A Note on the Complexity of Associative-Commutative Lambek Calculus

Christophe Costa Florêncio
Utrecht University

1. Introduction

In this paper the NP-completeness of the system **LP** (associative-commutative Lambek calculus) will be shown. The complexity of **LP** has been known for some time, it is a corollary of a result for multiplicative intuitionistic linear logic (**MILL**)[1] from (Kanovich, 1991) and (Kanovich, 1992).

We show that this result can be strengthened: **LP** remains NP-complete under certain restrictions. The proof does not depend on results from the area of linear logic, it is based on a simple linear-time reduction from the minimum node-cover problem to recognizing sentences in **LP**.

2. Definitions

First some definitions are in order:

Definition 1 *The* degree *of a type is defined as*

$$\begin{array}{lcl} \mathrm{degree}(A) & = & 0 \textit{ if } A \in \mathrm{Pr} \\ \mathrm{degree}(B\backslash A) & = & 1 + \mathrm{degree}(A) + \mathrm{degree}(B) \\ \mathrm{degree}(A/B) & = & 1 + \mathrm{degree}(A) + \mathrm{degree}(B) \end{array}$$

In other words, the degree of a type can be determined by counting the number of operators it contains.

Definition 2 *The* Order *of a type is defined as*

$$\begin{array}{lcl} \mathrm{order}(A) & = & 0 \textit{ if } A \in \mathrm{Pr} \\ \mathrm{order}(B\backslash A) & = & \max(1 + \mathrm{order}(A) + \mathrm{order}(B)) \\ \mathrm{order}(A/B) & = & \max(1 + \mathrm{order}(A) + \mathrm{order}(B)) \end{array}$$

Definition 3 *A* domain subtype *is a subtype that is in domain position, i.e. for the type $((A/B)/C)$ the domain subtypes are B and C.*
For the type $(C\backslash(B\backslash A))$ the domain subtypes are C and B.

A range subtype *is a subtype that is in range position, i.e. for the type $((A/B)/C)$ the range subtypes are (A/B) and A.*
For the type $(C\backslash(B\backslash A))$ the range subtypes are $(B\backslash A)$ and A.

In an applicaton $A/B, B \vdash A$ or $B, B\backslash A \vdash A$ the type B is an argument *and A/B and $B\backslash A$ are known as* functors*.*

Definition 4 *Let $G = (V, E)$ be an undirected graph, where V is a set of nodes and E is a set of edges, represented as tuples of nodes. A* node-cover *of G is a subset $V' \subseteq V$ such that if $(u, v) \in E$, then $u \in V'$ or $v \in V'$. That is, each node 'covers' its incident edges, and a node cover for G is a set of nodes that covers all the edges in E. The* size *of a node-cover is the number of nodes in it.*

The node-cover problem *is the problem of finding a node-cover of minimum size (called an* optimal node-cover*) in a given graph.*

The node-cover problem can be restated as a decision problem*: does a node-cover of given size k exist for some given graph?*

Proposition 5 *The decision problem related to the node-cover problem is* NP*-complete, The node-cover problem is* NP*-hard.*

This problem has been called one of the 'six basic NP-complete problems' in (Garey and Johnson, 1979).

1. The systems **LP** and **MILL** are identical up to derivation from the empty sequent, i.e. the only difference is that $\vdash n/n$ is not derivable in **LP**.
The system **MILL** is closely related to **MILL1**, another system that has interesting linguistic applications, see (Moot and Piazza, 2001).

 Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6), pp. 159–162. Universitá di Venezia.

3. Complexity of LP

Theorem 6 *Deciding membership for the unidirectional product-free fragment of* **LP**, with all types restricted to a maximum degree of 2 and a maximum order of 1, is NP-complete in $|\Sigma|$.

`Proof`: It is well known that **LP** is in NP.

What remains to be shown is existence of a p-time reduction from an NP-complete problem. Let $G = (E, V)$ be an undirected graph, $ne = |E|$. Let $C = \mathrm{C}(G)$ be a minimum node cover of G, and $\min(G) = |\mathrm{C}(G)|$. The graph G can be reduced to a grammar $Gr = \mathrm{gram}(G)$ as follows:

1. Assign s to s.
2. Let f be the function that maps node V_n to type v_n. For every edge $E_x \in E$, where $E_x = \langle V_y, V_z\rangle$, let $v_y = f(V_y), v_z = f(V_z)$. Assign types $v_y\backslash v_y$, $v_y\backslash(s\backslash s)$ and $v_z\backslash v_z$, $v_z\backslash(s\backslash s)$ to symbol v_x.
3. For every node $V_n \in V$, assign $f(V_n) = v_n$ to node.

The intuition behind this reduction is that node stands for any node in G, and e_x for the *connection* of edge E_x to any of the two nodes it is incident on.

Note that this reduction always yields a unidirectional product-free grammar, with all types restricted to a maximum degree of 2 and a maximum order of 1. Also note that this reduction sets $|\Sigma|$ to the number of edges plus two.

We will now show that accepting a sentence s of the form $\mathsf{s}\underbrace{\mathsf{node}\ldots\mathsf{node}}_{i\ \text{times}}\ \mathsf{v}_1\ldots\ \mathsf{v}_{ne}$ as being in $\mathrm{L}(\mathrm{gram}(G))$ while rejecting $\mathsf{s}\underbrace{\mathsf{node}\ldots\mathsf{node}}_{i-1\ \text{times}}\ \mathsf{v}_1\ldots\ \mathsf{v}_{ne}$ will indicate that there is a node cover of size i for G. Simply iterating from $i = 1$ to $i = ne$ will lead to acceptance when $i = \min(G)$.

Parsing such a sentence will yield a *solution*: one can collect the assignments to the symbol node used in the derivation to obtain a minimum node cover.

Let T be some set of types (taken from the assignments to node in $\mathrm{gram}(G)$) assigned to the substring $\underbrace{\mathsf{node}\ldots\mathsf{node}}_{i\ \text{times}}$ of s. Let U be some set of types assigned to the substring $\mathsf{v}_1\ldots\mathsf{v}_{ne}$ under the same restrictions.

1. Assume that $i < \min(G)$. Since by the form of s $|T| \leq i$, $|T| < \min(G)$, so for every minimum node cover C, there is a $V_n \in C$ such that $f(V_n) \notin T$. Since for every edge $\langle V_y, V_z\rangle \in E$, there is some v_n in s that has been assigned either the type $v_x\backslash v_x$ or $v_x\backslash(s\backslash s)$, $v_x = f(V_y)$ or $v_x = f(V_z)$.

   Since for every edge $\langle V_y, V_z\rangle \in E$, $f(V_y) \in C$ or $f(V_z) \in C$, there is some v_m in s that has been assigned $v_n\backslash v_n$ or $v_n\backslash(s\backslash s)$, $v_n \notin T$.

   Since $\Gamma, pT, \Gamma' \not\vdash_{\mathbf{LP}} \Gamma, \Gamma'$ (where pT is a primitive type), in order to derive (just) s, all the types in T have to occur as argument to an application in the derivation. Given the form of $\mathrm{gram}(G)$ this is possible just if the functor is a type assigned to $\mathsf{v}_{1\leq n\leq ne}$. Thus $s_{1\leq i<\min(G)} \notin \mathrm{L}(\mathrm{gram}(G))$.

2. Assume $i = \min(G)$. Then there is a T such that $|T| = i$. Let Tc be $\{f(V_n)|V_n \in C\}$, for some C. Given s and assignments of types such that for each $1 \leq p < ne$, $v_p\backslash(s\backslash s)$ occurs at most once ...

   Since **LP** is associative and commutative any rearrangment is allowed during a derivation. This property can be used to 'sort' the assignments to the symbols node and v_n in the following way: each occurrence of node (assigned type $v_x \in Tc$) is followed by all v_n's that are assigned type $v_x\backslash v_x$, followed by a single v_n assigned $v_n\backslash(s\backslash s)$. The substring thus obtained is associated with a sequent that derives $(s\backslash s)$. The whole of s minus s, can be arranged into a number of these substrings, and since $A\backslash A, A\backslash A \vdash_{\mathbf{LP}} A\backslash A$, the associated sequent will derive $s\backslash s$. Since s is only assigned s in $\mathrm{gram}(G)$, we finally get the derivation $s, s\backslash s \vdash s$.

This shows that the reduction given is indeed a reduction from an NP-complete problem. □

Example: Reducing $G = (\{(1,2),(1,3),(3,4),(2,4)\},\{1,2,3,4\})$ will yield

$$
\mathrm{gram}(G):\quad
\begin{array}{rcl}
\mathsf{s} & \mapsto & s\\
\mathsf{v}_1 & \mapsto & v_1\backslash v_1, v_1\backslash(s\backslash s), v_2\backslash v_2, v_2\backslash(s\backslash s)\\
\mathsf{v}_2 & \mapsto & v_1\backslash v_1, v_1\backslash(s\backslash s), v_3\backslash v_3, v_3\backslash(s\backslash s)\\
\mathsf{v}_3 & \mapsto & v_3\backslash v_3, v_3\backslash(s\backslash s), v_4\backslash v_4, v_4\backslash(s\backslash s)\\
\mathsf{v}_4 & \mapsto & v_2\backslash v_2, v_2\backslash(s\backslash s), v_4\backslash v_4, v_4\backslash(s\backslash s)\\
\mathsf{node} & \mapsto & v_1, v_2, v_3, v_4
\end{array}
$$

The corresponding minimal node cover is $\{1,4\}$ or $\{2,3\}$.

As a final remark, note that there exists an alternative reduction $\mathrm{gram}'(G)$:

1. Assign s to s.

2. For every edge $E_x \in E$, where $E_x = \langle V_y, V_z\rangle$, let $v_y = f(V_y), v_z = f(V_z)$. Assign types $v_y\backslash v_y$ and $v_z\backslash v_z$ to symbol e_x.

3. For every node $V_n \in V$, assign $v_x\backslash(s\backslash s)$ to c and $f(V_n) = v_n$ to node.

   Example: Applying this procedure to the same graph yields:

$$
\mathrm{gram}'(G):\quad
\begin{array}{rcl}
\mathsf{s} & \mapsto & s\\
\mathsf{v}_1 & \mapsto & v_1\backslash v_1, v_2\backslash v_2\\
\mathsf{v}_2 & \mapsto & v_1\backslash v_1, v_3\backslash v_3\\
\mathsf{v}_3 & \mapsto & v_3\backslash v_3, v_4\backslash v_4\\
\mathsf{v}_4 & \mapsto & v_2\backslash v_2, v_4\backslash v_4\\
\mathsf{node} & \mapsto & v_1, v_2, v_3, v_4\\
\mathsf{c} & \mapsto & v_1\backslash(s\backslash s), v_2\backslash(s\backslash s), v_3\backslash(s\backslash s), v_4\backslash(s\backslash s)
\end{array}
$$

Accepting a sentence of the form $\mathsf{s}\,\underbrace{\mathsf{node}\ldots\mathsf{node}}_{i \text{ times}}\ \mathsf{v}_1\ldots\ \mathsf{v}_{ne}\,\underbrace{\mathsf{c}\ldots\mathsf{c}}_{i \text{ times}}$ as being in $\mathrm{L}(\mathrm{gram}(G))$ will indicate that there is a node cover of size i for G. Again, iterating from $i = 1$ to $i = ne$ will lead to acceptance when $i = \min(G)$.

4. Example Derivations

Given graph $G = (\{(1,2),(1,3),(3,4),(2,4)\},\{1,2,3,4\})$, the grammar $\mathrm{gram}(G)(G)$ and sentence '`s node node  v1 v2 v3 v4`' ($i = 4$) we get the solutions shown in Figures 1 and 2.

$$
\dfrac{
\dfrac{
\mathsf{s}\vdash s
\quad
\dfrac{
\dfrac{\mathsf{node}\vdash v_1 \quad \mathsf{v1}\vdash v_1\backslash v_1}{\mathsf{node}\circ\mathsf{v1}\vdash v_1}[\backslash E]
\quad \mathsf{v2}\vdash v_1\backslash(s\backslash s)}
{(\mathsf{node}\circ\mathsf{v1})\circ\mathsf{v2}\vdash s\backslash s}[\backslash E]
}{\mathsf{s}\circ((\mathsf{node}\circ\mathsf{v1})\circ\mathsf{v2})\vdash s}[\backslash E]
\quad
\dfrac{
\dfrac{\mathsf{node}\vdash v_4 \quad \mathsf{v3}\vdash v_4\backslash v_4}{\mathsf{node}\circ\mathsf{v3}\vdash v_4}[\backslash E]
\quad \mathsf{v4}\vdash v_4\backslash(s\backslash s)}
{(\mathsf{node}\circ\mathsf{v3})\circ\mathsf{v4}\vdash s\backslash s}[\backslash E]
}{(\mathsf{s}\circ((\mathsf{node}\circ\mathsf{v1})\circ\mathsf{v2}))\circ((\mathsf{node}\circ\mathsf{v3})\circ\mathsf{v4})\vdash s}[\backslash E]
$$

$$
\begin{array}{ll}
(\mathsf{s}\circ((\mathsf{node}\circ\mathsf{v1})\circ\mathsf{v2}))\circ((\mathsf{node}\circ\mathsf{v3})\circ\mathsf{v4})\vdash s & \\ \hline
(\mathsf{s}\circ((\mathsf{node}\circ\mathsf{v1})\circ\mathsf{v2}))\circ(\mathsf{node}\circ(\mathsf{v3}\circ\mathsf{v4}))\vdash s & [ass]\\ \hline
(\mathsf{s}\circ(\mathsf{node}\circ(\mathsf{v1}\circ\mathsf{v2})))\circ(\mathsf{node}\circ(\mathsf{v3}\circ\mathsf{v4}))\vdash s & [ass]\\ \hline
((\mathsf{s}\circ\mathsf{node})\circ(\mathsf{v1}\circ\mathsf{v2}))\circ(\mathsf{node}\circ(\mathsf{v3}\circ\mathsf{v4}))\vdash s & [ass]\\ \hline
(\mathsf{s}\circ\mathsf{node})\circ((\mathsf{v1}\circ\mathsf{v2})\circ(\mathsf{node}\circ(\mathsf{v3}\circ\mathsf{v4})))\vdash s & [ass]\\ \hline
(\mathsf{s}\circ\mathsf{node})\circ(((\mathsf{v1}\circ\mathsf{v2})\circ\mathsf{node})\circ(\mathsf{v3}\circ\mathsf{v4}))\vdash s & [ass]\\ \hline
(\mathsf{s}\circ\mathsf{node})\circ((\mathsf{node}\circ(\mathsf{v1}\circ\mathsf{v2}))\circ(\mathsf{v3}\circ\mathsf{v4}))\vdash s & [comm]
\end{array}
$$

Figure 1: A derivation for '`s node node v1 v2 v3 v4`' corresponding to the minimum node cover $\{v_1, v_4\}$.

$$
\dfrac{
\dfrac{\text{s} \vdash s \qquad \dfrac{\dfrac{\text{node} \vdash v_2 \quad \text{v1} \vdash v_2\backslash v_2}{\text{node} \circ \text{v1} \vdash v_2}[\backslash E] \quad \text{v4} \vdash v_2\backslash(s\backslash s)}{(\text{node} \circ \text{v1}) \circ \text{v4} \vdash s\backslash s}[\backslash E]}{\text{s} \circ ((\text{node} \circ \text{v1}) \circ \text{v4}) \vdash s}[\backslash E]
\qquad
\dfrac{\dfrac{\text{node} \vdash v_3 \quad \text{v2} \vdash v_3\backslash v_3}{\text{node} \circ \text{v2} \vdash v_3}[\backslash E] \quad \text{v3} \vdash v_3\backslash(s\backslash s)}{(\text{node} \circ \text{v2}) \circ \text{v3} \vdash s\backslash s}[\backslash E]
}{(\text{s} \circ ((\text{node} \circ \text{v1}) \circ \text{v4})) \circ ((\text{node} \circ \text{v2}) \circ \text{v3}) \vdash s}[\backslash E]
$$

$$
\begin{array}{ll}
(\text{s} \circ ((\text{node} \circ \text{v1}) \circ \text{v4})) \circ ((\text{node} \circ \text{v2}) \circ \text{v3}) \vdash s & \\
\hline
(\text{s} \circ ((\text{node} \circ \text{v1}) \circ \text{v4})) \circ (\text{node} \circ (\text{v2} \circ \text{v3})) \vdash s & [ass] \\
\hline
(\text{s} \circ (\text{node} \circ (\text{v1} \circ \text{v4}))) \circ (\text{node} \circ (\text{v2} \circ \text{v3})) \vdash s & [ass] \\
\hline
((\text{s} \circ \text{node}) \circ (\text{v1} \circ \text{v4})) \circ (\text{node} \circ (\text{v2} \circ \text{v3})) \vdash s & [ass] \\
\hline
(\text{s} \circ \text{node}) \circ ((\text{v1} \circ \text{v4}) \circ (\text{node} \circ (\text{v2} \circ \text{v3}))) \vdash s & [ass] \\
\hline
(\text{s} \circ \text{node}) \circ (((\text{v1} \circ \text{v4}) \circ \text{node}) \circ (\text{v2} \circ \text{v3})) \vdash s & [ass] \\
\hline
(\text{s} \circ \text{node}) \circ ((\text{node} \circ (\text{v1} \circ \text{v4})) \circ (\text{v2} \circ \text{v3})) \vdash s & [comm] \\
\hline
(\text{s} \circ \text{node}) \circ (\text{node} \circ ((\text{v1} \circ \text{v4}) \circ (\text{v2} \circ \text{v3}))) \vdash s & [ass] \\
\hline
(\text{s} \circ \text{node}) \circ (\text{node} \circ (\text{v1} \circ (\text{v4} \circ (\text{v2} \circ \text{v3})))) \vdash s & [ass] \\
\hline
(\text{s} \circ \text{node}) \circ (\text{node} \circ (\text{v1} \circ ((\text{v2} \circ \text{v3}) \circ \text{v4}))) \vdash s & [comm]
\end{array}
$$

Figure 2: A derivation for '`s node node v1 v2 v3 v4`' corresponding to the minimum node cover $\{v_2, v_3\}$.

References

Garey, Michael R. and David S. Johnson, editors. 1979. *Computers and Intractability. A Guide to the Theory of NP-completeness.* Freeman, New York.

Kanovich, Max I. 1991. The multiplicative fragment of linear logic is NP-complete. ITLI Prepublication Series X-91-13, University of Amsterdam.

Kanovich, Max I. 1992. Horn programming in linear logic is NP-complete. In *Proceedings, Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 200–210. IEEE Computer Society Press, 22-25 June.

Moot, Richard and Mario Piazza. 2001. Linguistic applications of first order multiplicative linear logic. *Journal of Logic, Language and Information*, 10(2):211–232.