Theoretical Computer Science 58 (1988) 175–181
North-Holland

DECOMPOSITIONS OF NONDETERMINISTIC REDUCTIONS

Klaus-Jörn LANGE

Fachbereich Informatik, Universität Hamburg, D-2000 Hamburg 13, Fed. Rep. Germany

Abstract. Nondeterministic reductions with a polynomial time bound or logarithmic space bound are characterized in terms of formal language operations like nonerasing homomorphisms and Kleene's star by relativizing the well-known equations $\mathrm{NP} = \mathrm{LOG}(H(\mathrm{DSPACE}(\log n)))$, $\mathrm{NSPACE}(\log n) = \mathrm{LOG}(H(1\text{-}\mathrm{DSPACE}(\log n)))$, and $\mathrm{NSPACE}(\log n) = \mathrm{LOG}(\mathrm{DSPACE}(\log n)^*)$. As corollaries we get $\Sigma_{k+1}^{\mathrm{P}} = \mathrm{LOG}(H(\Pi_k^{\mathrm{P}}))$ and $\mathrm{O}\Sigma_{k+1}^{\mathrm{L}} = \mathrm{LOG}((\mathrm{O}\Pi_k^{\mathrm{L}})^*)$. Further on, we derive the relation $\mathrm{NPOL}(A) = \mathrm{NLOG}(\mathrm{NLOG}(A))$ for every language A. Finally, we get that $\mathrm{A}\Sigma_{\log}^{\mathrm{L}}$ contains the logarithmic oracle hierarchy.

1. Introduction

This paper contains part of the author's "Habilitationsschrift" [7], where the following, roughly sketched results can be found in details, in particular Theorem 3.2, Theorem 5.2, and Corollary 5.5.

In [5] several types of deterministic and nondeterministic polynomial time reducibilities were introduced, among them Turing, conjunctive Turing, and many-one reducibilities. We are now going to relate and compare these three types with respect to determinism vs. nondeterminism and polynomial time vs. logarithmic space.

In this context the problem of space bounded relativization occurs. Essentially, there are three possibilities to bound the space of an oracle machine (respectively Turing transducer). One is to regard the oracle tape (respectively output tape) as a working tape. But this is not useful for sublinear, in particular logarithmic, space bounds. For instance, this case does not cover the many-one log-space reduction (see [15]). A second approach, by Ladner and Lynch in [4], bounds the running time and thus the length of the oracle tape (respectively output tape) exponentially by the space bound. But this restriction seems to be too weak in the nondeterministic case, where inclusions like $\mathrm{NSPACE}(\log n) \subset \mathrm{P}$ or $\mathrm{NSPACE}(\log n) \subset \mathrm{DSPACE}(\log^2 n)$ do not relativize in this way. This led to the third approach by Ruzzo, Simon and Tompa in [13]. They restrict the machines to working deterministically while writing on the oracle tape (respectively output tape). After having finished an oracle query, the oracle machines may continue nondeterministically.

Of course, the last two approaches coincide in the deterministic case. In the following we will relativize the equations

(1) $\mathrm{NP} = \mathrm{LOG}(H(\mathrm{DSPACE}(\log n)))$,

(2) $\mathrm{NSPACE}(\log n) = \mathrm{LOG}(H(1\text{-}\mathrm{DSPACE}(\log n)))$, and

(3) $\mathrm{NSPACE}(\log n) = \mathrm{LOG}(\mathrm{DSPACE}(\log n)^*)$,

where $H(\mathscr{A})$ denotes the class of all nonerasing homomorphic images of elements in a language class $\mathscr{A}$ and 1-DSPACE(log n) is the class of all languages, recognizable by deterministic log-space bounded Turing machines with a one-way input tape.

In Section 3 we will see that (1) above relativizes by a slight extension of the method of Cook in [2]. In Section 4, (2) will be relativized with respect to Ladner & Lynch reducibilities, while in Section 5 the relativization of (3) will use the Ruzzo, Simon, & Tompa reducibility.

2. Preliminaries

Let λ benote the empty work, $|v|$ the length of a word v, and v^R its reversal.

For a language A let Co-A be its complement. (We do not fix the underlying alphabet since its exchange needs intersections and unions with regular sets only. But these operations have no influence in the complexity of a problem.) If $\mathscr{A}$ is a language family, let $H(\mathscr{A})$ (respectively $\mathscr{A}^*$, Co-$\mathscr{A}$) be the class of all nonerasing homomorphic images of elements of $\mathscr{A}$ (respectively of their Kleene closure, of their complement).

Let $\mathscr{CF}$ denote the class of context-free languages. Further on, set L := DSPACE(log n), NL := NSPACE(log n) and let 1-L and 1-NL be the corresponding classes, where the input is given one-way.

The reader is assumed to be familiar with Turing reducibilities (performed by oracle Turing machines) and many-one reducibilities (performed by Turing transducers). If we consider deterministic reducibilities under a logarithmic space bound, we get the classes LOG(A) and L(A) of all sets many-one respectively Turing reducible to an (oracle) set A. If the underlying oracle machine works conjunctively (see [5]; i.e., if it has to reject whenever an oracle query is answered negatively), we get the class $L_c(A)$, which is closely related to the many-one reducibility by $L_c(A) = \text{LOG}((A\$)^*)$ for nonempty A, where \$ is a new symbol, not occurring in any word of A.

Working with alternating Turing machines, STA-notion (see, for instance, [12]) is useful. STA(f, g, h) denotes the set of all languages recognizable by alternating Turing machines, which are simultaneously f-space, g-time, and h-(-1) alternation bounded. The logarithmic alternation hierarchy (see [1]) is then defined by $A\Sigma_k^L := \text{STA}(\log, -, k)$ for $k \geq 1$. In [13] the logarithmic oracle hierarchy $(O\Sigma_k^L)_{k\geq 1}$ is introduced, which probably does not coincide with the logarithmic alternation hierarchy.

3. Polynomial time reducibilities

Ladner, Lynch, and Selman considered in [5] several types of polynomial time reducibilities, among them $\leq_T^{NP}$, $\leq_T^P$, $\leq_c^{NP}$, $\leq_c^P$, $\leq_m^{NP}$, and $\leq_m^P$ (the nondeterministic, respectively deterministic Turing, conjunctive Turing, and many-one reducibility; in [5] conjunctivity was considered for truth table reducibilities and not for Turing

reducibilities). We denote the closure of an (oracle) set A under $\leqslant_T^{NP}$ by $NP(A) := \{L \mid L \leqslant_T^{NP} A\}$ and define in the same way $P(A)$, $NP_c(A)$, $P_c(A)$, $NPOL(A)$, and $POL(A)$.

The following simple proposition is stated without proof.

Proposition 3.1. (a) $NP(A) = NP_c(0 \cdot A \cup 1 \cdot \text{Co-}A)$,
(b) $NP_c(A) = NPOL((A\$)^*)$, *and*
(c) $P_c(A) = POL(A\$)^*)$,
for every nonempty (oracle) language A, where \$ is a new symbol not occurring in any element of A.

Remark. The proof method of Proposition 3.1(a) does not work for deterministic reductions and neither for the strong nondeterministic reducibilities of Long in [8]. We only have $P(A) \supset P_c(0 \cdot A \cup 1 \cdot \text{Co-}A)$. In case of equality, Δ_2^P would coincide with its subclass D^P (see [10]) of intersections of elements of NP with elements of Co-NP.

Theorem 3.2

$$NPOL(A) = LOG(H(LOG(A))) \quad \textit{for arbitrary } A.$$

Proof (*sketch*). Since $NPOL(A)$ is closed under nonerasing homomorphisms, we get $LOG(H(LOG(A))) \subseteq NPOL(A)$. For the converse, let B a set many-one reduced to A by a nondeterministic, polynomial-time bounded machine M. Following the method of Cook in [2], we can encode computations of M on some input v into satisfiable boolean formulae $F(v)$, where the content of the output tape is given by the values of certain boolean variables of $F(v)$. Thus, we consider the language $L(A) := \{\langle F, x\rangle \mid x$ is an F-satisfying assignment encoding an output word $w \in A\}$. Along the lines of [9], it is possible to show $L(A) \in LOG(A)$: on input $\langle F, x\rangle$, decide in logarithmic space whether F is satisfied by x. If not, print a "rejection symbol" $\#$ not occurring in any word of A on the output tape. Else, print the output word of M (given by x).

Forgetting the assignment x by a letter-to-letter homomorphism (mapping 0's and 1's into 0's), we get the language $h(L(A)) \in H(LOG(A))$. Since $h(x)$ is independent of x (only depending in $|v|$), $v \to \langle F(v), h(x)\rangle$ is computable by a log-space Turing machine. Hence $B \in LOG(H(LOG(A)))$. □

By Proposition 3.1 we obtain the following corollary.

Corollary 3.3. (a) $NP_c(A) = LOG(H(L_c(A)))$ *and*
(b) $NP(A) = LOG(H(L(A)))$.

Remark. By [4, 14], there exist oracles A and B such that $NL(A) \subsetneq P(A)$ and $P(B) \subsetneq NL(B)$. On the other hand, Corollary 3.3 gives us, for all C,

$$LOG(H(L(C))) = LOG(H(NL(C))) = LOG(H(P(C))) = NP(C).$$

See also with the remark at the end of Section 5.

Clearly, Corollary 3.3 yields $\Sigma_k^P = \text{Log}(H(\Delta_k^P))$ for $k \geq 1$, which will be strengthened by the following result.

Proposition 3.4

$$\text{NP}(\text{NP}(A)) = \text{NP}_c(\text{Co-NP}(A)) \quad \textit{for arbitrary } A.$$

Proof. Obviously, $\text{NP}_c(\text{Co-NP}(A)) \subset \text{NP}(\text{Co-NP}(A)) = \text{NP}(\text{NP}(A))$. If $B \in \text{NP}(A)$ via a machine M and $C \in \text{NP}(\text{Co-B})$ via a machine M', then it is possible to simulate M' conjunctively by (pre-)guessing and (post-)verifying all oracle queries, choosing as an oracle the set $B' := 0 \cdot \text{Co-}B \cup 1 \cdot A \cup 2 \cdot \text{Co-}A$ which clearly is an element of Co-NP(A). Again, details can be found in [7]. □

As a consequence, we see that the H-operation characterizes the polynomial hierarchy, which was already shown by Wrathall [16] in a stronger version.

Corollary 3.5

$$\Sigma_{k+1}^P = \text{Log}(H(\Pi_k^P)) \quad \textit{for } k \geq 0.$$

Proof. This results from $\text{L}_c(\Pi_k^P) = \Pi_k^P$. □

4. Log space bounded reducibilities of the Ladner & Lynch type

Let NL(A) and L(A) be the relativization of NL and L with oracle A introduced by Ladner and Lynch in [4] (where these sets were denoted by NL^A and L^A). If the oracle machines concerned are restricted to work conjunctively (see [5]), we get the classes $\text{NL}_c(A)$ and $\text{L}_c(A)$. The corresponding many-one reducibilities yield $\text{NLog}(A)$ (see [6]) and the well-known class $\text{Log}(A)$.

In the following we need similar classes defined by deterministic log-space machines restricted to have a one-way input tape. These classes are denoted by 1-L(A), 1-$\text{L}_c(A)$, and 1-$\text{Log}(A)$.

Similar to the polynomial case in Section 3, it is possible to show for arbitrary, but nonempty A the following proposition.

Proposition 4.1. (a) $\text{NL}(A) = \text{NL}_c(0 \cdot A \cup 1 \cdot \text{Co-}A)$,
(b) $\text{NL}_c(A) = \text{NLog}((A\$)^*)$, *and*
(c) $\text{L}_c(A) = \text{Log}((A\$)^*)$.

The following result was shown in [6].

Theorem 4.2

$$\text{NLog}(A) = \text{Log}(H(1\text{-Log}(A))) \quad \textit{for arbitrary } A.$$

Now, Proposition 4.1 yields the following corollary.

Corollary 4.3. (a) $\mathrm{NL}_c(A) = \mathrm{LOG}(H(1-\mathrm{L}_c(A)))$, *and*
(b) $\mathrm{NL}(A) = \mathrm{LOG}(H(1-\mathrm{L}(A)))$.

These results give a tight connection between polynomial-time and logarithmic-space reducibilities of the Ladner & Lynch type as expressed in the following theorem.

Theorem 4.4. (a) $\mathrm{NLOG}(\mathrm{NLOG}(A)) = \mathrm{NPOL}(A)$,
(b) $\mathrm{NL}_c(\mathrm{NL}_c(A)) = \mathrm{NP}_c(A)$, *and*
(c) $\mathrm{NL}(\mathrm{L}(A)) = \mathrm{NL}_c(\mathrm{NL}(A)) = \mathrm{NP}(A)$.

Proof. $\mathrm{NLOG}(\mathrm{NLOG}(A)) = \mathrm{NLOG}(A)) = \mathrm{LOG}(H(\mathrm{LOG}(A))$ was shown in [6]. Applications of Propositions 3.1 and 4.1 as well as Theorem 3.2 give the result. □

Remark. $\mathrm{NL}(\mathrm{NL}(A)) = \mathrm{NL}(\mathrm{L}(A))$ for arbitrary A should not be expected since it would imply the collapse of the polynomial hierarchy (choose A to be a P-complete set and use Corollary 3.4).

5. Log-space reducibilities of the Ruzzo, Simon & Tompa type

Ruzzo, Simon and Tompa restricted in [13] the Ladner & Lynch reduction, by excluding nondeterministic generations of oracle queries. In the deterministic case, these reducibilities coincide with the unrestricted ones. For nondeterministic machines we get the classes $\mathrm{NL}\langle A\rangle$, $\mathrm{NL}_c\langle A\rangle$, and $\mathrm{NLOG}\langle A\rangle$. Thus a many-one reduction is performed by a log-space transducer which, after performing some nondeterministic computations without output, generates deterministically the output. Similar to the previous sections, we get for an arbitrary oracle A the following proposition.

Proposition 5.1. $\mathrm{NL}\langle A\rangle = \mathrm{NL}_c\langle 0\cdot A \cup 1\cdot \mathrm{Co}\text{-}A\rangle$.

Remark. $\mathrm{NL}_c\langle A\rangle = \mathrm{NLOG}\langle (A\$)^*\rangle$—corresponding to Propositions 3.1(b) and 4.1(b)—would imply $\mathrm{A}\Sigma_2^{\mathrm{L}} = \mathrm{O}\Sigma_2^{\mathrm{L}}$ and hence the collapse of both the logarithmic oracle hierarchy and of the logarithmic alternation hierarchy (see [7]).

Theorem 5.2. $\mathrm{NL}_c\langle A\rangle = \mathrm{LOG}(L_c(A)^*) = \mathrm{LOG}(\mathrm{LOG}(A)^*)$. (*The latter inclusion holds for nonempty A only.*)

Proof (*sketch*). We have $\mathrm{LOG}(A) \subset \mathrm{L}_c(A)$ by definition and $\mathrm{L}_c(A)^* \subset \mathrm{NL}_c\langle A\rangle$ by the *-closure of $\mathrm{NL}_c\langle A\rangle$. To show $\mathrm{NL}_c\langle A\rangle \subset \mathrm{LOG}(\mathrm{LOG}(A)^*)$, we encode computations of a conjunctive nondeterministic logarithmical-space bounded oracle machine M

with oracle A into the set $\text{LIN}(A)^*$, where, for $A \subset Y^*$, output alphabet X, and new symbols ¢ and \$, we define

$$\text{LIN}(A) := \{w¢y¢uw^R\$ \mid y \in A,\ u \in (X^*\$X^*¢Y^*¢)^*, \text{ and } w \in X^*\}.$$

This encoding can be done within logarithmic space by extending the method of Flajolet and Steyaert, who showed $\text{NL} = \text{LOG}(\text{L}^*)$ in [3]. □

Corollary 5.3. $\text{NL}\langle A\rangle = \text{LOG}(\text{L}(A)^*)$.

With the same methods used in Section 3 we obtain the following corollary.

Corollary 5.4

$$\text{O}\Sigma^{\text{L}}_{k+1} = \text{LOG}((\text{O}\Pi^{\text{L}}_k)^*) = \text{NL}_c\langle \text{O}\Pi^{\text{L}}_k\rangle \quad \textit{for } k \geq 0.$$

Remark. In [7], $\text{A}\Sigma^{\text{L}}_{k+1} = \text{NLOG}\langle \text{A}\Pi^{\text{L}}_k\rangle$ is shown for $k \geq 1$.

As a further consequence, we get a new common upper bound for the context-free languages and the logarithmic oracle hierarchy.

Consider the class $\text{A}\Sigma^{\text{L}}_{\log} := \text{STA}(\log, -, \text{O}(\log)) := \bigcup_{c>0}\text{STA}(\log, -, c \cdot \log)$. The inclusions $\mathscr{CF} \subset \text{A}\Sigma^{\text{L}}_{\log} \subset \text{NC}^2$ (the second level of the Pippenger hierarchy; see [11, 12]) are well-known. Since it is possible to show the *-closure of $\text{A}\Sigma^{\text{L}}_{\log}$ by an alternating version of the Savitch algorithm (see [7]), we can improve the inclusion $\bigcup_{k\geq 1}\text{O}\Sigma^{\text{L}}_k \subset \text{A}\Sigma^{\text{L}}_{\log^2} := \text{STA}(\log, -, \text{O}(\log^2))$ in [12] to inclusion in the following corollary.

Corollary 5.5. $\bigcup_{k\geq 1}\text{O}\Sigma^{\text{L}}_k \subset \text{A}\Sigma^{\text{L}}_{\log}$.

Proof (*by induction*). $k = 1$: $\text{O}\Sigma^{\text{L}}_1 = \text{A}\Sigma^{\text{L}}_1 \subset \text{A}\Sigma^{\text{L}}_{\log}$. Since $\text{A}\Sigma^{\text{L}}_{\log}$ is closed under complement and Kleene's star, $\text{O}\Sigma^{\text{L}}_k \subset \text{A}\Sigma^{\text{L}}_{\log}$ implies $\text{O}\Sigma^{\text{L}}_{k+1} = \text{LOG}((\text{O}\Pi^{\text{L}}_k)^*) \subset \text{A}\Sigma^{\text{L}}_{\log}$ by Corollary 5.4. □

Remark. Another consequence of Theorem 5.2 is the "unseparability" of the Ruzzo, Simon & Tompa reducibility: it is possible to show that $\text{L} = \text{NL}$ implies $\text{L}(A) = \text{NL}\langle A\rangle$ for each oracle set A (details can be found in [17]) in contrast to the Ladner & Lynch reducibility (see [4, 14]).

References

[1] A. Chandra, D. Kozen and L. Stockmeyer, Alternation, *J. Assoc. Comput. Mach.* **28** (1981) 114–133.

[2] S. Cook, The complexity of theorem proving procedures, in: *Proc. 3rd Ann. Symp. on Theory of Computing* (1971) 151–158.

[3] P. Flajolet and J. Steyaert, Complexity of classes of languages and operators, Rapt. de Recherche No. 92, IRIA Laboria, 1974.

[4] R. Ladner and N. Lynch, Relativization of questions about log-space computability, *Math. Systems Theory* **10** (1976) 19–32.
[5] R. Ladner, N. Lynch and A. Selman, A comparison of polynomial time reducibilities, *Theoret. Comput. Sci.* **1** (1975) 103–123.
[6] K.-J. Lange, Nondeterministic log-space reductions, in: *Proc. 11th Symp. of Mathematical Foundations of Computer Science* (1984) Lecture Notes in Computer Science **176** Springer, Berlin, 1984) 378–388.
[7] K.-J. Lange, Nichtdeterministische Reduktionen und Logarithmische Hierarchien, Techn. Rep. 119/86, Institut für Informatik, University of Hamburg, 1985 (in German).
[8] T. Long, Strong nondeterministic polynomial-time reducibilities, *Theoret. Comput. Sci.* **21** (1982) 1–25.
[9] N. Lynch, Log-space recognition and translation of parenthesis languages, *J. Assoc. Comput. Mach.* **24** (1977) 583–590.
[10] C. Papadimitriou and M. Yannakakis, The complexity of facets (and some facets of complexity), *J. Comput. System Sci.* **28** (1984) 244–259.
[11] W. Ruzzo, Tree-size bounded alternation, *J. Comput. System Sci.* **21** (1980) 218–235.
[12] W. Ruzzo, On uniform circuit complexity, *J. Comput System Sci.* **22** (1981) 365–383.
[13] W. Ruzzo, J. Simon and M. Tompa, Space-bounded hierarchies and probabilistic computations, *J. Comput. System Sci.* **28** (1984) 216–230.
[14] W. Savitch, A note on relativized log space, *Math. Systems Theory* **16** (1983) 229–235.
[15] L. Stockmeyer and A. Meyer, Word problems requiring exponential time: preliminary report, in: *Proc. 5th Ann. ACM Symp. on Theory of Computing* (1973) 1–9.
[16] C. Wrathall, Complete sets and the polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1976) 23–33.
[17] B. Kirsig and K.-J. Lange, Separation with the Ruzzo, Simon, and Tompa relativization implies $\text{DSPACE}(\log n) \neq \text{NSPACE}(\log n)$, *Inform. Process. Lett.* **25** (1987) 13–15.