The PL Hierarchy Collapses

Mitsunori Ogihara
Department of Computer Science
University of Rochester
Rochester, NY 14627

February 14, 1996

Abstract

It is shown that the PL hierarchy

\[ PLH = PL \cup PL^\text{PL} \cup PL^{PLP} \cup \ldots \]

defined in terms of the Ruzzo-Simon-Tompa relativization collapses to PL.

1 Introduction

The oracle separations proven by Baker, Gill, and Solovay [BGS75] initiated the study of complexity classes by relativization. In order to study the NL =?L question, various relativization models for nondeterministic logspace have been proposed [LL76,Sim77,RS81,RST84]. Among them, the so-called Ruzzo-Simon-Tompa model (the RST-model, in short) [RST84], which demands that nondeterministic Turing machines run deterministically while generating query strings, is widely accepted because of its reasonability—for any oracle A, L^A \subseteq NL^A \subseteq P^A. Given this reasonable model of relativization, it is quite reasonable for one to wonder what are the complexity classes defined by stacking logspace complexity classes: for a logspace class C, does the C hierarchy in terms of the RST-model collapse? The answer to this question was given for some classes. Ruzzo, Simon, and Tompa showed that the hierarchy with respect to BPL [Gil77] (the bounded-error probabilistic logspace with unlimited computation time) collapses to BPL. Also, the NL = coNL theorem proven independently by Immerman [Imm88] and Szelepcsényi [Sze88] implies that the NL hierarchy collapses to NL. In this paper, we obtain the answer to the question for PL (the probabilistic logspace with unlimited computation time) [Gil77]: the PL hierarchy collapses to PL.

Our proof is built on top of some precedent work. Beigel, Reingold, and Spielman [BRS95] showed that PP is closed under intersection. Their proof makes use of the rational functions of Paturi and Saks [PS94] to approximate threshold functions, which extends the work of Newman [New64]. Furthermore, Fortnow and Reingold [FR96] strengthened the technique and showed that PP is even closed under polynomial-time constant round truth-table reductions. Intuitively, we show that the proof by Fortnow and Reingold can be carried over to PL. To this end, we use a
characterization of PL in terms of polynomial time-bounded nondeterministic logspace machines derived from Jung’s result [Jun85] that PL is equal to the polynomial time-bounded PL. Such a characterization is shown in Allender and Ogihara [AO94], where they prove that PL is closed under both conjunctive truth-table reductions and disjunctive truth-table reductions.

2 Preliminaries

In this section, we set down some notation and define relevant complexity classes. The alphabet we use is \( \Sigma = \{0,1\} \). \( \mathbb{Z} \) and \( \mathbb{N} \) respectively denote the set of all integers and the set of all nonnegative integers. \( \langle \cdot, \cdot \rangle \) denotes a logspace computable and logspace invertible pairing function (not necessarily onto).

The class PL was originally defined by Gill [Gil77].

**Definition 2.1** [Gil77] A language \( L \) belongs to PL if there exists a logarithmic space-bounded probabilistic Turing machine \( M \) with unlimited computation time such that for every \( x, x \in L \) if and only if the probability that \( M \) on \( x \) accepts is at least a half.

Let \( \text{PL}_{\text{poly}} \) denote the polynomial time-bounded version of PL. Jung [Jun85] showed that PL = \( \text{PL}_{\text{poly}} \), and furthermore, Allender and Ogihara [AO94] showed that the equivalence holds relative to any oracle.

**Proposition 2.2** [AO94] For every oracle \( H \), \( \text{PL}^H = (\text{PL}_{\text{poly}})^H \).

Based on the above equivalence, one can obtain a characterization of PL in terms of nondeterministic Turing machines. For a time-bounded nondeterministic Turing machine \( M \) and \( x \in \Sigma^* \), let \( \text{acc}_M(x) \) and \( \text{ rej}_M(x) \) respectively denote the number of accepting computation paths and that of rejecting computation paths of \( M \) on \( x \) and let \( \text{ gap}_M(x) \) denote \( \text{acc}_M(x) - \text{ rej}_M(x) \). Define the complexity class \( \text{GapL} \) [AO94] (see also, GapP [FFK94]) as follows.

**Definition 2.3** \( \text{GapL} = \{ \text{gap}_M \mid M \text{ is a logarithmic space-bounded, polynomial time-bounded nondeterministic Turing machine} \} \).

The following propositions are proven by Allender and Ogihara [AO94].

**Proposition 2.4** [AO94] A language \( L \) belongs to PL if and only if there exists some \( f \in \text{GapL} \) such that for every \( x, x \in L \) if and only if \( f(x) \geq 0 \).

**Proposition 2.5** Let \( f \) be a function in \( \text{GapL} \), \( g : \Sigma^* \times \mathbb{N} \rightarrow \Sigma^* \) be a function in PL, and \( p \) be a polynomial. Then the following functions \( h_1, h_2, \) and \( h_3 \) all belong to \( \text{GapL} \):

1. \( h_1(x) = -f(x) \).
2. \( h_2(x) = \Sigma_{i=1}^{p(|x|)} f(g(x,i)) \).
3. \( h_3(x) = \Pi_{i=1}^{p(|x|)} f(g(x,i)) \).
Given a function $f \in \text{GapL}$ witnessing that a language $L$ is in PL, define $g$ by $g(x) = 2f(x) + 1$. Then $g$ always takes on odd values and witnesses that $L$ is in PL. By Proposition 2.5, $g$ belongs to GapL. So, we have the following characterization of PL.

**Proposition 2.6** A language $L$ is in PL if and only if there exists a function $f$ in GapL such that for every $x$,

$$f(x) \geq 1 \text{ if } x \in L \text{ and } f(x) \leq -1 \text{ otherwise.}$$

### 2.1 GapL functions to approximate the characteristic function of languages in PL

Proposition 2.6 states that the problem of testing whether a GapL function takes a positive or a negative value characterizes PL. Newman [New64] show that the sign function can be approximated by the fraction of two polynomials. The Newman’s construction gives us a method for approximating threshold functions by rational functions [PS94,BRS95,FR96].

**Definition 2.7** Let $m \geq 1$ and $k \geq 1$. Define polynomials $P_m(z)$ and $Q_m(z)$ in $\mathbb{Z}[z]$ by

\begin{align*}
(1) \quad P_m(z) &= (z - 1) \prod_{i=1}^{m} (z - 2^i)^2 \quad \text{and} \\
(2) \quad Q_m(z) &= -(P_m(z) + P_m(-z)),
\end{align*}

and define $R_{m,k}(z)$ and $S_{m,k}(z)$ by

\begin{align*}
(3) \quad R_{m,k}(z) &= \left(\frac{2P_m(z)}{Q_m(z)}\right)^{2k} \quad \text{and} \\
(4) \quad S_{m,k}(z) &= (1 + R_{m,k}(z))^{-1}.
\end{align*}

Furthermore, define polynomials $A_{m,k}(z)$ and $B_{m,k}(z)$ by

\begin{align*}
(5) \quad A_{m,k}(z) &= Q_m(z)^2 \quad \text{and} \\
(6) \quad B_{m,k}(z) &= Q_m(z)^2 + (2P_m(z))^{2k}
\end{align*}

**Lemma 2.8** For every $m, k \geq 1$ in $\mathbb{N}$ and every $z \in \mathbb{Z}$, the following properties hold.

1. $S_{m,k}(z) = A_{m,k}(z) / B_{m,k}(z)$.
2. If $1 \leq z \leq 2^m$, then $1 - 2^{-k} \leq S_{m,k} \leq 1$.
3. If $-2^m \leq z \leq -1$, then $0 \leq S_{m,k}(z) \leq 2^{-k}$.

**Proof** Let $m, k \geq 1$ be in $\mathbb{N}$. The first equivalence is proven by the routine calculation, so, we omit the proof. Note that $P_m(z) \geq 0$ if and only if $z \geq 1$. Let $z$ be in $\{1, \ldots, 2^m\}$. We claim that $P_m(z) \leq |P_m(-z)|/4$. This is seen as follows: If $z = 1$, then $P_m(z) = 0$, so, the claim holds. On the other hand, if $z \geq 2$, then there exists a unique $t, 1 \leq t \leq m$, such that $2^t \leq z < 2^{t+1}$, and
this $t$ satisfies $|z - 2^t| \leq z/2 \leq |z - 2^t|/2$. Since $|z - 1| \leq |z - 1|$ and for every $i, 1 \leq i \leq m$, $|z - 2^i| \leq |z - 2^i|$, we have $P_m(z) \leq |P_m(-z)|/4$.

The claim is proven. So, for every $z \in \mathbb{Z}$,

$$0 \leq \frac{2P_m(z)}{Q_m(z)} \leq \frac{2}{3} \quad \text{if } 1 \leq z \leq 2^m \text{ and}$$

$$\frac{2P_m(z)}{Q_m(z)} \leq -2 \quad \text{if } -2^m \leq z \leq -1.$$

Since $(2/3)^2 \leq 1/2$, for every $z \in \mathbb{Z}$,

$$0 \leq R_{m,k}(z) \leq 2^{-k} \quad \text{if } 1 \leq z \leq 2^m \text{ and}$$

$$R_{m,k}(z) \geq 2^{k} \quad \text{if } -2^m \leq z \leq -1.$$

Since $S_{m,k}(z) = (1 + R_{m,k}(z))^{-1}$ and $(1 + 2^{-k})(1 - 2^{-k}) < 1$, for every $z, 1 \leq z \leq 2^m$,

$$1 - 2^{-k} \leq S_{m,k}(z) \leq 1.$$

Also, since $(1 + 2^k)^{-1} \leq 2^{-k}$, for every $z, -2^m \leq z \leq -1$,

$$0 \leq S_{m,k}(z) \leq 2^{-k}.$$

This proves the lemma.

\section{The PL Hierarchy Collapses}

The following lemma states that logarithmic space-bounded oracle Turing machines can be normalized so that the queries, including the query order, are independent of the oracle.

\newcommand{\m}{m}

\begin{lemma}
Let $L \in \text{PL}^H$ for some oracle $H$. Then there exist polynomials $p$ and $q$ and a logarithmic space-bounded nondeterministic Turing machine $N$ such that for every $x$,

1. independent of the oracle and the nondeterministic choices, $N$ on $x$ makes exactly $p(|x|)$ queries and exactly $q(|x|)$ nondeterministic moves, and furthermore, $N$ on $x$ makes no nondeterministic moves while generating queries; and

2. $x \in L$ if and only if $\text{gap}_N(x) \geq 0$.
\end{lemma}

\begin{proof}
Let $M$ be the base probabilistic logarithmic space-bounded machine witnessing that $L \in \text{PL}^H$. By Proposition 2.2, we may assume that $M$ is polynomial time-bounded. There is a polynomial $q$ such that for every $x$, $M$ on $x$ tosses at most $q(|x|)$ coins regardless of its oracle. Without changing the acceptance probability, we can modify $M$ so that $M$ tosses exactly $q(|x|)$ coins. Then by replacing the coin tosses of $M$ by nondeterministic moves, $M$ becomes a nondeterministic oracle Turing machine satisfying the condition on the number of nondeterministic moves in (1) as well as (2). We will construct a new machine $N$ from this $M$ so that the condition on the query strings is met while preserving the other properties. Recall that the RST-model
demands that $M$ should run deterministically while it generates query strings. So, without loss of generality, we may assume that $M$ has a special state, called GENERATE-state, such that (i) $M$ enters GENERATE-state if and only if it is at the beginning of query string generation and (ii) once it enters GENERATE-state, $M$ runs deterministically until it enters QUERY-state. For each $n$, let $\mathcal{T}_n$ be the set of all IDs of $M$ on an input of length $n$ at GENERATE-state. For every input $x$ of length $n$ and every potential query string $y$ of $M$ on $x$, there is an ID $I \in \mathcal{T}_n$ such that $M$ on $x$ generates $y$ as the query string from ID $I$, and thus, simulation of $M$ on $x$ from ID $I$ generates $y$. Furthermore, since $M$ is logarithmic space-bounded, $\mathcal{T}_n$ is bounded by some polynomial in $n$. Let $r_1$ be such a polynomial. Also, since $M$ is polynomial time-bounded, let $r_2$ be a polynomial bounding the run-time of $M$. Now define $p(n) = r_1(n) r_2(n)$ and define $N$ to be the machine that, on input $x$, simulates $M$ on $x$ as follows:

- At the very beginning of the computation, $N$ sets a binary counter $c$ to 0.
- When $M$ enters GENERATE-state, $N$ records the current ID $I$ of $M$.
- When $M$ enters QUERY-state, $N$ increments the counter $c$, resets a binary counter $d$, and does the following:
  - By cycling through all IDs $J$ in $\mathcal{T}_{|x|}$, $N$ asks its oracle all potential query strings of $M$ on $x$. Each time a query is made, $N$ increments the counter $d$. If $J = I$, then $N$ records the answer $b$ from the oracle. Otherwise, $N$ ignores the answer from the oracle.
  - When the above process is done, if $d < r_1(|x|)$, then $N$ queries some fixed string $u$, e.g., the empty string, $r_1(|x|) - d$ times.
  - $N$ returns to the simulation of $M$ on $x$ with $b$ as the answer to the current query of $M$.
- When $M$ enters a halting state, if $c < r_2(|x|)$, then $N$ executes the above query process $r_2(|x|) - c$ times, but this time, $N$ ignores all the answers from the oracle. After accomplishing this, $N$ accepts if and only if $M$ has accepted.

Note that $N$ on $x$ makes exactly $q(|x|)$ nondeterministic moves and the number of accepting computation paths of $N$ on $x$ is identical to that of $M$ on $x$. The number of queries of $N$ on $x$ is exactly $p(|x|)$ regardless of its oracle. For every $i, 1 \leq i \leq p(|x|)$, the $i$th query string of $N$ on $x$ is determined independent of its oracle or its nondeterministic moves. Thus, the remaining part of the condition (1) is met. This proves the lemma.

**Theorem 3.2** $\text{PL}^{\text{PL}} = \text{PL}$.

**Proof** Let $L \in \text{PL}^{\text{PL}}$ be witnessed by a nondeterministic Turing machine $N$ and a language $H \in \text{PL}$ satisfying the conditions in Lemma 3.1 with polynomials $p$ and $q$. For each $x$ and $i, 1 \leq i \leq p(|x|)$, let $y_{x,i}$ denote the $i$th query string of $N$ on $x$. Let $f$ be a function in GapL witnessing that $H \in \text{PL}$ as in Proposition 2.6. There exists a polynomial $\mu$ such that for every $x$ and $i, 1 \leq i \leq p(|x|), 1 \leq |f(y_{x,i})| \leq 2^{\mu(|x|)}$. Let us fix such a polynomial $\mu$. Define $\kappa(n) = p(n) + q(n) + 1$ and for each $x$ and $i, 1 \leq i \leq p(|x|)$, define

$$T(x, i, 1) = S_{\mu, \kappa}(f(y_{x,i}))$$

and

$$T(x, i, 0) = 1 - S_{\mu, \kappa}(f(y_{x,i})).$$

5
where $S_{\nu, \kappa}$ is the short-hand of $S_{\nu(\lfloor x \rfloor), \kappa(\lfloor x \rfloor)}$. By Lemma 2.8, for every $x$, $i, 1 \leq i \leq p(\lfloor x \rfloor)$, and $b \in \{0, 1\}$,

(7) if $\chi_H(y_{x,i}) = b$, then $1 - 2^{\alpha(\lfloor x \rfloor)} \leq T(x, i, b) \leq 1$, and

(8) if $\chi_H(y_{x,i}) \neq b$, then $0 \leq T(x, i, b) \leq 2^{\alpha(\lfloor x \rfloor)}$.

Furthermore, define

\[
\alpha(x, i, 1) = A_{\nu, \kappa}(f(y_{x,i})), \\
\alpha(x, i, 0) = B_{\nu, \kappa}(f(y_{x,i})) - A_{\nu, \kappa}(f(y_{x,i})), \text{ and} \\
\beta(x, i) = B_{\nu, \kappa}(f(y_{x,i})),
\]

where $A_{\nu, \kappa}$ is the short-hand of $A_{\nu(\lfloor x \rfloor), \kappa(\lfloor x \rfloor)}$ and $B_{\nu, \kappa}$ is the short-hand of $B_{\nu(\lfloor x \rfloor), \kappa(\lfloor x \rfloor)}$. Then for every $x$, $i, 1 \leq i \leq p(\lfloor x \rfloor)$, and $b \in \{0, 1\}$,

\[
T(x, i, b) = \frac{\alpha(x, i, b)}{\beta(x, i)}.
\]

For each $x$ and $w \in \{0, 1\}^{p(\lfloor x \rfloor)}$, define

\[
C(x, w) = \prod_{i=1}^{p(\lfloor x \rfloor)} T(x, i, w_i),
\]

where $w_i$ denotes the $i$th bit of $w$. Then, by (7) and (8), we have

(9) if $w = \chi_H(y_{x,1}) \cdots \chi_H(y_{x,p(\lfloor x \rfloor)})$, then $1 - p(\lfloor x \rfloor)2^{\alpha(\lfloor x \rfloor)} \leq C(x, w) \leq 1$, and

(10) if $w \neq \chi_H(y_{x,1}) \cdots \chi_H(y_{x,p(\lfloor x \rfloor)})$, then $0 \leq C(x, w) \leq 2^{\alpha(\lfloor x \rfloor)}$.

Define

\[
\gamma(x, w) = \prod_{i=1}^{p(\lfloor x \rfloor)} \alpha(x, i, w_i) \quad \text{and} \\
\delta(x) = \prod_{i=1}^{p(\lfloor x \rfloor)} \beta(x, i)
\]

Then, for every $x$ and $w$,

\[
C(x, w) = \gamma(x, w) / \delta(x).
\]

Define predicate $e$ as follows:

(11) For each $x$, $w$, $|w| = p(\lfloor x \rfloor)$, and $u$, $|u| = q(\lfloor x \rfloor)$, $e(x, w, u) = 1$ if and only if $M$ on $x$ with nondeterministic guesses $u$ accepts assuming that the answer to the $i$th query is affirmative if and only if $w_i = 1$.

Define

\[
D(x) = \sum_{w : |w| = p(\lfloor x \rfloor), |u| = q(\lfloor x \rfloor)} e(x, w, u)C(x, w) \quad \text{and} \\
\theta(x) = \sum_{u : |u| = q(\lfloor x \rfloor)} e(x, w, u)\gamma(x, w).
\]

Clearly, $D(x) = \theta(x) / \delta(x)$. By (9) and (10), the following properties hold.
1. There is a unique \( w_x \in \Sigma^{|\sigma|} \) such that
\[
1 - p(|x|)2^{-\alpha(|x|)} \leq C(x, w_x) \leq 1
\]
and for every \( w \neq w_x \),
\[
0 \leq C(x, w) \leq 2^{-\alpha(|x|)}.
\]
2. If \( x \in L \), then the number of \( u, |u| = q(|x|) \), such that \( e(x, w_x, u) = 1 \) is at least \( 2^{\varphi(|x|)-1} \).
3. If \( x \notin L \), then the number of \( u, |u| = q(|x|) \), such that \( e(x, w_x, u) = 1 \) is at most \( 2^{\varphi(|x|)-1} - 1 \).

Since \( \kappa(n) = p(n) + q(n) + 1 \), for every \( x \), if \( x \in L \), then
\[
D(x) = 2^{\varphi(|x|)-1} (1 - p(|x|)2^{-\alpha(|x|)})
\]
\[
\geq 2^{\varphi(|x|)-1} (1 - 2^{\varphi(|x|)}2^{-\alpha(|x|)})
\]
\[
= 2^{\varphi(|x|)-1} - 2^{-2}
\]
\[
= 2^{\varphi(|x|)-1} - 1/4,
\]
and if \( x \notin L \), then
\[
D(x) \leq (2^{\varphi(|x|)-1} - 1) + 2^{\varphi(|x|)-1} \varphi(|x|)2^{-\alpha(|x|)}
\]
\[
= 2^{\varphi(|x|)-1} - 1 + 2^{-1}
\]
\[
= 2^{\varphi(|x|)-1} - 1/2.
\]
This implies for every \( x \),
\[
x \in L \text{ if and only if } D(x) \geq 2^{\varphi(|x|)-1} - \frac{1}{4}.
\]
Finally, define \( h(x) = 4\theta(x) - (2^{\varphi(|x|)+1} - 1)\delta(x) \). Then, for every \( x, x \in L \) if and only if \( h(x) \geq 0 \).

We claim that \( h \in \text{GapL} \). Define \( \pi \) to be the function that maps each \( w \) to \( 2|w| \). It is obvious that \( \pi \in \text{GapL} \). Thus, by Theorem 2.5, the function that maps each \( x \) to \( \mathcal{P}_n(|x|)(f(x)) \), i.e.,
\[
(f(x) - 1) \prod_{|a| \leq |x|} (f(x) - \pi(0))^2,
\]
is in \( \text{GapL} \). For much the same reason, the function that maps each \( x \) to \( \mathcal{Q}_n(|x|)(f(x)) \) is in \( \text{GapL} \). Since \( y_{x,i} \) is logarithmic-space computable, by Theorem 2.5, \( \alpha, \beta \in \text{GapL} \). This implies \( \delta \in \text{GapL} \). Since the function that maps each \( x \) to \( 2^{\varphi(|x|)+1} - 1 \) belongs to \( \text{GapL} \), the proof will be completed if we show that \( \theta \in \text{GapL} \).

Let \( M \) be such that \( \alpha = \text{gap}_M \). Define \( G \) to be the nondeterministic Turing machine that, on input \( x \), behaves as follows:

**Step 1** \( G \) first sets a one-bit counter \( c \) to 0.

**Step 2** \( G \) starts simulating \( N \) on \( x \) nondeterministically; that is, if \( N \) makes its \( i \)th nondeterministic move, then so does \( G \) thereby guessing bit \( u_i \). When \( N \) makes its \( i \)th query \( y_{x,i} \), \( G \) does the following.

(a) \( G \) nondeterministically guesses \( w_i \in \{0, 1\} \) and simulates \( M \) on \( \langle x, i, w_i \rangle \). If \( M \) rejects, then \( G \) flips the bit \( c \).
(b) $G$ returns to the simulation of $N$ on $x$ assuming that the answer to the query is affirmative if and only if $w_i = 1$.

**Step 3** When $N$ enters the halting state, $G$ does the following.

(a) If $N$ has accepted, then $G$ accepts if and only if $c = 0$.
(b) If $N$ has rejected, then $G$ nondeterministically guesses a bit $d \in \{0, 1\}$ and accepts if and only if $d = 0$.

Note that, at the beginning of Step 3, $e(x, w, u) = 1$ holds if and only if $N$ has accepted with $w$ and $u$. In the case that $N$ has rejected, i.e., $e(x, w, u) = 0$, $G$ generates one accepting path and one rejecting path, so, there is no contribution to $\text{gap}_G(x)$ along $w$ and $u$. In the case that $N$ has accepted, i.e., $e(x, w, u) = 1$, the one-bit counter $c$ is the parity of the number of accepting simulations of $M$ that $G$ has encountered. Since $G$ accepts if and only if the parity is 0, the number of accepting computation paths along $w$ and $u$ is the sum of all

$$\prod_{i \in I} \text{acc}_M(x, i, w_i) \prod_{i \in I} \text{rej}_M(x, i, w_i),$$

where $I$ ranges over all subsets of $\{1, \ldots, p(|x|)\}$ of even cardinality. Also, the number of rejecting computation paths along $w$ and $u$ is the sum of all

$$\prod_{i \in I} \text{acc}_M(x, i, w_i) \prod_{i \in I} \text{rej}_M(x, i, w_i),$$

where $I$ ranges over all subsets of $\{1, \ldots, p(|x|)\}$ of odd cardinality. Note for every $i$ and $w_i$, that $\text{acc}_M(x, i, w_i) - \text{rej}_M(x, i, w_i) = \text{gap}_M(x, i, w_i)$. Thus, the difference between the above two sums is equal to

$$\prod_{i = 1}^{p(|x|)} (\text{acc}_M(x, i, w_i) - \text{rej}_M(x, i, w_i)) = \prod_{i = 1}^{p(|x|)} \text{gap}_M(x, i, w_i).$$

Thus, for every $x$,

$$\text{gap}_G(x) = \sum_{w, u: |w| = p(|x|), |u| = q(|x|)} e(x, w, u) \prod_{i = 1}^{p(|x|)} \alpha(x, i, w_i)$$

$$= \sum_{w, u: |w| = p(|x|), |u| = q(|x|)} e(x, w, u) \gamma(x, w)$$

$$= \theta(x)$$

Since both $N$ and $M$ are logarithmic space-bounded, so is $G$. Hence, $\theta$ is in GapL. This proves the theorem.

Allender and Ogihara [AO94] observe that the PL hierarchy coincides with the logspace-uniform $\text{AC}^0$ closure of PL. So, we immediately obtain the following corollary.

**Corollary 3.3** $\text{PLH} = \text{AC}^0(\text{PL}) = \text{PL}$.

This gives rise to question whether PL is closed under logspace-uniform $\text{NC}^1$-reductions. Very recently, the question has been resolved affirmatively by Beigel [Bei].
Acknowledgment

The author would like to thank Eric Allender for enjoyable discussions and insightful comments and Marius Zimand and Gabi Istrate for careful reading of the manuscript.

References


