Soft Linear Logic and the Polynomial Hierarchy (Extended Abstract)

Brian F. Redmond

Department of Computing, Mathematics and Statistical Sciences
Grande Prairie Regional College
10726 - 106 Avenue, Grande Prairie, AB
Canada T8V 4C4
E-mail: bredmond@GPRC.ab.ca

Abstract

In this paper we argue that Lafont's system of Soft Linear Logic [3] is expressive enough to characterize any level of the Polynomial Hierarchy. This characterization is obtained using the existing additive connectives and does not require the introduction of new rules to SLL.

1 Introduction

In [4], Mairson and Terui showed that the additives are superfluous for the characterization of polynomial time algorithms in Lafont's Soft Linear Logic (SLL) [3]. Nevertheless, provided only external (lazy) reduction of proof nets is used, the additives do not violate the polynomial bound on reduction and, we argue, may be used, *without modification*, to express nondeterminism in SLL. The idea is to define a new data type $\mathbf{T}$ of "lazy" trees with additive branching and boolean leaves using second-order quantification and to investigate the Kleisli category corresponding to this monad. Since generic maps for the sequent $\mathbf{S}^{(n)} \vdash \mathbf{B}$ represent polynomial time algorithms (see [3]), we argue in this paper that generic maps for the sequent $\mathbf{S}^{(n)} \vdash \mathbf{T}(\mathbf{B})$ should represent nondeterministic polynomial time algorithms. Unfortunately, this idea does not work out quite so cleanly (without adding new rules explicitly) as data types in SLL do not, in general, form monads. Nevertheless, with a slightly less elegant encoding, this idea can be used to obtain a characterization of NP in SLL which has an easy generalization to any level of the Polynomial Hierarchy. This characterization is novel in that it encodes a polynomial time verifier directly in SLL, and does not require a nondeterministic cut-elimination procedure and/or modified rules for the additives (cf. [5, 6, 1, 2]). Due to space limitations, only the main ideas behind the characterizations are given here. The full details will be presented in future work.

2 Soft Linear Logic

In this section we briefly recall Lafont's Soft Linear Logic [3]. Formulas are given by the following grammar:

$$A \quad ::= \quad \alpha \mid A \,\&\, A \mid \mathbf{1} \mid A \otimes A \mid A \multimap A \mid !A \mid \forall \alpha.A$$

where α is an atomic type variable. Sequents are written intuitionistically in the form $\Gamma \vdash A$ where Γ is a finite (possibly empty) list of formulas and A is a single formula. The rules are as follows:

- Identity, exchange and cut:

$$\frac{}{A \vdash A}\ (id) \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, B, A \vdash C}\ (ex) \qquad \frac{\Gamma \vdash A \quad \Delta, A \vdash C}{\Gamma, \Delta \vdash C}\ (cut)$$

- Multiplicatives:

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C} \qquad \frac{}{\vdash \mathbf{1}} \qquad \frac{\Gamma \vdash C}{\Gamma, \mathbf{1} \vdash C}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \qquad \frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C}$$

- Additives:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B} \qquad \frac{\Gamma, A \vdash C}{\Gamma, A \,\&\, B \vdash C} \qquad \frac{\Gamma, B \vdash C}{\Gamma, A \,\&\, B \vdash C}$$

- Soft exponential rules *soft promotion* and *multiplexing* (of rank $n \geq 0$):

$$\frac{\Gamma \vdash A}{!\Gamma \vdash !A} \qquad \frac{\Gamma, A^{(n)} \vdash C}{\Gamma, !A \vdash C}$$

- Quantification rules:

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall \alpha.A}\ \alpha \notin FTV(\Gamma) \qquad \frac{\Gamma, A[B/\alpha] \vdash C}{\Gamma, \forall \alpha.A \vdash C}$$

Recall that for any polynomial expression P and formula A, the expression $A\langle P \rangle$ denotes formula A with each subformula of the form $!B$ replaced by B^P (see [3] for details).

Proofs in this paper will be presented using the sequent calculus formulation above, but will be implicitly identified with their corresponding proof nets as given in [3]. Also, nets will be reduced using *external reduction* only. The main result of Lafont's paper is that *generic*[1] proofs for the sequent $\mathbf{S}^{(n)} \vdash \mathbf{B}$ correspond to polynomial time algorithms and vice versa.

[1] A generic proof is one without multiplexing.

3 Lazy Trees and Nondeterminism

Recall the inductive data types in SLL, including the type of booleans $\mathbf{B} = \forall\alpha.(\alpha \,\&\, \alpha) \multimap \alpha$ and the type of boolean strings $\mathbf{S} = \forall\alpha.!((\alpha \multimap \alpha) \,\&\, (\alpha \multimap \alpha)) \multimap \alpha \multimap \alpha$. We add to this the data type of lazy trees with parameter types U and V:

$$\mathbf{T}_{U,V} = \forall\alpha.!(\alpha \otimes U \,\&\, \alpha \otimes U \multimap \alpha) \multimap (V \multimap \alpha) \multimap \alpha$$

If $U = \mathbf{M} \multimap \mathbf{M}$ and $V = \mathbf{M}$, where $\mathbf{M}$ is the type representing Turing machines (see [3]), we claim that generic proofs for the sequent $\mathbf{S}^{(n)} \vdash \mathbf{T}\langle P\rangle$ correspond to nondeterministic polynomial time computations, and vice versa.[2]

Recall that each SLL proof can be interpreted in second-order intuitionistic linear logic as SLL is a subsystem of ILL. Therefore, such a tree can be evaluated (in ILL) to true iff one its branches terminates in an accepting state. This leads to the following definition.

Definition 3.1 *We say that a generic proof π for the sequent $\mathbf{S}^{(n)} \vdash \mathbf{T}\langle P\rangle$* accepts *a boolean string of type $\mathbf{S}$ if the corresponding tree $\mathbf{T}\langle P\rangle$ evaluates (in the above sense) to the net representing true. The set of all boolean strings that are accepted by π, denoted $L(\pi)$, is called the* language corresponding to π.

We have the theorem:

Theorem 3.2 *Let π be a generic proof for the sequent $\mathbf{S}^{(n)} \vdash \mathbf{T}\langle P\rangle$. Then $L(\pi) \in NP$.*

Proof: For each input of type $\mathbf{S}$ that is accepted by π, there is a generic proof for the sequent $\vdash \mathbf{T}\langle P\rangle \multimap \mathbf{B}$ that encodes a path to a leaf of the tree that evaluates to true (otherwise, all branches evaluate to false). Indeed, let $U = \mathbf{M} \multimap \mathbf{M}$ and $V = \mathbf{M}$, and recall that there is a generic proof out for the sequent $\mathbf{M} \vdash \mathbf{B}$ which says if the machine is in an accepting state. If a tree of type $\mathbf{T}$ has rank n, we instantiate α at $\mathbf{M}^{\&n} := \mathbf{M} \,\&\, \cdots \,\&\, \mathbf{M}$ ($\max(n,1)$ times) and let γ denote a generic proof for the sequent $\mathbf{M}^{\&n} \otimes U \,\&\, \mathbf{M}^{\&n} \otimes U \vdash \mathbf{M}^{\&n}$ which encodes a particular path to a leaf of the tree and evaluates the current configuration with a net of type U:

$$\dfrac{\dfrac{\dfrac{\dfrac{\gamma}{\mathbf{M}^{\&n} \otimes U \,\&\, \mathbf{M}^{\&n} \otimes U \vdash \mathbf{M}^{\&n}}}{\vdash \mathbf{M}^{\&n} \otimes U \,\&\, \mathbf{M}^{\&n} \otimes U \multimap \mathbf{M}^{\&n}}}{\vdash\, !(\mathbf{M}^{\&n} \otimes U \,\&\, \mathbf{M}^{\&n} \otimes U \multimap \mathbf{M}^{\&n})} \qquad \dfrac{\dfrac{\dfrac{\mathsf{diag}}{V \vdash \mathbf{M}^{\&n}}}{\vdash V \multimap \mathbf{M}^{\&n}} \qquad \dfrac{\pi_1\mathsf{out}}{\mathbf{M}^{\&n} \vdash \mathbf{B}}}{(V \multimap \mathbf{M}^{\&n}) \multimap \mathbf{M}^{\&n} \vdash \mathbf{B}}}{!(\mathbf{M}^{\&n} \otimes U \,\&\, \mathbf{M}^{\&n} \otimes U \multimap \mathbf{M}^{\&n}) \multimap (V \multimap \mathbf{M}^{\&n}) \multimap \mathbf{M}^{\&n} \vdash \mathbf{B}}}{\mathbf{T} \vdash \mathbf{B}}$$

[2]The presence of the polynomial P in the conclusion is unfortunate; it is a consequence of the fact that constructors of data types in SLL cannot in general be expressed uniformly.

Note that this proof is generic, so there is an analogous generic proof for the sequent $\mathbf{T}\langle P\rangle \vdash \mathbf{B}$ for any polynomial P. Then evaluation:

$$\dfrac{\dfrac{\pi}{\mathbf{S}^{(n)} \vdash \mathbf{T}\langle P\rangle} \quad \dfrac{}{\mathbf{B}\vdash\mathbf{B}}}{\mathbf{S}^{(n)}, \mathbf{T}\langle P\rangle \multimap \mathbf{B} \vdash \mathbf{B}}$$

is a polynomial time verifier for the language. (Note that the time bound does not depend on the size of the "certificate".) □

Conversely, we have:

Theorem 3.3 *If a predicate is computable on boolean strings by a nondeterministic Turing machine in polynomial time $P(n)$ and in polynomial space $Q(n)$, there is a generic proof for the sequent $\mathbf{S}^{(\deg P+\deg Q+1)} \vdash \mathbf{T}\langle P\rangle$ which corresponds to this predicate.*

Proof: There is a generic proof init for the sequent $\mathbf{S}^{(\deg Q+1)} \vdash \mathbf{M}\langle Q\rangle$ that writes a string of size n on a tape of size $Q(n)$ and puts the machine in the starting state. The nondeterministic transition function is encoded by a pair of generic proofs, left and right, each of type $U = \mathbf{M}\langle Q\rangle \multimap \mathbf{M}\langle Q\rangle$. All together, we get the generic proof:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\alpha\vdash\alpha \quad \dfrac{\mathsf{left}}{\vdash U}}{\alpha\vdash\alpha\otimes U} \quad \dfrac{\alpha\vdash\alpha \quad \dfrac{\mathsf{right}}{\vdash U}}{\alpha\vdash\alpha\otimes U}}{\alpha\vdash\alpha\otimes U \,\&\, \alpha\otimes U} \quad \dfrac{}{\alpha\vdash\alpha}}{\alpha\otimes U \,\&\, \alpha\otimes U \multimap \alpha, \alpha\vdash\alpha}}{\alpha\otimes U \,\&\, \alpha\otimes U \multimap \alpha \vdash \alpha\multimap\alpha}}{(\alpha\otimes U \,\&\, \alpha\otimes U \multimap \alpha)^P \vdash (\alpha\multimap\alpha)^P} \quad \dfrac{\dfrac{\mathsf{init}}{\mathbf{S}^{(\deg Q+1)}\vdash V} \quad \dfrac{\alpha\vdash\alpha \quad \alpha\vdash\alpha}{\alpha\multimap\alpha,\alpha\vdash\alpha}}{\mathbf{S}^{(\deg Q+1)}, \alpha\multimap\alpha, V\multimap\alpha\vdash\alpha}}{\mathbf{S}^{(\deg Q+1)}, (\alpha\multimap\alpha)^P \multimap\alpha\multimap\alpha, (\alpha\otimes U \,\&\, \alpha\otimes U \multimap \alpha)^P, V\multimap\alpha\vdash\alpha}}{\mathbf{S}^{(\deg Q+1)}, \mathbf{N}\langle P\rangle, (\alpha\otimes U \,\&\, \alpha\otimes U \multimap \alpha)^P, V\multimap\alpha\vdash\alpha}}{\mathbf{S}^{(\deg P+\deg Q+1)}, (\alpha\otimes U \,\&\, \alpha\otimes U \multimap \alpha)^P, V\multimap\alpha\vdash\alpha}}{\mathbf{S}^{(\deg P+\deg Q+1)} \vdash (\alpha\otimes U \,\&\, \alpha\otimes U \multimap \alpha)^P \multimap (V\multimap\alpha)\multimap\alpha}}{\mathbf{S}^{(\deg P+\deg Q+1)} \vdash \mathbf{T}\langle P\rangle}$$

where $V = \mathbf{M}\langle Q\rangle$ and $U = \mathbf{M}\langle Q\rangle \multimap \mathbf{M}\langle Q\rangle$, which corresponds to the NP-predicate. □

4 The Polynomial Hierarchy

The results of the previous section generalize to any level of the Polynomial Hierarchy (PH) by first encoding a second lazy product type $A \times B$ which behaves in a similar fashion to the & rule. The sum type in SLL may be used

for this purpose. We may think of $\times$ as a *universal* product and & as an *existential* product. We then define a second type of lazy trees:

$$\mathbf{T}^{*}_{U,V} = \forall\alpha.!(\alpha \otimes U \times \alpha \otimes U \multimap \alpha) \multimap (V \multimap \alpha) \multimap \alpha$$

which evaluates to true (in ILL) iff all of its branches are in accepting states. More generally, we define a hierarchy of lazy tree types inductively for each $i \geq 1$ as follows:

$$\mathbf{T}^{\Sigma_1}_{U,V} = \mathbf{T}_{U,V} \qquad \mathbf{T}^{\Pi_1}_{U,V} = \mathbf{T}^{*}_{U,V} \qquad \mathbf{T}^{\Sigma_{i+1}}_{U,V} = \mathbf{T}_{U,\mathbf{T}^{\Pi_i}_{U,V}} \qquad \mathbf{T}^{\Pi_{i+1}}_{U,V} = \mathbf{T}^{*}_{U,\mathbf{T}^{\Sigma_i}_{U,V}}$$

along with the appropriate notions of evalution in ILL (i.e., the appropriate generalization of def. 3.1). This leads to the following theorem whose proof will be deferred to the full version of this paper.

Theorem 4.1 *For each $i \geq 1$, generic proofs in SLL for the sequent $\mathbf{S}^{(n)} \vdash \mathbf{T}^{\Sigma_i}\langle P\rangle$ (respectively, $\mathbf{S}^{(n)} \vdash \mathbf{T}^{\Pi_i}\langle P\rangle$) correspond to algorithms in Σ_iP (respectively, Π_iP), and vice versa.*

Acknowledgements: The author would to thank Robin Cockett at the University of Calgary for the initial suggestion that such a characterization should be possible based on abstract categorical considerations. Moreover, the author would like to thank the anonymous referees whose suggestions have been incorporated into the final version of this paper.

References

[1] M. Gaboardi, J.-Y. Marion & S. Ronchi Della Rocca, Soft Linear Logic and Polynomial Complexity Classes, Electronic Notes in Theoretical Computer Science 205(6), pages 67–87, 2008.

[2] M. Gaboardi, J.-Y. Marion & S. Ronchi Della Rocca, A logical account of PSPACE, Proceedings of the 35th ACM SIGPLAN-SIGACT symposium on Principles of programming languages (POPL'08), pages 121–131, 2008.a

[3] Yves Lafont, Soft Linear Logic and Polynomial Time, Theoretical Computer Science 318(1-2), pages 163–180, 2004.

[4] Harry G. Mairson, Kazushige Terui, On the Computational Complexity of Cut-Elimination in Linear Logic, ICTCS 2003: 23-36.

[5] Satoshi Matsuoka, Nondeterministic Linear Logic, IPSJ SIGNotes PROgramming No. 12, 1996.

[6] F. Maurel, Nondeterministic Light Logics and NP-Time, LNCS 2701, 241-255, 2003.

[7] B. Redmond, Bounded Combinatory Logic and Polynomial Complexity. Submitted to Information and Computation (special issue dedicated to ICC, 2013).