ON THE COMPLEXITY OF NORMALIZATION FOR THE PLANAR λ-CALCULUS

ANUPAM DAS, DAMIANO MAZZA, LÊ THÀNH DŨNG (TITO) NGUYỄN, NOAM ZEILBERGER

Recall that an untyped λ-term t is *linear* if there exists a list Γ – the list of free variables in t – such that $\Gamma \vdash t$ is derivable with the rules below (with Γ and Δ disjoint in *app*):

$$\frac{}{x \vdash x}\ var \qquad \frac{\Gamma \vdash t \quad \Delta \vdash u}{\Gamma, \Delta \vdash t\, u}\ app \qquad \frac{\Gamma, x \vdash t}{\Gamma \vdash \lambda x.t}\ lam \qquad \frac{\Gamma, y, x, \Delta \vdash t}{\Gamma, x, y, \Delta \vdash t}\ exc$$

Call a linear λ-term t *planar* when there is an *ordered* list Γ such that $\Gamma \vdash t$ is derivable in the subsystem *without the exc rule*: for example, $\lambda x.\, \lambda y.\, f\, x\, y$ is planar but $\lambda x.\, \lambda y.\, f\, y\, x$ is not. Planar λ-terms are closed under β-reduction. Furthermore, this notion is motivated by semantics (non-symmetric monoidal closed categories), topology (a linear λ-term is planar when its representation as a syntax tree with binding edges is a planar combinatorial map) and linguistics (in the Lambek calculus [Lam58], a precursor of linear logic).

Less attention has been paid, however, to the *computational* consequences of planarity. There is a recent implicit complexity result [NP20] using planar λ-terms, where general linear λ-terms would be too expressive. Here, we focus on the *complexity* of *normalizing* λ-terms, asking ourselves whether planarity lowers it. For linear (possibly non-planar) λ-terms, we know that:

Theorem 0.1 ([Mai04])**.** *The following decision problem is* P*-complete under logarithmic space reductions:*

- *Input: two (untyped) linear λ-terms t and u.*
- *Output: are t and u β-convertible, that is, do they have the same normal form?*

(Note that the complexity of the β-convertibility problem for simply typed (possibly non-linear) λ-terms is much higher, namely TOWER-complete – this is implicit in [Sta79], as explained in [Ngu23].)

We believe that this problem is still P-complete when t and u are planar. Two years ago, we claimed this as a theorem[1] but the proposed proof – which purported to provide a logspace reduction from the Circuit Value Problem (CVP), just like Mairson's proof of Theorem 0.1 – contained a subtle yet serious flaw, described at the end of Section 2.

In this extended abstract, we outline another attempt to reduce CVP to planar normalization.

1. The Circuit Value Problem

For our purposes, a *boolean circuit* with n gates can be seen as a list of n equations defining the values of the boolean variables $x_1, \ldots, x_n$, such as the following example:

$$x_1 := 1;\ x_2 := 0;\ x_3 := 1;\ x_4 = x_1 \wedge x_2;\ x_5 = \neg x_1;\ x_6 = x_5 \wedge x_3;\ x_7 = x_4 \vee x_6$$

Here, equations 4 to 7 define $x_7 = (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3) = \mathsf{if}\ x_1\ \mathsf{then}\ x_2\ \mathsf{else}\ x_3$, so the final result of the circuit is $(\mathsf{if}\ 1\ \mathsf{then}\ 0\ \mathsf{else}\ 1) = 0$. In each equation, the right-hand side contains either a constant $0/1$ or the application of an operator $\neg, \wedge, \vee$. Furthermore, we require that in the latter case, the arguments given to the operator have been defined *before* the current equation; in other words, the enumeration $x_1, x_2, \ldots$ is a *topological ordering* of the circuit.

Theorem 1.1 ([GHR95, Theorem 6.2.1])**.** *The* Topologically Ordered Circuit Value Problem (TopCVP)*, defined below, is* P*-complete.*

- *Input: a topologically ordered boolean circuit, as in the above example.*
- *Output: the final value computed by the circuit.*

[1] In a talk at the Structure Meets Power 2021 workshop: `http://noamz.org/talks/smp.2021.06.28.pdf`

2. Planar booleans do not suffice

To encode the Circuit Value Problem in the linear λ-calculus, Mairson [Mai04] uses a linear encoding of booleans. Unfortunately, his encoding represents 0 as a non-planar λ-term, namely $\lambda x.\, \lambda y.\, \lambda f.\, f\, y\, x$.

A planar linear encoding of booleans has been introduced in [Ngu21, Chapter 7] to give a strictly *linear* variant of the previously mentioned result of [NP20], whose original statement used planar *affine* λ-terms.

$$\texttt{false} = \lambda k.\, \lambda f.\, k\, f\, (\lambda x.\, x) \qquad \texttt{true} = \lambda k.\, \lambda f.\, k\, (\lambda x.\, x)\, f$$

While our reduction targets untyped λ-terms, it can be useful to think of these terms as the only inhabitants in normal form of the type

$$\texttt{Bool} = \forall \alpha\beta.\, ((\alpha \multimap \alpha) \multimap (\alpha \multimap \alpha) \multimap \beta) \multimap (\alpha \multimap \alpha) \multimap \beta$$

This can be seen as the image, by a continuation-passing-style transformation, of an encoding using linear λ-terms *with pairs* proposed by Matsuoka [Mat15] in his alternative proof of Theorem 0.1:

$$\texttt{false}' = \lambda f.\, (f,\, \lambda x.\, x) \qquad \texttt{true}' = \lambda f.\, (\lambda x.\, x,\, f) \qquad \texttt{Bool}' = \forall \alpha.\, (\alpha \multimap \alpha) \multimap (\alpha \multimap \alpha) \otimes (\alpha \multimap \alpha)$$

We can also define boolean connectives acting on the encodings of [Ngu21] (we have $\texttt{cstt}\, b =_\beta \texttt{true}$ and $\texttt{cstf}\, b =_\beta \texttt{false}$ for $b \in \{\texttt{true}, \texttt{false}\}$), using the notations $\texttt{id} = \lambda x.\, x$ and $f \circ g = \lambda x.\, f\, (g\, x)$:

$$\begin{aligned}
\texttt{cstt} &= \lambda b.\, \lambda k.\, \lambda f.\, b\, (\lambda g.\, \lambda h.\, k\, \texttt{id}\, (g \circ h))\, f \\
\texttt{cstf} &= \lambda b.\, \lambda k.\, \lambda f.\, b\, (\lambda g.\, \lambda h.\, k\, (g \circ h)\, \texttt{id})\, f \\
\texttt{not} &= \lambda b.\, b\, (\lambda g.\, \lambda h.\, g\, (\texttt{cstt}\, (h\, \texttt{true})))\, \texttt{cstf} \\
\texttt{and} &= \lambda b_1.\, \lambda b_2.\, \lambda k.\, b_1\, (\lambda f_1.\, b_2\, (\lambda f_2.\, \lambda f_3.\, k\, (\lambda x.\, f_1\, (f_2\, x))\, f_3))
\end{aligned}$$

(disjunction can be derived by De Morgan's laws). This is enough to translate boolean *formulas* into planar linear λ-terms.

However, to transpose Mairson's methodology for encoding boolean circuits to the planar linear setting, we would need a planar λ-term $\texttt{copy}$ such that (similarly to the W combinator in Curry's BCKW)

$$\forall t \in \{\texttt{true}, \texttt{false}\},\ \texttt{copy}\, f\, t =_\beta f\, t\, t$$

We have not been able to find such a term; we did manage to define a planar λ-term $\texttt{copy}'$ that satisfies $\texttt{copy}'\, t\, f =_\beta f\, t\, t$, but this is significantly different in a planar setting. Hence the gap in our previous attempt at reducing CVP to β-convertibility of planar λ-terms.

3. A new encoding of TopCVP

Our new idea is to work with an encoding of *bit vectors*, on which we implement the following operations:

$$\mathsf{not}_{i,n}(\langle x_1, \ldots, x_n \rangle) = \langle x_1,\, \ldots,\, x_n,\, \neg x_i \rangle \qquad \mathsf{and}_{i,j,n}(\langle x_1, \ldots, x_n \rangle) = \langle x_1,\, \ldots,\, x_n,\, x_i \wedge x_j \rangle$$

and the analogous $\mathsf{or}_{i,j,n}, \mathsf{false}_n, \mathsf{true}_n \colon \{0,1\}^n \to \{0,1\}^{n+1}$ for $1 \le i, j \le n$. Let also $\mathsf{last}_n(\langle x_1, \ldots, x_n \rangle) = x_n$.

The value of our example circuit from Section 1 can then be expressed, using these operations, as

$$\mathsf{last}_7 \circ \mathsf{or}_{4,6,6} \circ \mathsf{and}_{5,3,5} \circ \mathsf{not}_{1,4} \circ \mathsf{and}_{1,2,3} \circ \mathsf{true}_2 \circ \mathsf{false}_1 \circ \mathsf{true}_0(\langle\rangle)$$

3.1. **Representation of bit vectors.** Unsurprisingly, we use the Church encoding of k-tuples, together with the above-mentioned type $\texttt{Bool}$, to represent vectors of k bits. For instance, $\langle 0, 1, 0 \rangle$ is encoded as

$$\overline{\langle 0, 1, 0 \rangle} = \lambda k.\, k\, \texttt{false}\, \texttt{true}\, \texttt{false}$$

which should be seen as an inhabitant of the type

$$\texttt{Bool}_3 = \forall \gamma.\, (\texttt{Bool} \multimap \texttt{Bool} \multimap \texttt{Bool} \multimap \gamma) \multimap \gamma$$

3.2. **Implementing vectorial operations.** First, we implement $\mathsf{fetch}_{i,n}(\langle x_1, \dots, x_n \rangle) = \langle x_1, \dots, x_n, x_i \rangle$:

$$\mathtt{fetch}_{i,n} = \lambda v.\, v\, (\lambda x_1.\, \dots\, \lambda x_n.\, x_1\, c_1\, f_1\, (\dots (x_n\, c_n\, f_n\, \overline{\langle 0, \dots, 0 \rangle}) \dots))$$

$$\text{where } c_j = \begin{cases} \lambda g.\, \lambda h.\, (g \circ (\lambda k.\, \lambda b_1.\, \dots\, \lambda b_{n+1}.\, k\, b_1 \dots b_{i-1}\, (\mathtt{cstt}\, b_i)\, b_{i+1}\, \dots\, b_n\, (\mathtt{cstt}\, b_{n+1})) \circ h) & \text{if } i = j \\ \lambda g.\, \lambda h.\, (g \circ (\lambda k.\, \lambda b_1.\, \dots\, \lambda b_{n+1}.\, k\, b_1 \dots b_{j-1}\, (\mathtt{cstt}\, b_j)\, b_{j+1}\, \dots\, b_{n+1}) \circ h) & \text{otherwise} \end{cases}$$

$$\text{and } f_j = \begin{cases} \lambda k.\, \lambda b_1.\, \dots\, \lambda b_{n+1}.\, k\, b_1 \dots b_{i-1}\, (\mathtt{cstf}\, b_i)\, b_{i+1}\, \dots\, b_n\, (\mathtt{cstf}\, b_{n+1}) & \text{if } i = j \\ \lambda k.\, \lambda b_1.\, \dots\, \lambda b_{n+1}.\, k\, b_1 \dots b_{j-1}\, (\mathtt{cstf}\, b_j)\, b_{j+1}\, \dots\, b_{n+1} & \text{otherwise} \end{cases}$$

Note that by replacing every $\mathtt{cstt}\, b_{n+1}$ by $\mathtt{cstf}\, b_{n+1}$ and vice versa, we get an implementation of $\mathsf{not}_{i,n}$!

We then set $\mathtt{and}_{i,j,n} = \mathtt{and}'_n \circ \mathtt{fetch}_{i,n+1} \circ \mathtt{fetch}_{j,n}$ where $\mathtt{and}'_n$ implements an in-place conjunction

$$\mathsf{and}'_n(\langle x_1, \dots, x_{n+2} \rangle) = \langle x_1,\, \dots,\, x_n,\, x_{n+1} \wedge x_{n+2} \rangle$$

To define $\mathtt{and}'_n$, we reuse the planar λ-term $\mathtt{and}$ that implements the conjunction on the booleans of Section 2:

$$\mathtt{and}'_n = \lambda v.\, \lambda k.\, v\, (\lambda x_1.\, \dots\, \lambda x_{n+2}.\, k\, x_1\, \dots\, x_n\, (\mathtt{and}\, x_{n+1}\, x_{n+2}))$$

Finally, we take:

$$\mathtt{last}_{i,n} = \lambda v.\, v\, (\lambda x_1.\, \dots\, \lambda x_n.\, x_1\, (\lambda g.\, \lambda h.\, g \circ h)\, \mathtt{id}\, (\dots (x_{n-1}\, (\lambda g.\, \lambda h.\, g \circ h)\, \mathtt{id}\, x_n) \dots))$$

References

[GHR95] Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to Parallel Computation: P-Completeness Theory*. Oxford University Press, 06 1995. doi:10.1093/oso/9780195085914.001.0001.

[Lam58] Joachim Lambek. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170, 1958. doi:10.1080/00029890.1958.11989160.

[Mai04] Harry G. Mairson. Linear lambda calculus and PTIME-completeness. *Journal of Functional Programming*, 14(6):623–633, November 2004. doi:10.1017/S0956796804005131.

[Mat15] Satoshi Matsuoka. A new proof of P-time completeness of linear lambda calculus. In Ansgar Fehnker, Annabelle McIver, Geoff Sutcliffe, and Andrei Voronkov, editors, *20th International Conferences on Logic for Programming, Artificial Intelligence and Reasoning - Short Presentations, LPAR 2015, Suva, Fiji, November 24-28, 2015*, volume 35 of *EPiC Series in Computing*, pages 119–130. EasyChair, 2015. doi:10.29007/svwc.

[Ngu21] 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://nguyentito.eu/thesis.pdf.

[Ngu23] Lê Thành Dũng Nguyễn. Simply typed convertibility is TOWER-complete even for safe lambda-terms, 2023. arXiv:2305.12601.

[NP20] 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.

[Sta79] Richard Statman. The typed λ-calculus is not elementary recursive. *Theoretical Computer Science*, 9:73–81, 1979. Journal version of a FOCS 1977 paper. doi:10.1016/0304-3975(79)90007-0.