below). *There exists a regular tree language whose indicator function cannot be computed by any λ-transducer with purely affine memory.*

More precisely, we work with a λ-calculus whose grammar of types is $A, B ::= o \mid A \multimap B \mid !A$. A type is purely affine when it does not contain '!'. We shall also say that a type is almost purely affine if the only '!'s that it contains are applied to o, e.g. $!o \multimap !o$ is almost purely affine but not $!(o \multimap o)$. This is closely related to Kanazawa's notion of almost affine λ-terms [8].

Gallot et al.'s aforementioned work [7] avoids the limitation evidenced by Corollary 2 by extending the transducer model with common automata-theoretic features (finite states and regular look-ahead). With this, they show that keeping the usual linear λ-terms leads to the previously mentioned characterization of MSOTs, while almost linear λ-terms *à la* Kanazawa [8] yield a larger class called "MSO tree transductions with sharing" (MSOT-S).

Contributions. First, we study (almost) purely affine λ-transducers, relating them to yet another machine model: *tree-walking tree transducers*, see e.g. [4]. Those are devices with a finite-state control and a reading head moving around the nodes of the input tree; in one step, the head can move to the parent or one of the children of the current node.

▶ **Theorem 3.** *We have the following comparisons in expressive power:*

$$\textit{purely affine } \lambda\textit{-transducer} \subseteq \textit{reversible tree-walking transducer}$$
$$\textit{almost purely affine } \lambda\textit{-transducer} \subseteq \textit{tree-walking transducer}$$

Corollary 2 then follows immediately from a result of Bojańczyk and Colcombet [1]: there exists a regular tree language not recognized by any tree-walking automaton. Theorem 3 can also be used to reprove one direction of the equivalences in Gallot et al.'s [7] characterizations of MSOTs and of MSOT-Ss, extending it from linear to affine λ-terms.

We also give a new characterization of the class MSOT-S^2 of functions that can be written as compositions of two MSO transductions with sharing. Let us say that a type is of *almost !-depth 1* when a '!' nested under another '!' can only be applied to o.

▶ **Theorem 4.** Almost !-depth 1 *affine λ-transducer $\circ$ MSO relabeling $\equiv$ MSOT-S^2.*

Preprocessing by an MSO relabeling – which can change the node labels but keeps the tree structure as it is – morally corresponds for generic automata-theoretic reasons to the finite states and regular look-ahead of [7]. To prove the left-to-right inclusion, we compile these λ-transducers to *invisible pebble tree transducers* [4] – an extension of tree-walking transducers known to capture MSOT-S^2.

Key tool: the Interaction Abstract Machine (IAM). To translate λ-transducers into tree-walking or invisible pebble tree transducers, we use a mechanism that evaluates a λ-term using a pointer to its syntax tree that is moved by local steps – just like a tree-walking reading head. This mechanism is none other than the IAM, an operational version of the geometry of interaction (GoI) – cf. [15, Chapter 3] for more on its history.

The IAM satisfies a well-known *reversibility* property (see [15, Proposition 3.3.4] for instance). In the purely affine case, this allows us to get a reversible tree-walking transducer in Theorem 3, defined analogously to the existing literature on reversible graph-walking automata such as [13]. In the almost purely affine and almost !-depth 1 cases, we use an ad-hoc optimization of the IAM that breaks reversibility.

In the theory of higher-order recursion schemes, connections between automata and λ-terms have been established using abstract machines [14] and the GoI [3]. Let us also mention a connection [9] between the categorical GoI and attribute grammars; the latter, when used to define tree-to-tree functions, are essentially the same as tree-walking transducers (cf. [5, Section 3.2]). See [11, §1.1] for more connections of this kind.

References

1 Mikołaj Bojańczyk and Thomas Colcombet. Tree-walking automata do not recognize all regular languages. *SIAM Journal on Computing*, 38(2):658–701, 2008. doi:10.1137/050645427.

2 Mikołaj Bojańczyk and Amina Doumane. First-order tree-to-tree functions, 2020. Corrected version with erratum of a LICS 2020 paper. arXiv:2002.09307v2.

3 Pierre Clairambault and Andrzej S. Murawski. On the Expressivity of Linear Recursion Schemes. In Peter Rossmanith, Pinar Heggernes, and Joost-Pieter Katoen, editors, *44th International Symposium on Mathematical Foundations of Computer Science (MFCS 2019)*, volume 138 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:14. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.MFCS.2019.50.

4 Joost Engelfriet, Hendrik Jan Hoogeboom, and Bart Samwel. XML navigation and transformation by tree-walking automata and transducers with visible and invisible pebbles. *Theoretical Computer Science*, 850:40–97, January 2021. doi:10.1016/j.tcs.2020.10.030.

5 Joost Engelfriet and Sebastian Maneth. A comparison of pebble tree transducers with macro tree transducers. *Acta Informatica*, 39(9):613–698, 2003. doi:10.1007/s00236-003-0120-0.

6 Joost Engelfriet and Heiko Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Informatica*, 26(1/2):131–192, 1988. doi:10.1007/BF02915449.

7 Paul Gallot, Aurélien Lemay, and Sylvain Salvati. Linear high-order deterministic tree transducers with regular look-ahead. In Javier Esparza and Daniel Král', editors, *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 38:1–38:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.MFCS.2020.38.

8 Makoto Kanazawa. Almost affine lambda terms. Technical report NII-2012-003E, National Institute of Informatics, Tokyo, 2012. URL: https://www.nii.ac.jp/TechReports/public_html/12-003E.pdf.

9 Shin-ya Katsumata. Attribute grammars and categorical semantics. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations*, volume 5126 of *Lecture Notes in Computer Science*, pages 271–282. Springer, 2008. doi:10.1007/978-3-540-70583-3_23.

10 Lê Thành Dũng Nguyễn. *Implicit automata in linear logic and categorical transducer theory*. PhD thesis, Université Paris XIII (Sorbonne Paris Nord), December 2021. URL: https://theses.hal.science/tel-04132636.

11 Lê Thành Dũng Nguyễn, Camille Noûs, and Cécilia Pradic. Two-way automata and transducers with planar behaviours are aperiodic, 2023. arXiv:2307.11057.

12 Lê Thành Dũng Nguyễn and Cécilia Pradic. Implicit automata in typed λ-calculi I: aperiodicity in a non-commutative logic. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 135:1–135:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.ICALP.2020.135.

13 Alexander Okhotin. Graph-walking automata: From whence they come, and whither they are bound. In Michal Hospodár and Galina Jirásková, editors, *Implementation and Application of Automata - 24th International Conference, CIAA 2019, Košice, Slovakia, July 22-25, 2019, Proceedings*, volume 11601 of *Lecture Notes in Computer Science*, pages 10–29. Springer, 2019. doi:10.1007/978-3-030-23679-3_2.

14 Sylvain Salvati and Igor Walukiewicz. Simply typed fixpoint calculus and collapsible pushdown automata. *Mathematical Structures in Computer Science*, 26(7):1304–1350, October 2016. doi:10.1017/S0960129514000590.

15 Gabriele Vanoni. *On Reasonable Space and Time Cost Models for the λ-Calculus*. PhD thesis, Alma Mater Studiorum – Università di Bologna, June 2022. doi:10.48676/unibo/amsdottorato/10276.