Nondeterministic Light Logics and NP-Time

François Maurel

Preuves, Programmes et Systèmes*
Francois.Maurel@pps.jussieu.fr

Abstract. This paper relates two distinct traditions: the one of complexity classes characterisations through light logics and models of nondeterminism. Light logics provide an implicit characterisation of P-Time algorithms through the Curry-Howard isomorphism: every derivation reduces to its normal form in polynomial time and every polynomial Turing machine can be simulated by a derivation. In this paper, we extend Intuitionistic Light Affine Logic, a logic with full weakening, with a simple rule for nondeterminism and get a completeness result for NP-Time algorithms which is, as far as we know, the first Curry-Howard characterisation of NP complexity. We conclude by a reformulation of the P $\neq$ NP conjecture.
Keywords. Light logics, implicit characterisations of complexity classes, NP complexity.

1 Introduction

There has been quite a huge work to characterise complexity classes through various machines [15] or programming languages [13,4,9]. Light logics have provided a new insight by giving Curry-Howard characterisations to P-Time. Girard introduced the first light logic in [7]. It is called Light Linear Logic. LLL proof nets represent P-Time in the sense that (1) every net in LLL can only perform polynomial cut-eliminations on its *size*, the polynomial bound depending on a graph theoretic value, its *depth*; (2) conversely, every polynomial Turing machine is encodable inside LLL. Light Affine Logic, defined by Asperti in [1] and its intuitionnistic variant ILAL lead to many simplifications. Some links between these traditions (lambda calculus, safe recursion and light logics) are studied by Roversi in [17] and Murawsky and Ong in [14].

In this paper, we first recall the syntax and reduction rules of ILAL. Then, we introduce nILAL, obtained by adding a Sum deduction rule to ILAL. This deduction rule is a logical counterpart to nondeterministic choice in process calculi. We show that nILAL represents NP in the same way that ILAL represents P. Finally, we conclude by reformulating the P $\neq$ NP conjecture in our setting.

1.1 NP-Time Problems

The notion of polynomial time problems can be defined in many models and, fortunately, these classes generally coincide. A deterministic Turing machine is

* CNRS & Université Paris VII, Case 7014, 2 place Jussieu, 75251 Paris cedex 05.

an abstract machine with finitely many states and an unbounded finite memory but with a bounded local view on it. A configuration of the machine is the data of the state and the memory. A transition, i.e. a step between two configurations, is uniquely determined by the current state of the machine and the local access to the memory. A deterministic Turing machine representing a function is in P-Time if there exists a polynomial bound (on the size of the input) for the number of transitions before it stops. Note that we consider functions or relations from integers to booleans, as is customary, but also to integers.

The notion of nondeterministic Turing machines is a generalisation of deterministic Turing machines where many transitions can occur: at each step, the machine makes a nondeterministic choice between these transitions. The notion of calculation is usually enriched by an *accepting* state: the result of a computation is taken into account if and only if it ends in the accepting state. We then say that the machine answers. A NP-Turing machine is a machine for which there is a polynomial bound on the size of its input for the number of transitions it may perform before it stops. There is an asymmetry between *true* and *false*: a class of inputs is said to be in NP if there exists a NP-Turing machine which may answer *true* on a given input if and only if this input is in the considered class; a class is said to be in coNP if there exists a NP-Turing machine which may answer *false* on a given input if and only if this input is not in the considered class. A problem is said to be in P if it is solvable by a P-Time Turing machine. A problem is said to be in NP if it is solvable by a NP-Turing machine *i.e.* a NP-Turing machine which may answer *true* if and only if the answer is *true*. Symmetrically, a problem is said to be in coNP if it is co-solvable by a NP-Turing machine i.e. a NP-Turing machine which may answer *false* if and only if the answer is *false*.

1.2 Intuitionistic Light Affine Logic

Light Affine Logic [1,2] has been introduced by Asperti in 1998 as a simplification of LLL. It leads to a similar logical characterisation of P-Time algorithms.

Syntax of LAL. To impose a polynomial cut elimination, the syntax of LAL is very much constrained. This requires a new connective: the *paragraph*. The syntax of formulas, ranged over by $A, B \ldots$, is $\alpha \mid \alpha^{\perp} \mid A \invamp A \mid A \otimes A \mid\ !A \mid\ ?A \mid \S A$. The negation $\perp$ is defined by de Morgan laws. Dual pairs of connectives or constants are : $(\otimes, \invamp)$, $(!, ?)$, $(\S, \S)$ and $(\forall, \exists)$. The linear implication $\multimap$ is defined by $A \multimap B = A^{\perp} \invamp B$.

The elimination of a cut between two weakened formulas (which can occur in LAL) is problematical; so, we simply restrict ourselves to the intuitionistic fragment of LAL (called ILAL), described in Fig. 1. Following [1,7,2], these rules may be explained by some remarks. In linear logic, the *of course* modality enables one to duplicate a proof net, whatever the number of formulas $?A$ in its context. In light logics, one needs to constrain the duplication in order to remain polytime. Hence, the *of course* rule admits only a single or empty context. Besides, the usual dereliction rule is replaced by the *paragraph* rule, which tags the conclusion with §.

$$\frac{}{A \vdash A}\ \mathrm{Ax} \qquad \frac{\Gamma \vdash A \qquad A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\ \mathrm{Cut}$$

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, B, A, \Delta \vdash C}\ \mathrm{Perm.}$$

$$\frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B}\ \mathrm{Contr.} \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ \mathrm{Weak.}$$

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\ \otimes_l \qquad \frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ \otimes_r$$

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

$$\frac{B \vdash A}{!B \vdash !A}\ \text{of course} \qquad \frac{\vdash A}{\vdash !A}\ \text{of course}$$

$$\frac{\Gamma, \Delta \vdash A}{!\Gamma, \S\Delta \vdash \S A}\ \text{paragraph}$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, \forall\alpha.A}\ \text{for all}\ (\alpha \notin FV(\Gamma)) \qquad \frac{\Gamma, A[B/\alpha] \vdash C}{\Gamma, \forall\alpha.A \vdash C}\ \text{there is}$$

Fig. 1. Sequent calculus presentation of ILAL

Proof Nets of ILAL. Proof nets in ILAL are just like standard intuitionistic proof nets (plus a §-box). Since [7], it is customary to consider a stratified cut-elimination strategy on them. The intuitive idea is to reduce at depth 0 then 1 and so on. Every link has a measure (called size in this paper) and the total measure does not increase too fast during reduction. The enables a simple enough proof of P-Time. However, a recent work [18] shows that alternative ILAL *non stratified* strategies cut-eliminate in P-Time.

Every proof net has a unique output (conclusion) but many inputs (hypothesis). For example, an axiom introduces a net with one input linked to one output and a cut links the output of a net with an input of another net. Links $\lambda, \otimes_r$ and contraction have two outputs and a single input, @ and $\otimes_l$ have two inputs and a single output, $\forall$ has one input and one output, and the weakening has only one input and no output. There are boxes for $\forall$ (on the right), and exponential rules (*of course* and *paragraph*). A complete description of ILAL proof nets can be found in [2]. This cannot be included here for space reasons but Fig. 2 gives the normalisation rules (where the diamond and box symbols can be either $\forall$, ! or § depending upon the types) and important points are recalled in Defini-

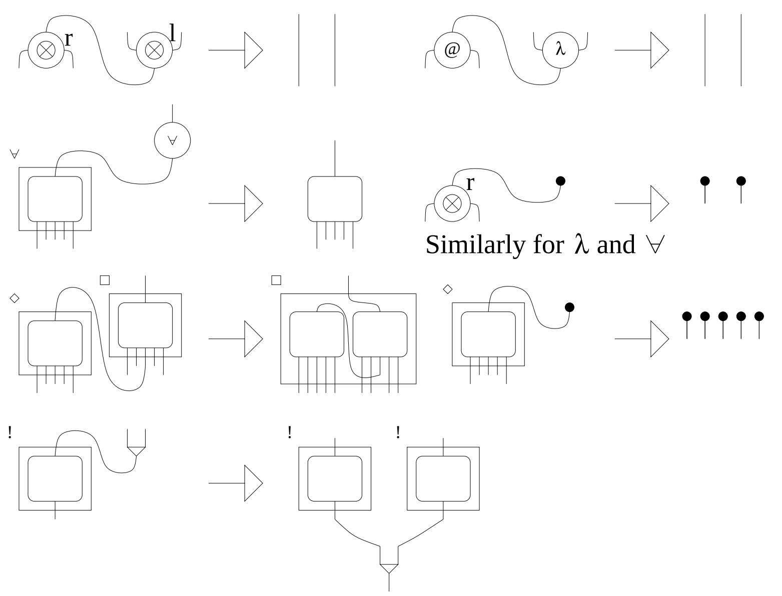

Fig. 2. Cut elimination for ILAL

tion 2.1. Examples of derivations can be found in Fig. 3. They represent both nonequivalent kinds of integers (unary and binary ones) and the iterator which only works on unary integers.

Definition 1.1 (Depth). *The* depth *of a net is the maximal nesting of exponential (!/§)-boxes.*

A crucial property in ILAL is that the depth of a net cannot increase during a reduction and boxes cannot be opened since there is no dereliction. In particular, a cut between exponential boxes (either ! or §) is eliminated by merging the boxes. From that, Asperti adapts Girard's proof in [7] and proves

Proposition 1.1 (ILAL is in P-Time). *For every $d \in \mathbb{N}$, there exist polynomials S_d and R_d such that for every net of size s and depth d, the maximal size of its reducts is bounded by $S_d(s)$ and the maximal number of steps necessary to reach the normal form is bounded by $R_d(s)$.*

Conversely, Asperti and Roversi [2] adapt Girard's encoding of P-Time Turing machines inside proof nets, and prove that

Proposition 1.2 (P-Time is in ILAL). *Any P-Time function from binary integers to binary integers or booleans can be represented in ILAL.*

A Peano int: SS0=$\lambda fx.f(f\ x)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\alpha \vdash \alpha} \quad \dfrac{\dfrac{}{\alpha \vdash \alpha} \quad \dfrac{}{\alpha \vdash \alpha}}{\alpha \multimap \alpha, \alpha \vdash \alpha}}{\alpha \multimap \alpha, \alpha \multimap \alpha, \alpha \vdash \alpha}}{\alpha \multimap \alpha, \alpha \multimap \alpha \vdash \alpha \multimap \alpha}}{!(\alpha \multimap \alpha), !(\alpha \multimap \alpha) \vdash \S(\alpha \multimap \alpha)}}{!(\alpha \multimap \alpha) \vdash \S(\alpha \multimap \alpha)}}{\vdash !(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)}}{\vdash \forall\alpha.!(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)}$$

A Binary int: $11_2 = \lambda s_1 s_0\ x.s_1(s_1\ x)$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\alpha \vdash \alpha} \quad \dfrac{\dfrac{}{\alpha \vdash \alpha} \quad \dfrac{}{\alpha \vdash \alpha}}{\alpha \multimap \alpha, \alpha \vdash \alpha}}{\alpha \multimap \alpha, \alpha \multimap \alpha, \alpha \vdash \alpha}}{\alpha \multimap \alpha, \alpha \multimap \alpha \vdash \alpha \multimap \alpha}}{!(\alpha \multimap \alpha), !(\alpha \multimap \alpha) \vdash \S(\alpha \multimap \alpha)}}{!(\alpha \multimap \alpha) \vdash \S(\alpha \multimap \alpha)}}{!(\alpha \multimap \alpha), !(\alpha \multimap \alpha) \vdash \S(\alpha \multimap \alpha)}}{\dfrac{\vdash !(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)}{\vdash \forall\alpha.!(\alpha \multimap \alpha) \multimap !(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)}}$$

The iterator on Peano integers: $\lambda nfx.nfx$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{!(A \multimap A) \vdash !(A \multimap A)} \quad \dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{A \vdash A}}{A \multimap A, A \vdash A}}{\S(A \multimap A), \S A \vdash \S A}}{!(A \multimap A) \multimap \S(A \multimap A), !(A \multimap A), \S A \vdash \S A}}{\forall\alpha.!(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha), !(A \multimap A), \S A \vdash \S A}}{\vdash (\forall\alpha.!(\alpha \multimap \alpha) \multimap \S(\alpha \multimap \alpha)) \multimap !(A \multimap A) \multimap \S A \multimap \S A}$$

Fig. 3. Integers and iterator

2 An Analysis of P-Time in ILAL

At this point, it is worth recalling the main steps leading to the proof of P-Time completeness for ILAL, as formulated in Propositions 1.1 and 1.2. These steps are mainly guidelines and we refer the reader to [2] for an explicit proof. We will see in Sect. 3 that the proof of NP-Time completeness for nILAL follows essentially the same steps.

2.1 Polynomial Bounds: ILAL in P-Time (Proposition 1.1)

Step 2.1 (Reduction Rules) *In ILAL, the rewriting rules over proof structures are of the form $\overline{R} \mapsto \overline{S}$ where:*

- ***Connectedness.*** *$\overline{R}$ is a connected proof structure extended with (meta) variables for proof structures, proof nets, formulas and links.*
- ***Locality.*** *All links in $\overline{R}$ are in the same depth (depth is defined in Definition 2.1).*

Remark 2.1. The locality property is not true for LLL because the reduction of the visible cut in Fig. 4 cannot be performed before some reduction steps occur inside the !-box. These "internal" reduction steps are necessary to remove every auxiliary door but one of the !-box (this is mandatory to ensure a polynomial

The reduction is not local as seen in this net: if the left auxiliary door is removed (by reducing the weakening) then the contraction disappears and the box may be duplicated.

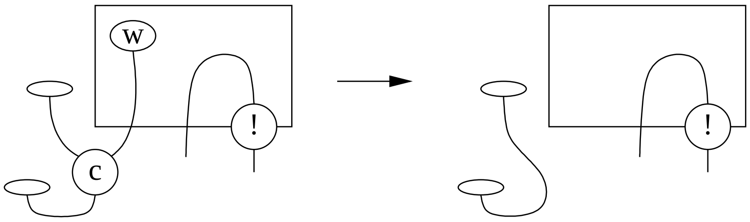

Fig. 4. Non locality of LLL

reduction). We will see next section that locality enables to extend the reduction strategies to nondeterministic SUM boxes; as for connectedness, that it enables us to *select* sub-nets and compute inside them.

Definition 2.1 (Boxes, depth and size). *The* depth *of a link in a ILAL net is the number of the exponential boxes which contain this link. The* size *of a net is the sum of the sizes of its links. The* partial size s_i *of a net is the sum of the sizes of the links at depth* i.

Step 2.2 (Size) *The size and partial sizes of a proof net are polynomially equivalent to the actual size and partial sizes of the proof net as a graph.*

Step 2.3 (Cuts) *There are two kinds of cuts: basic ones and exponential ones. At depth* d*, the reduction of:*

- *a basic cut only creates cuts at depth* d *and reduces the partial size* s_d*,*
- *an exponential cut creates only redexes at depths at least* $d + 1$.

Step 2.4 (Normalisation) *A complete ILAL normalisation strategy of a net of depth* d *is:*

- *in a preliminary round, reduce basic cuts at depth 0,*
- *for* $i < d$*, in round* i*, reduce exponential cuts at depth* i *and then basic ones at depth* $i + 1$.

Remark 2.2. The standard definitions of LLL or ILAL are more liberal, since they enable to interleave reduction of exponential cuts at depth i and of basic ones at depth $i + 1$; whereas here, we impose that basic cuts are reduced after exponential ones. This is possible in the case of ILAL, thanks to locality. This stricter strategy is introduced here because it is needed for confluence in the case of nondeterminism.

	depth 0	*depth* 1	*depth* 2	$\dots$	*depth* d
preliminary	s_0	s_1	s_2	$\dots$	s_d
round 0	s_0	$(s_0)s_1$	$(s_0)s_2$	$\dots$	$(s_0)s_d$
round 1	s_0	$(s_0)s_1$	$(s_0s_1)(s_0)s_2$	$\dots$	$(s_0s_1)(s_0)s_d$
round 2	s_0	$(s_0)s_1$	$(s_0s_1)(s_0)s_2$	$\dots$	$(s_0s_1s_0s_2)(s_0s_1)(s_0)s_d$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
round $d-1$	s_0	s_0s_1	$s_0^2s_1s_2$	$\dots$	$s_0^{2^d}\dots s_{d-1}s_d$

Fig. 5. Partial sizes

Step 2.5 (Partial sizes and normalisation) *If $s_i^{(k)}$ is the maximal partial size at depth i during round k and $S_i^{(k)}$ the maximal partial size during first part of round k, we have $\forall k \leq k'$, $S_{k'}^{(k+1)} \leq s_{k'}^{(k+1)} \leq s_k^{(k)}.s_{k'}^{(k)}$. Hence partial sizes are bounded by the values in Fig. 5.*

Proposition 2.1 (Time). *There are polynomials R_d and S_d such that for every proof net of size s and depth d, the maximal number of steps before reaching the normal form is bounded by $R_d(s)$ and the size of its reducts is bounded by $S_d(s)$.*

Proof. The preliminary round takes at most s_0 step since basic cuts reduce partial size. The exponential sub-round i takes at most $(s_0 \dots s_i) \dots (s_0s_1)(s_0)s_i$ steps it reduces the number of cuts. The basic sub-round i is similar to the preliminary round. □

Step 2.6 (Polystep/polytime) *Every step is polynomially bounded by the size of the net.*

Therefore, we remark that the reduction is polytime and not only polystep.

2.2 Completeness: P-Time in ILAL (Proposition 1.2)

Step 2.7 (Iterator) *A net $iter(n, f, input)$ where f is a net of type $A \multimap A$, n a Peano integer, and input an arbitrary input of the type A, reduces to $P(f \circ \dots \circ f(input))$ where P is some context independent from f, n or input.*

Step 2.8 (Encoding) *Turing machines of type $bint \to \tau$ are encoded (and simulated) in some types that are in a set $\Theta_{TM} = \{!bint \multimap \S^k\tau\}$. When given the translation n of an integer, the encoding of a Turing machine is equivalent to a net $C(iter(pol(length(n)), T, config(n)))$ where:*

- *C is a context from configurations to a representation of τ (there might be some* paragraphs *which come from the polynomial pol),*
- *iter is the iterator,*
- *length is the function from bint to int equivalent to $\lambda nfx.nffx$,*
- *pol is a polynomial, i.e. a function from int to int,*
- *config is a function from binary integers to configurations,*
- *T is the encoding of a deterministic transition.*

3 Adding Nondeterminism

Remark 3.1. In the sequel of this paper, the representation of an output type needs to take into account the notion of accepting state. We represent an output type τ (which needs not be iterable), where $\tau = bint$ or $\tau = bool$, by $bool * \tau$ where the first element is *true* if the computation ends in the accepting state.

3.1 The Connective $\oplus$ and the Sum Box

Although ILAL has no additive connectives, the connective $\oplus$ led to our first intuition on nondeterminism. The usual understanding of polarities (in games or ludics [12,8]) expresses that, during an interaction, a positive connective commits the first move and a negative one answers. The duality between the connectives $\oplus$ and & can be explained as: the connective $\oplus$ chooses what to do, it always play either $\oplus_1$ or $\oplus_2$ and the connective & can answer to any of $\oplus_1$ and $\oplus_2$. Hence interaction is well determined between these connectives. A personal discussion with Paul-André Melliès led the author to the study of a strange connective interpreted in terms of nondeterminism: the connective $\oplus_3$ is positive, but chooses nondeterministically between $\oplus_1$ and $\oplus_2$. So, we add a new rule $\oplus_3$ for $\oplus$ which is the same as the rule for & and get three rules for this connective:

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}\oplus_1 \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\oplus_2 \qquad \frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\oplus_3$$

Cut-elimination has to be performed in the following proof π:

$$\frac{\dfrac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\oplus_3 \qquad \dfrac{\vdash \Delta, A^\perp \qquad \vdash \Delta, B^\perp}{\vdash \Delta, A^\perp \& B^\perp}\&}{\vdash \Gamma, \Delta}\mathit{cut}$$

Hence, since we are seeking for a confluent system, we need a rule for choice. We introduce a Sum rule, inspired by what Ehrhard and Regnier use in [5] for algebraic purposes:

$$\frac{\vdash \Gamma \qquad \dots \qquad \vdash \Gamma}{\vdash \Gamma}\textsc{Sum}$$

The reduct of π is then:

$$\frac{\dfrac{\vdash \Gamma, A \qquad \vdash \Delta, A^\perp}{\vdash \Gamma, \Delta}\mathit{cut} \qquad \dfrac{\vdash \Gamma, B \qquad \vdash \Delta, B^\perp}{\vdash \Gamma, \Delta}\mathit{cut}}{\vdash \Gamma, \Delta}\textsc{Sum}$$

The rule $\oplus_3$ is then definable with the Sum rule.

Definition 3.1 (The logic nILAL). *The logic nILAL is formed by the rules of ILAL and a* Sum *rule:*

$$\frac{\Gamma \vdash A \qquad \dots \qquad \Gamma \vdash A}{\Gamma \vdash A}\textsc{Sum}$$

Reduction rules for nILAL are given in Fig. 6 and 7. The strategy is described in Definition 3.4. One may remark that the DOWN rule is not local. This is not surprising since it changes the complexity (if $\mathrm{P} \neq \mathrm{NP}$).

Remark 3.2. We may proceed in the same way with a variant of LLL called Elementary Light Logic. In that case, however, SUM boxes are (intuitively) encodable inside the logic. Indeed, ELL corresponds to elementary time and elementary time is closed under exponential. It is easy to deduce from the sequel of the paper, that this encoding cannot be considered in ILAL for complexity reasons: it would impose an exponential cut elimination.

3.2 Proof Nets

Proof structures for nILAL are proof structures for ILAL plus SUM boxes of conclusion Γ with *one or more* sub-proof structures of conclusions Γ. The proof net associated to a proof ending by a SUM rule is the SUM box of the proof nets associated to every immediate sub-proofs of the considered proof.

Definition 3.2 (Summand). *In a given net, a* summand *is a proof structure obtained by erasing all but one sub-net in every* SUM *box.*

Proposition 3.1 (Correctness). *A nILAL proof structure is a proof net (i.e. is sequentialisable) if and only if all its summands are ILAL proof nets.*

Proof. By induction on the structure of proofs or proof nets. □

Definition 3.3 (Depth). *The* depth *of a net is the maximal depth (in the sense of ILAL) of its summands.*

3.3 Normalisation

In order to define a P-Time normalisation strategy, cut elimination for ILAL is stratified (i.e. executed in several rounds). Similarly, the strategy that we define for nILAL is stratified. This is necessary for the representation theorem: one shouldn't decide a nondeterministic choice too early. The first rules for nILAL are described in Fig. 6.

Definition 3.4 (Normalisation). *We define a normalisation strategy for a proof net R of depth d:*

- *Preliminary. In a first preliminary round, eliminate every sub-*SUM *boxes of* SUM *boxes using the* MERGE *rule. One can suppose that the whole net is included in a* SUM *box*[1]*. This outer* SUM *box is called* principal*. Its immediate sub-nets are also called* principal.

The rest of the strategy is also described by rounds for nILAL but here, rounds only apply to principal nets: during a normalisation, one may encounter different rounds for different principal nets.

[1] By eventually plugging R in a singleton SUM box: SUM(R).

The MERGE **rule (associativity)**:

A net $\textsc{Sum}(R_1, \ldots, \textsc{Sum}(S_1, \ldots, S_n), \ldots, R_k)$ may be reduced into the net $\textsc{Sum}(R_1, \ldots, S_1, \ldots, S_n, \ldots, R_k)$.

The DOWN **rule (linearity)**:

A net of the form $\textsc{Sum}(\ldots, C[\textsc{Sum}(S_1, \ldots, S_k)], \ldots)$, where C is an arbitrary context, may be rewritten into $\textsc{Sum}(\ldots, \textsc{Sum}(C[S_1], \ldots, C[S_k]), \ldots)$.

The ILAL rules:

Reduce basic and exponential cuts as in ILAL.

Fig. 6. Cut elimination rules

- *In a second preliminary round, apply* DOWN *rule,* MERGE *rule and basic reduction at depth* 0*.*
- *For $i < d$, in round i, eliminate exponential cuts at depth i and then apply* DOWN *rule,* MERGE *rule and basic reduction at depth $i + 1$.*

Remark 3.3. SUM boxes rise to the top and gather in a single SUM box. One may also check that SUM boxes disappear: after round $i > 0$, there is no SUM box at depth i. This rule involves a lot of duplications. We add *selection* in Sect. 3.5 to avoid exponential complexity.

Let ψ be the function which takes a net and returns the set (and not the multi-set) of its principal sub-nets. A strategy is said to be confluent up to multiplicities when the strategy is confluent after applying ψ.

Proposition 3.2 (Confluence). *The normalisation strategy in nILAL is confluent up to multiplicities.*

Remark 3.4. The notion of confluence has to be relaxed because of the case of a net with a weakened ∀-box that contain a SUM box. There is a reduction path where the SUM box duplicates the context before being erased and another path where the SUM box is erased when the ∀-box is erased. Hence, the number of summands is not the same in both cases but the sets of (normal) summands are the same. This stratified strategy is also needed because of nondeterminism: a lax strategy where one can reduce SUM boxes before exponential ones is not confluent because one would do identical choices in a given box, and this is too uniform. If one needs a strictly confluent strategy, one may require that SUM boxes are reduced before basic cuts. A better solution is to slightly change the reduction rules of ILAL: the reduction of a weakened ∀-box would be the opening of the box with a weakening on top. The complete erasing of the box would wait for the reduction of inner SUM boxes.

3.4 An Encoding of NP-Turing Machines

Our encodings are still denoted by $[\![\ ]\!]$. To encode a nondeterministic transition, we simply use a $\textsc{Sum}$ box of deterministic transitions, see [7,16,2]. To encode the Turing machine, we iterate this nondeterministic transition on an input with a polynomial bound (depending on the input) on steps. This net, when fed with an input (the encoding of the initial configuration of the machine), reduces to a $\textsc{Sum}$ box of every possible results.

Lemma 3.1 (Iteration of Sum boxes).
A net $Iter(n, \textsc{Sum}((f_i)_{i\in[1..k]}), input)$ where the f_i are deterministic nets of type $A \multimap A$ reduces to $\textsc{Sum}(P(f_{i_1} \circ f_{i_2} \circ \ldots \circ f_{i_n}(input))_{i_1,\ldots,i_n\in[1..k]})$ where P is the same context as for Step 2.7.

Definition 3.5 (Representation). *A nondeterministic Turing machine f is* representable *in nILAL if and only if the net obtained by linking its representation in nILAL and the representation of an arbitrary input x reduces to a net which has a summand representing in the sense of ILAL any possible $f(x)$.*

Proposition 3.3 (Completeness). *Any NP-Time function from binary integers to binary integers or booleans can be represented in nILAL.*

The obvious reduction algorithm derived from this representation is exponential. However, next section proves that reduction can be done polynomially.

3.5 Time

When reducing a cut with a $\textsc{Sum}$ box, one needs to be careful in order to avoid useless duplications. One can remark that two principal sub-nets cannot communicate, hence since we are only interested in summands, if we seek for a future summand of a given principal sub-net, we only need to reduce this sub-net. This given sub-net is called selected. In any computation, one may only have one selected sub-net, the one leading to the desired summand.

Definition 3.6 (Selected sub-net). *In a reduction, one may consider a particular principal sub-net and perform the next reduction step if possible inside this sub-net. This sub-net is said to be* selected.

For the simplicity of exposition, we change the DOWN rule into the DOWN' one. When applying the DOWN and MERGE rules,

$$\begin{aligned} &\textsc{Sum}(\ldots, C[\textsc{Sum}_i(S_i)], \ldots) \\ \longmapsto\ &\textsc{Sum}(\ldots, \textsc{Sum}_i(C[S_i]), \ldots) \\ \longmapsto\ &\textsc{Sum}(\ldots, C[S_1], \ldots, C[S_k], \ldots) \end{aligned}$$

one may exponentially increase the amount of duplications. To avoid this, we use selection: we write the net as a $\textsc{Sum}$ box with at most two principal sub-nets: the selected one and a $\textsc{Sum}$ box with other former sub-nets. We add a new rule in Fig. 7, the DOWN' rule, which preserve the property of having only two principal

The DOWN' **rule**:

Every net of the form $\textsc{Sum}(C[\textsc{Sum}_i(S_i)], D)$ can be rewritten into $\textsc{Sum}(C[S_{i_0}], \textsc{Sum}(D, C[\textsc{Sum}_{i \neq i_0}(S_i)]))$ where i_0 corresponds to the newly selected sub-net.

Fig. 7. The DOWN' rule

sub-nets. The DOWN' is more complex than the DOWN rule but its reduction is cheaper since it only imposes one duplication of the context C. However, we could have kept the DOWN rule but proofs would have been heavier. We adapt our normalising strategy described in Definition 3.4 by replacing DOWN rules by DOWN' rules. This is safe since DOWN and DOWN' lead to the same normal forms.

Definition 3.7 (Selective reduction). *A reduction is said to be* selective *if at every step, there is a selected principal sub-net, and at every non initial step, the selected principal sub-net is a reduct of the previous one.*

Remark 3.5. The MERGE rule does not cost much: one has at most one MERGE rule after every DOWN' rules. Therefore, in the sequel, we may suppose that no MERGE rule can be applied in the selected principal net whenever needed.

Proposition 3.4 (Selective reduction). *For every net R and any summand of its normal form S, there is a selective reduction to a net which has S as a summand.*

Proof. The connectedness hypothesis and the definition of the normalisation strategy enable one to reduce principal sub-nets independently. □

Definition 3.8 (Sizes). *The* size *of a net is the maximal size its summands and the* SUM size *of a net is the sum of the sizes of its links, where the size of a* SUM *box is* 1*.*

Definition 3.9 (Sum reduction step). *We call* SUM reduction steps *the ones using* MERGE *or* DOWN' *rules. Other reduction steps are called* non-SUM reduction steps.

Proposition 3.5 (Time). *As in ILAL, for a net of size s and depth d, for every summand S of its normal form, the maximal number of non-*SUM *reduction steps necessary to reach a net which has S as a summand is bounded by $R_d(s)$.*

Proof. Consider a selective reduction leading to S and use Step 2.4 (normalisation). Check that Step 2.5 (partial sizes) is still valid. The sequel of the proof is similar of the one of Proposition 2.1. □

Now that non-SUM reduction steps are controlled, it is time to go on with SUM ones. They are as well controlled. Since it is valid, we suppose that no MERGE rule is applicable in the considered selected net.

Proposition 3.6 (Sum size). *During a selective reduction, the* SUM *size of the selected principal sub-net is bounded by* $s.S_d(s)$.

Proof. One can prove a result analogous to step 2.5 which establishes that each partial SUM size is bounded by the initial SUM size of the net times the partial size given in step 2.5. □

The DOWN' rules strictly lower the maximal nesting number of SUM boxes in the selected principal sub-net which is bounded by $S_d(s)$. Hence their applications are bounded by $S_d(s)$.

Proposition 3.7 (Maximal Sum size). *For every* $d \in \mathbb{N}$*, there exists a polynomial* M_d *such that for every selective reduction of a net* R *of* SUM *size* s *and depth* d*, the maximal* SUM *size of its reducts is bounded by* $M_d(s)$.

Proof. The non selected part of the net is a comb tree. Hence its SUM size is bounded by $s.S_d(s)+1$ times the number of SUM reductions plus 1. Therefore, we require $M_d(s) \geq (s.S_d(s)+1).(R_d(s')+1)$ where s' is the size of R. Since s' is smaller than s, we take $M_d(X) = (X.S_d(X)+1)(R_d(X)+1)$. □

Proposition 3.8 (Time). *For every* $d \in \mathbb{N}$*, there exists a polynomial* R_d *such that for every net* R *of size* s *and depth* d*, for every summand* S *of the normal form of* R*, the maximal number of cut reductions to reach a net which has* S *as a summand is bounded by* $R_d(s)$.

Proof. For a net of size s_0 and depth d, for every summand S of its normal form, the maximal number of non SUM reduction steps necessary to reach a net which has S as a summand is bounded by $R_d(s_0)$. We only need to prove that there are not too many SUM reduction steps and that they don't involve too many duplications. We need not be too careful, we can bound the number of DOWN' reductions by $S_d(s_0)$ and every duplication costs less than $M_d(s_0)$. So, the work to reduce the net is bounded by $((R_d+S_d)M_d)(s_0)$ and hence it is polynomial. □

Proposition 3.9 (Correctness). *For every net* $R : !bint \multimap \S^k\tau$ *there is a polynomial* P *such that for every input* x *of size* s *and depth* 1*, for every summand* S *of the normal form of the net* $f(x)$*, the maximal number of cut reductions to reach a net which has* S *as a summand is bounded by* $P(s)$.

Proof. Take an input of R of type *bint* and size s. Since a binary integer does not increase depth, the reduction toward a summand of $f(x)$ is polynomial in $s+s_0$, and hence polynomial in s. □

As for ILAL, we remark that the reduction is, here nondeterministically, polytime and not only polystep.

4 Reformulation of the P $\neq$ NP Conjecture

In this section, we reformulate the P $\neq$ NP conjecture. For simplicity, we only consider boolean problems. The following results are directly justified by the fact that we have a complete model of NP. However, we believe that they might have an interest thanks to the model.

If R is a net and b a boolean value, we write $R \leadsto b$ to express that the normal form of R has a summand representing b (in the accepting state 0).

Definition 4.1 (The relations $\cong_{\mathbf{NP}}$, $\cong_{\mathbf{coNP}}$ and ${}_{\mathbf{NP}}\leftrightharpoons_{\mathbf{coNP}}$). *We define three relations on nets of conclusions in Θ_{TM}:*

$R \cong_{NP} S \iff (\forall n \in \mathbb{N}, R([\![n]\!]) \leadsto true \Leftrightarrow S([\![n]\!]) \leadsto true)$

$R \cong_{coNP} S \iff (\forall n \in \mathbb{N}, R([\![n]\!]) \leadsto false \Leftrightarrow S([\![n]\!]) \leadsto false)$

$R \ {}_{NP}\leftrightharpoons_{coNP} S \iff (\forall n \in \mathbb{N}, R([\![n]\!]) \leadsto true \Leftrightarrow S([\![n]\!]) \not\leadsto false)$

Proposition 4.1 (Correctness/completeness). *Some natural results hold:*

1. *For every NP Turing machines M and N, M and N are NP-equivalent if and only if $[\![M]\!] \cong_{NP} [\![N]\!]$ and similarly for coNP.*
2. *For every proof net R of type in Θ_{TM}, there is a Turing machine M such that $[\![M]\!] \cong_{NP} R$ and similarly for coNP.*

Proof. This comes from the representation property and Proposition 3.9. □

Corollary 4.1 (Equivalence classes). *The translation $[\![\ ]\!]$ can be lifted in a bijection between NP and classes of $\cong_{NP}$ and between coNP and classes of $\cong_{coNP}$.*

Definition 4.2 (Uniformity and determinism). *A net is said to be* uniform *if $R \ {}_{NP}\leftrightharpoons_{coNP} R$. A net is said to be* deterministic *if R is uniform and every computation ends in the accepting state. The set $\mathfrak{Unif}$ is the set of uniform nets, and the set $\mathfrak{Det}$ is the set of deterministic nets.*

Recall from [15] that if NP = coNP, the whole polynomial hierarchy[2] collapses which implies P = NP. Therefore, P $\neq$ NP if and only if NP $\neq$ coNP.

Proposition 4.2 (Reformulation of the P $\neq$ NP conjecture). *We have:*

$P \neq NP \iff \exists R. \nexists S. R \ {}_{NP}\leftrightharpoons_{coNP} S \iff \exists S. \nexists R. R \ {}_{NP}\leftrightharpoons_{coNP} S$

$P \neq NP \iff \pi_1({}_{NP}\leftrightharpoons_{coNP}) \neq NP \iff \pi_2({}_{NP}\leftrightharpoons_{coNP}) \neq coNP$ *where π_i is the i-projection.*

We define three functions. The functions $/_{true}$ (*resp.* $/_{false}$) which takes a net of any type in Θ_{TM} and returns an NP-equivalent (*resp.* coNP-equivalent) net that never answers false (*resp.* true) and the function ϕ defined by:

$$\begin{array}{rcl} \phi: \quad \mathrm{NP} \cap \mathrm{coNP} & \longrightarrow & \mathfrak{Unif} \\ (M : \mathrm{NP}, N : \mathrm{coNP}) & \longmapsto & \textsc{Sum}(M/_{true}, N/_{false}) \end{array}$$

Remark that ϕ is a bijection up to equivalence of uniform nets.

Proposition 4.3 (Uniformity and NP $\cap$ coNP). *We have:*

$P = NP \cap coNP \iff \forall R \in \mathfrak{Unif}. \exists S \in \mathfrak{Det}.\ R \cong_{NP} S \iff \forall R \in \mathfrak{Unif}. \exists S \in \mathfrak{Det}.\ R \cong_{coNP} S$

[2] The polynomial hierarchy is an ordered set of complexity classes, smaller than P-SPACE, containing P, NP and coNP.

5 Conclusions

We have presented a new model of NP which is, as far as we know, the first implicit characterisation of NP through the Curry-Howard isomorphism.

Concerning the possible applications of this work for the study of the class NP, we cannot see a clear guideline but some points should be interesting. Some logical languages (such as Λ_{LA} defined in [16]) seem sufficient for programming purposes and one might have intuitions on their nondeterministic versions. Another work might be to characterise NP-complete problems in our system. One may also try to find a geometrical notion as the nesting of Sum boxes to get some continuum of complexity classes. Some links with semantics works [3,10] may also be interesting. Finally, following the first motivation, one should also understand whether the model can be adapted to LLL, SLL or FALL [11,17].

Thanks. Thanks to Patrick Baillot, Pierre-Louis Curien and Paul-André Melliès as well as to anonymous referees for their valuable comments and remarks.

References

1. Asperti A., **Light Affine Logic**. LICS98, 1998.
2. Asperti A., Roversi L., **Intuitionistic Light Affine Logic**. ACM Transactions on Computational Logic, 2002.
3. Baillot P., **Stratified coherent spaces: a denotational semantics for Light Linear Logic**. ICC00, 2000.
4. Bellantoni S., Cook S., **A new recursion-theoretic characterisation of the polytime functions**. Computational Complexity, 1992.
5. Ehrhard T., Regnier L., **The Differential Lambda-Calculus**. 2001.
6. Girard J.-Y., **Proof-nets: the parallel syntax for proof-theory**. Logic and Algebra. eds Ursini and Agliano, Marcel Dekker, New York 1996.
7. Girard J.-Y., **Light Linear Logic**. Information and Computation, 1998.
8. Girard J.-Y., **Locus Solum**. Mathematical Structures in Computer Science 11(3), 2001.
9. Hofmann M., **Linear types and non-size increasing polynomial time computation**. To appear in TCS, 1999.
10. Kanovich M. I., Okada M., Scedrov A., **Phase semantics for light linear logic**. Technical report, 1997.
11. Lafont Y., **Soft Linear Logic and Polynomial Time**. To appear in TCS.
12. Laurent O., **Polarized games**. Seventeenth annual IEEE symposium on Logic In Computer Science, IEEE Computer Society, 2002.
13. Leivant D., Marion J.Y., **Lambda calculus characterisations of poly-time**. TLCA 1993.
14. Murawski A. S., Ong C.-H. L., **Can safe recursion be interpreted in light logic ?**. Second International Workshop on Implicit Computational Complexity, 2000.
15. Papadimitriou C., **Computational complexity**. Addison Wesley, 1994.
16. Roversi L., **A P-Time Completeness Proof for Light Logics**. CSL99, 1999.
17. Roversi L., **Flexible Affine Light Logic**. Submitted, 2001.
18. Terui K., **Light Affine Calculus and Polytime Strong Normalisation**. LICS01, 2001.