JOURNAL OF COMPUTER AND SYSTEM SCIENCES **55**, 547–560 (1997)
ARTICLE NO. SS971516

Universally Serializable Computation

Lane A. Hemaspaandra* and Mitsunori Ogihara†

Department of Computer Science, University of Rochester, Rochester, New York 14627

Received May 21, 1996

Cai and Furst proved that every PSPACE language can be solved via a large number of identical simple tasks, each of which is provided with the original input, its own unique task number, and at most three bits of output from the previous task. In the Cai–Furst model, the tasks are required to be run in the order specified by the task numbers. To study the extent to which the Cai–Furst PSPACE result is due to this strict scheduling, we remove their ordering restriction, allowing tasks to execute in any serial order. That is, we study the extent to which complex tasks can be decomposed into large numbers of simple tasks that can be scheduled arbitrarily. We provide upper bounds on the complexity of the sets thus accepted. Our bounds suggest that Cai and Furst's surprising PSPACE result is due in large part to the fixed order of their task execution. In fact, our bounds suggest the possibility that even relatively low levels of the polynomial hierarchy cannot be accepted via large numbers of simple tasks that can be scheduled arbitrarily. However, adding randomization recaptures the polynomial hierarchy. The entire polynomial hierarchy can be accepted by large numbers of arbitrarily scheduled probabilistic tasks passing only a single bit of information between successive tasks (and using J. Simon's "exact counting" acceptance mechanism). In fact, we show that the class of languages so accepted is exactly $\mathrm{NP}^{\mathrm{PP}}$. © 1997 Academic Press

1. INTRODUCTION

A key theme in modern computer science is the decomposition of a complex task into a number of smaller, simpler tasks. Although this is often studied in the context of parallel computation, Cai and Furst [CF91] investigate this issue in terms of serial computation. They propose a model in which a given computation is decomposed into an exponential number of polynomial-time tasks that are identical except that each is given, in addition to the problem input, its own unique task number plus a few bits of output allowed to be passed on from the previous task. The computation "accepts" if the final task outputs some particular value (e.g., "1"). Building on Barrington's [Bar89] seminal work on branching programs, Cai and Furst prove that every PSPACE language can be accepted in this way, even if the tasks run in logarithmic space rather than in polynomial time. Keeping in mind that Cai and Furst additionally require the tasks to be executed in the lexicographical order of their task numbers and, indeed, use the term "clock" in place of what here is called a task number, let us consider their explanation of their surprising result:

> this result seems to be paradoxical; but of course there is no contradiction here, as the clock provides... weak access to space. It seems remarkable that any PSPACE language computation can be "strung out" in such a fashion. (One would not have expected that the rise and fall of the sun or other terrestrial bodies provided us with any [such] valuable computing resource as space.) [CF91]

They are indeed correct in pointing out that the rising and setting of the sun, in this setting, provides a weak form of access to space. However, there is a second pillar of their result that is either so obvious or so subtle as to be left implicit in the notion of "clock". The rising and setting of the sun provides a linearity to time. Indeed, it seems possible that their results might be relying deeply on this linearity—that is, on the lexicographic task execution order.

Note, in fact, that it is natural to consider the removal of their fixed, lexicographically ordered schedule requirement. For example, if the tasks exist in a distributed setting with some centralized control, then in the Cai–Furst model the transfer step between tasks is, e.g., "Task 10010110111 just ran and had output 011; task 10010111000, you're up." If task 10010111000 is located on a processor currently busy with other tasks, delay may result. With the ordering removed, the central controller merely seeks, and passes the current output bits to, any processor with an as yet unexecuted task. Indeed, the central controller need not even know the name of the exact task that is about to run. Thus, removing the linear execution order requirement both provides flexibility of execution order and potentially lessens the amount of information that must be exchanged.

So, both because we find relaxing the linear-order constraint natural and because we wish to determine the extent to which the Cai–Furst result depends on ordering (rather than merely depending on the "weak access to space" provided by the task number), this paper studies the class of languages accepted in the Cai–Furst model *when the*

* Supported in part by Grants NSF-INT-9116781/JSPS-ENGR-207, NSF-CCR-9322513, and NSF-INT-9513368/DAAD-315-PRO-fo-ab. Work done in part while visiting the Tokyo Institute of Technology. E-mail: lane@cs.rochester.edu.

† Supported in part by Grants NSF-INT-9116781/JSPS-ENGR-207, and NSF-CCR-9701911. E-mail: ogihara@cs.rochester.edu.

0022-0000/97 $25.00
Copyright © 1997 by Academic Press
All rights of reproduction in any form reserved.

order of execution is arbitrary. That is, we require that the same answer—whether to accept or reject—be reached regardless of the order of task execution. In particular (detailed definitions can be found in Section 2), if the tasks are allowed to pass on a k-ary value as their output, we call the class so accepted SSF_k. (Cai and Furst called their classes SF_k.[1] Our extra "S" stands for "symmetric," as our execution order can equally well be any permutation of the tasks.)

We note that $\mathrm{SF}_1 = \mathrm{SSF}_1 = \mathrm{P}$, so there is no difference at the trivial level. We completely characterize SSF_2 as the strongly disjoint unions of NP sets with $\oplus\mathrm{P}$ sets. On one hand, this indicates that, even without ordering and with just one (binary) bit of information passing between tasks, quite complex sets can be accepted via a large number of simple tasks. On the other hand, our characterization suggests that SSF_2 may be strictly weaker than SF_2; we display a relativized world A in which $\mathrm{SSF}_2^A \subsetneq \mathrm{SF}_2^A$. Although it would be nice to show "$\mathrm{SSF}_2 = \mathrm{SF}_2$ implies the collapse of the polynomial-time hierarchy," we note that showing this is even harder than resolving a central open question regarding the relationship between nondeterministic computation and parity computation. We provide an upper bound showing that, for each $k \geqslant 3$, all SSF_k sets can be accepted via a fixed amount of access to NP and generalized parity classes. It follows from our upper bound that there is a relativized world A in which $\mathrm{NP}^{\mathrm{NP}^A} \not\subseteq \bigcup_k \mathrm{SSF}_k^A$ and, thus, in this world, for each $k \geqslant 3$, $\mathrm{SSF}_k^A \subsetneq \mathrm{SF}_k^A$ and $\mathrm{PSPACE}^A \neq \bigcup_k \mathrm{SSF}_k^A$. This contrasts with the result of Cai and Furst that for each A, $\mathrm{PSPACE}^A = \mathrm{SF}_5^A = \mathrm{SF}_6^A = \cdots$.

Although these results raise the possibility that large numbers of flexibly schedulable tasks do not suffice even to accept sets from the second level of the polynomial hierarchy, we show that large numbers of flexibly schedulable *probabilistic* tasks *do* suffice to accept all sets from the polynomial hierarchy. In particular, we consider the case in which tasks are probabilistic, and in which the tokens passed between tasks are drawn from the set $\{0, 1\}$. Our acceptance criterion is Simon's exact counting criterion: the machine is said to accept if the probability that 0 is mapped to 1 equals $\frac{1}{2}$. As always, we require that the tasks be universally serializable: some task ordering leads to acceptance if and only if every task ordering leads to acceptance. We prove that the class of languages thus accepted is exactly $\mathrm{NP}^{\mathrm{PP}}$. Since Toda's Theorem [Tod91] implies that $\mathrm{NP}^{\mathrm{PP}} = \mathrm{NP}^{\mathrm{PP}^{\mathrm{PH}}}$, we see that the entire polynomial hierarchy (and more) can be accepted by flexibly scheduled probabilistic tasks passing only one bit of information between successive tasks.

[1] The notation used in [CF87] and [Ogi94] is off by one in the k subscript compared with the notation of [CF91]. We uniformly use the notation of the former two papers. That is, SF_k and SSF_k are about machines that pass k-ary output between tasks.

Thus, in summary, we give characterizations, upper bounds, and relativized evidence all suggesting that linearity of scheduling does play a central role in yielding the great power of the Cai–Furst classes, but that adding probabilism to the model recovers a substantial portion of this power. Section 2 presents definitions and preliminaries. Section 3 proves our results regarding universally serializable deterministic tasks, and Section 4 proves our results regarding universally serializable probabilistic tasks.

2. DEFINITIONS

All our sets are collections of strings over the alphabet $\Sigma = \{0, 1\}$. Let $\mathbf{N} = \{1, 2, ...\}$. We adopt the standard polynomial-time reducibility notions of Ladner, Lynch, and Selman [LLS75]. Definitions of classes or notions not defined in this section (e.g., P, NP, and coNP) can be found in any standard complexity theory textbook [Pap94, BC93]. For any nondeterministic Turing machine M and any $x \in \Sigma^*$, we denote by $acc_M(x)$ the number of accepting computation paths of M on input x.

Definition 2.1 [CH90]. For each $k \geqslant 2$, define: $L \in \mathrm{Mod}_k\mathrm{P}$ if and only if there is a nondeterministic polynomial-time Turing machine M such that, for each x, it holds that $x \in L \Leftrightarrow acc_M(x) \not\equiv 0$ (modulo k). As is standard, $\oplus\mathrm{P}$ will be used to represent $\mathrm{Mod}_2\mathrm{P}$, a class first studied by Papadimitriou and Zachos [PZ83], and Goldschlager and Parberry [GP86].

OptP [Kre88], informally stated, is the class of functions computing the maxima of multivalued NP functions.

Definition 2.2 [Kre88]. For a nondeterministic Turing transducer M, define $S_M(x)$ to be the set of all outputs of M on x. A function f belongs to OptP if there exist a polynomial time-bounded nondeterministic Turing transducer M and a function $\mu \in \{\max, \min\}$ such that for every x

$$f(x) = \begin{cases} \mu(S_M(x)), & \text{if } S_M(x) \neq \varnothing \\ \lambda, & \text{otherwise.} \end{cases}$$

Definition 2.3 [HH91]. $\oplus$OptP is the class of languages L such that for some $\oplus\mathrm{P}$ set L' and for some $f \in \mathrm{OptP}$, $(\forall x \in \Sigma^*)\ [x \in L \Leftrightarrow f(x) \in L']$.

It is known (see [HH91]) that this is equivalent to the class of languages accepted via a $\oplus\mathrm{P}$ machine given both the input and advice—depending on the input—from an OptP function.

Fix $k \geqslant 1$. Following Cai and Furst [CF91], consider a "machine" specified by (a) a task-name-length polynomial $t(\cdot)$, (b) a token set $\{0, 1, ..., k-1\}$, and (c) a polynomial-time program f: $\Sigma^* \times \mathbf{N} \times \{0, 1, ..., k-1\} \rightarrow \{0, 1, ..., k-1\}$.

Define, for any input x and any $\pi \in S_{2^{t(|x|)}}$, where S_i is the symmetric group on $\{1, ..., i\}$,

$$h_0(x, \pi) = 0,$$
$$h_i(x, \pi) = f(x, \pi(i), h_{i-1}(x, \pi)) \quad \text{for } i \in \{1, ..., 2^{t(|x|)}\},$$
$$output(x, \pi) = \begin{cases} accept, & \text{if } h_{2^{t(|x|)}}(x, \pi) = 1, \\ reject, & \text{otherwise.} \end{cases}$$

Again following Cai and Furst, for the special case $k = 1$ we alter the definition of *output* to define acceptance to mean that the program $f(x, \pi(2^{t(|x|)}), h_{2^{t(|x|)}-1})$ itself ends in some accepting state.

Definition 2.4 [CF91]. A language L is in SF_k ("safe storage via k tokens") if there exists a machine of the form described above (with k as the k above) such that, for each x, it holds that $x \in L \Leftrightarrow output(x, I) = accept$, where I is the identity permutation (on $\{1, ..., 2^{t(|x|)}\}$).

Definition 2.5. A language L is in SSF_k ("symmetric safe storage via k tokens") if there exists a machine of the form described above (with k as the k above) such that, for each x,

1. $(\forall \pi_1, \pi_2 \in S_{2^{t(|x|)}})[output(x, \pi_1) = output(x, \pi_2)]$, and
2. $x \in L \Leftrightarrow output(x, I) = accept$.

The following results are known about the SF_k classes. $\mathrm{SF}_1 = \mathrm{P}$ [CF91]. Ogihara has shown directly that $\mathrm{SF}_2 = \oplus\mathrm{OptP}$ [Ogi94]. Let NNT denote the nearly near-testable sets [HH91], the class of sets L having a polynomial-time computable function f that on each input x either states that $x \in L$, or that $x \notin L$, or that exactly one of x and the lexicographical predecessor of x is in L, or that not exactly one of x and the lexicographical predecessor of x is in L. Let $R^p_m(\mathrm{NNT})$ denote $\{L \mid (\exists A \in \mathrm{NNT})[L \leqslant^p_m A]\}$. Note that the four NNT options correspond exactly to the four possible input-output mappings allowed to an SF_2 task and the "map everything to zero" option allows one to lay SF_2 computations one after the other in an NNT set. So, $\mathrm{SF}_2 = R^p_m(\mathrm{NNT})$. Ogihara's SF_2 result and this observation about the close connection between SF_2 and NNT are in harmony, as it is known that $\oplus\mathrm{OptP} = R^p_m(\mathrm{NNT})$ [HH91]. Immediately, one can apply to SF_2 the various results of Hemaspaandra and Hoene [HH91]. For example, though $\mathrm{SF}_2 = R^p_m(\mathrm{NNT})$, nonetheless $\mathrm{SF}_2 = \mathrm{NNT}$ only if $\mathrm{P} = \mathrm{NP} = \mathrm{PP}^{\mathrm{PH}}$. Ogihara ([Ogi94]; see also [BS95]) has put upper and lower bounds on the complexity of SF_3 and SF_4. Recently, in a very interesting use of the notion of "query order" (see [HHH97, HHW] for an introduction to query order), Hertrampf [Her97] has completely characterized SF_3 and SF_4. Finally, as mentioned earlier, Cai and Furst [CF91], building on the work of Barrington [Bar89], proved that $\mathrm{PSPACE} = \mathrm{SF}_5 = \mathrm{SF}_6 = \cdots$, a result that, as described in Section 1, motivated this paper.

The intuition behind Definition 2.5 is that, as discussed in Section 1, we wish to study how crucial the task scheduling is in the serializable computation. The permutation π represents a task scheduling, that is, the order of the tasks, and for each i, $h_i(x, \pi)$ represents the output after the ith task has executed. Membership in SSF_k requires that the final acceptance or rejection behavior be independent of the order of task execution. Thus, we describe this as "universally serializable computation." We note that computation that, like SSF_k, is invariant (in some property) under a certain variety of changes is not a new notion in complexity theory. For example, in a quite different context, invariance under some perturbation is also required in the theory of "robust computation" (see the survey [Hem93]).

Note that, though both SF_k and SSF_k computations employ a large number of tasks, in both cases each individual task is allowed to run for just a short amount of time, and to pass on only a small amount of information to the next task.

For any complexity class $\mathscr{C}$, we denote by co$\mathscr{C}$ the class $\{L \mid \overline{L} \in \mathscr{C}\}$, where $\overline{L} = \Sigma^* - L$. For any set S, we denote by $\|S\|$ the cardinality of S. We assume that the integers have their standard binary encodings.

Let $\mathscr{C}_1$ and $\mathscr{C}_2$ be any classes. Following the convention of Gundermann, Nasser, and Wechsung [GNW90], we use $\mathscr{C}_1 \vee \mathscr{C}_2$ to denote $\{L \mid (\exists L_1 \in \mathscr{C}_1)(\exists L_2 \in \mathscr{C}_2)[L = L_1 \cup L_2]\}$, and we use $\mathscr{C}_1 \wedge \mathscr{C}_2$ to denote $\{L \mid (\exists L_1 \in \mathscr{C}_1)(\exists L_2 \in \mathscr{C}_2)[L = L_1 \cap L_2]\}$. In a slight abuse of notation, we similarly use $\mathscr{C}_1 - \mathscr{C}_2$ to denote $\{L \mid (\exists L_1 \in \mathscr{C}_1)(\exists L_2 \in \mathscr{C}_2)[L = L_1 \cap \overline{L_2}]\}$.

Definition 2.6. We say that $L \in \mathrm{NP} \overset{+}{\vee}_z \oplus\mathrm{P}$ ("L is the strongly disjoint union of an NP language and a $\oplus$P language") if there exist an NP language L_1 and a $\oplus$P language L_2, where M is a machine defining L_2 in the sense of the above definition, such that

1. $L = L_1 \cup L_2$,
2. $L_1 \cap L_2 = \emptyset$, and
3. $L_1 \subseteq \{x \mid acc_M(x) = 0\}$.

This says not only that L is the disjoint union of an NP set and a $\oplus$P set, but also that when an input is in the NP language, the $\oplus$P set rejects via having zero accepting paths. This special "zero rejects" condition is somewhat reminiscent of the ModZP classes of Beigel [Bei91], and thus our use of the letter "Z" in our notation.

Boolean hierarchies for a variety of classes including NP [CGH+88; CGH+89], UP [HR97], and R [BJY90] have been studied recently in complexity theory. For our purposes, it suffices to consider the most standard Boolean hierarchy, namely, that built over NP. This hierarchy contains exactly those sets that can be accepted via a finite tree of Boolean operations on NP sets.

Definition 2.7 [CGH$^+$88]. 1. $\mathrm{BH}(0) = \mathrm{P}$.

2. For each $k \geqslant 1$, $\mathrm{BH}(k) = \mathrm{NP} - \mathrm{BH}(k-1)$.

3. $\mathrm{BH} = \bigcup_{k \geqslant 0} \mathrm{BH}(k)$.

We defer to Section 4 the definition of universally serializable probabilistic machines.

3. UNIVERSALLY SERIALIZABLE COMPUTATION

Note that $\mathrm{SSF}_1 = \mathrm{SF}_1 = \mathrm{P}$ since no information can be passed on from a task to another if there is only one token. Note that for every $k \geqslant 1$, $\mathrm{SSF}_k \subseteq \mathrm{SF}_k$. Note also that as the SF_k are all contained in PSPACE, so are the SSF_k. What can we say about SSF_k, $k \geqslant 2$? In particular, for $k \geqslant 2$, does it hold that $\mathrm{SSF}_k = \mathrm{SF}_k$? Below, we completely characterize SSF_2 and upper bound SSF_k, $k \geqslant 3$, in such ways as to suggest that these SF and SSF classes differ.

Theorem 3.1. 1. $\mathrm{SSF}_2 = \mathrm{NP} \overset{+}{\vee}_z \oplus\mathrm{P}$.

2. *For each* $k \geqslant 3$, $\mathrm{SSF}_k \subseteq \{L \mid L \leqslant_m^{\mathrm{FP}^{\mathrm{NP}}_{(k^k+2)\text{-truth-table}}} \mathrm{coMod}_{k!}\mathrm{P}\}$.

Before proving the theorem we mention that, as an immediate corollary of Part 2 of Theorem 3.1, we can state the following more accessible upper bound on the complexity of the SSF_k classes. For each $k \geqslant 3$, $\mathrm{SSF}_k \subseteq \{L \mid L \leqslant_{\mathrm{btt}}^p \mathrm{NP} \cup \mathrm{Mod}_{k!}\mathrm{P}\}$, where $\leqslant_{\mathrm{btt}}^p$ denotes the standard notion of bounded truth-table reduction [LLS75].

Proof of Theorem 3.1. 1. First we prove that $\mathrm{SSF}_2 \subseteq \mathrm{NP} \overset{+}{\vee}_z \oplus\mathrm{P}$. Let $L \in \mathrm{SSF}_2$, and let f and t represent this in the sense of Section 2. Let

$$L_1 = \{x \mid (\exists j\colon 1 \leqslant j \leqslant 2^{t(|x|)})\,[f(x,j,0) = f(x,j,1) = 1]\},$$

$$L_2 = \{x \mid \|\{i \mid 1 \leqslant i \leqslant 2^{t(|x|)} \text{ and } f(x,i,0) = 1 \text{ and } f(x,i,1) = 0\}\| \equiv 1 \text{ (modulo 2)}\}.$$

Note that $L_1 \in \mathrm{NP}$ and $L_2 \in \oplus\mathrm{P}$. We claim $L = L_1 \cup L_2$, and that the union has exactly the strong disjointness property required to establish membership in $\mathrm{NP} \overset{+}{\vee}_z \oplus\mathrm{P}$. One can see this as follows.

The inclusion $L_1 \subseteq L$ follows from the fact that, if x belongs to L_1, then by picking a j such that $f(x,j,0) = f(x,j,1) = 1$ and a π such that $\pi(2^{t(|x|)}) = j$, we can force this specific task scheduling function to move the token from 0 to 1.

In order to prove the strong disjointedness, assume, by way of contradiction, that $x \in L_1 \cap L_2$. By the definition of L_1, there exists some j such that $f(x,j,0) = f(x,j,1) = 1$. Pick such a j. By the definition of L_2, the number of m such that $f(x,m,0) = 1$ and $f(x,m,1) = 0$ is odd, and thus, there exists such an m. Pick such an m. The numbers j and m are different by their respective definitions. Now for a permutation π with $\pi(2^{t(|x|)}) = j$, the token is delivered to 1 at the end. On the other hand, for a permutation π' such that $\pi'(2^{t(|x|)}) = m$ and $\pi'(2^{t(|x|)} - 1) = j$, the token is delivered to 0 at the end. Thus, we lose the universality that was assumed for this function f. Hence, x cannot be both in L_1 and in L_2.

We can see as follows that $L_2 \subseteq L$. Suppose $x \in L_2$. Let $q = \|\{i \mid 1 \leqslant i \leqslant 2^{t(|x|)} \text{ and } f(x,i,0) = 1 \text{ and } f(x,i,1) = 0\}\|$. Since q is odd, there exists some i, call it i', with this property. There is no ℓ, $1 \leqslant \ell \leqslant 2^{t(|x|)}$, such that both $f(x,\ell,0)$ and $f(x,\ell,1)$ equal 0. If such an ℓ were to exist, then $i' \neq \ell$, and thus, we can pick permutations π such that $\pi(2^{t(|x|)}) = \ell$ and π' such that $\pi'(2^{t(|x|)}) = i'$ and $\pi'(2^{t(|x|)} - 1) = \ell$. Here π would yield $x \notin L$, yet π' would yield $x \in L$, contradicting the robustness required by the definition of SSF_2. Also, note that $x \notin L_1$, as we know from above that $x \in L_1 \Rightarrow x \notin L_2$. Thus, if $x \in L_2$, it must hold that:

$$(\forall \ell\colon 1 \leqslant \ell \leqslant 2^{t(|x|)})\,[(f(x,\ell,0) = 0 \text{ and } f(x,\ell,1) = 1) \text{ or } (f(x,\ell,0) = 1 \text{ and } f(x,\ell,1) = 0)]. \tag{1}$$

Since q is odd, this implies $x \in L$. Thus, we have shown so far that $L_1 \cup L_2 \subseteq L$ and that L_1 and L_2 are strongly disjoint.

Finally, we show that $L \subseteq L_1 \cup L_2$. Let $x \in L - (L_1 \cup L_2)$. There is no ℓ, $1 \leqslant \ell \leqslant 2^{t(|x|)}$, such that $f(x,\ell,0) = f(x,\ell,1) = 0$; otherwise we would have $x \notin L$ by assuming $\pi(2^{t(|x|)}) = \ell$. Combining this with $x \in L_1$, we have (1), and in that case we have: $x \in L$ if and only if $x \in L_2$. But this contradicts our assumption that $x \in L - (L_1 \cup L_2)$.

We now prove $\mathrm{NP} \overset{+}{\vee}_z \oplus\mathrm{P} \subseteq \mathrm{SSF}_2$. Let $L \in \mathrm{NP} \overset{+}{\vee}_z \oplus\mathrm{P}$. Without loss of generality, let the NP language be $L_1 = \{x \mid (\exists j\colon 1 \leqslant j \leqslant 2^{q_1(|x|)})[R_1(x,j)]\}$, where $q_1(\cdot)$ is a polynomial and $R_1(\cdot,\cdot)$ is a polynomial-time computable predicate. Without loss of generality, let the $\oplus\mathrm{P}$ language be $L_2 = \{x \mid \|\{j \mid 1 \leqslant j \leqslant 2^{q_2(|x|)} \text{ and } R_2(x,j)\}\| \equiv 1 \text{ (modulo 2)}\}$, where $q_2(\cdot)$ is a polynomial and $R_2(\cdot,\cdot)$ is a polynomial-time computable predicate. Let $t(\cdot)$ be a polynomial such that $(\forall n)[2^{t(n)} \geqslant 2^{q_1(n)} + 2^{q_2(n)}]$. Define

$$f(x,i,0) = \begin{cases} 1, & \text{if } (i \leqslant 2^{q_1(|x|)} \text{ and } R_1(x,i)) \text{ or} \\ & (2^{q_1(|x|)} < i \leqslant 2^{q_1(|x|)} + 2^{q_2(|x|)} \text{ and} \\ & \quad R_2(x, i - 2^{q_1(|x|)})) \\ 0, & \text{otherwise,} \end{cases}$$

$$f(x,i,1) = \begin{cases} 0, & \text{if } 2^{q_1(|x|)} < i \leqslant 2^{q_1(|x|)} + 2^{q_2(|x|)} \text{ and} \\ & R_2(x, i - 2^{q_1(|x|)}) \\ 1, & \text{otherwise.} \end{cases}$$

Note that by the strong disjointedness of L_1 and L_2, and the above definitions, f and $t(\cdot)$ do meet the robustness condition required of the definition of SSF_2, and accept (in the sense of an SSF_2 machine) exactly $L_1 \cup L_2$.

(2) Let $L \in \mathrm{SSF}_k$, and let f and t represent this in the sense of Section 2. Suppose we wish to determine whether $x \in L$. Let $\mathscr{F}$ denote the set of all mappings of $\{0, ..., k-1\}$ to itself. For each i, $1 \leqslant i \leqslant 2^{t(|x|)}$, define $\varphi_i \in \mathscr{F}$ be such that:

$$\text{for each } a, b \in \{0, ..., k-1\},$$
$$\varphi_i(a) = b \text{ if and only if } f(x, i, a) = b.$$

Let $\sigma = [\alpha_1 \cdots \alpha_m]$ be a sequence of mappings in $\mathscr{F}$. We use $\bigodot_\sigma(z)$ to denote $\alpha_1(\alpha_2(\alpha_3(\cdots \alpha_m(z) \cdots)))$. For each $\alpha \in \mathscr{F}$, let *freq*(α) denote the number of i, $1 \leqslant i \leqslant 2^{t(|x|)}$, such that $\varphi_i = \alpha$.

Define $A = \{(x, \alpha, m) \mid m \leqslant k^2 + k$ and *freq*$(\alpha) \geqslant m\}$, keeping in mind that x is an implicit argument of *freq* and that we consider k to be fixed. Since m is a fixed constant, $A \in \mathrm{NP}$. Let us now return to considering x to be fixed (we used it as an argument in A as we require A to belong to NP uniformly over all x). For each $\alpha \in \mathscr{F}$, by using a binary search over $[0, k^2+k]$ with queries to A, we do the following:

- We first check whether *freq*$(\alpha) \geqslant k^2 + k$; and
- if *freq*$(\alpha) < k^2 + k$, then we compute the exact value of *freq*(α).

The total number of queries to A is at most

$$\lceil \log_2(1 + k^2 + k) \rceil \cdot \|\mathscr{F}\|. \tag{2}$$

Note that if $k \geqslant 3$, this number is less than or equal to k^{k+2}. Define

$$\sigma_0 = [\underbrace{\beta_1 \cdots \beta_1}_{freq(\beta_1)} \cdots \underbrace{\beta_m \cdots \beta_m}_{freq(\beta_m)}],$$

where $\beta_1, ..., \beta_m$ is an enumeration of all α with *freq*$(\alpha) < k^2 + k$. Define $i_0 = \bigodot_{\sigma_0}(0)$. Let $\mathscr{A}$ be the set of all mappings such that *freq*$(\alpha) \geqslant k^2 + k$. Let $\alpha_1, ..., \alpha_p$ be an enumeration of all $\alpha \in \mathscr{A}$. Consider the following condition:

There is a sequence σ of length at most k such that only mappings in $\mathscr{A}$ appear in σ and $\bigodot_\sigma(i_0) = 1$. (3)

We claim that if $x \in L$, then (3) holds. This is seen as follows. Suppose $x \in L$. Then there is a sequence σ satisfying (3) except for the length condition. In fact,

$$[\underbrace{\alpha_1 \cdots \alpha_1}_{freq(\alpha_1)} \cdots \underbrace{\alpha_p \cdots \alpha_p}_{freq(\alpha_p)}]$$

is one such sequence. If σ is of length strictly greater than k, then we can, by the pigeon-hole principle, partition σ onto three parts τ_1, τ_2, τ_3 such that $\sigma = \tau_1 \tau_2 \tau_3$, τ_2 is nonempty, and $\bigodot_{\tau_2\tau_3}(i_0) = \bigodot_{\tau_3}(i_0)$. Note that $\tau_1 \tau_3$ satisfies (3) except for the length condition and is shorter than σ. By repeating this process, we can construct a sequence satisfying (3). Taking the contrapositive, we may also state that if (3) does not hold, then $x \notin L$. Note that (3) is dependent only on $\mathscr{A}$. Note further that since $\mathscr{A}$ is a subset of the set of mappings from k tokens to k tokens, there is a finite upper bound —namely 2^{k^k}—on the number of possibilities for $\mathscr{A}$. So we may compute in time constant (in our fixed k, and in particular independent of the input x) a table, *table*, that for each of these possibilities lists whether (3) holds and, in cases where it does hold, lists one minimum-length σ satisfying (3). Thus, after computing i_0 and enumerating all $\alpha \in \mathscr{A}$, by using *table*, we can check whether (3) holds.

So, suppose (3) holds. Let σ_1 be one of the minimum-length sequences satisfying (3). This can be found by look-up in *table*. For each $\alpha \in \mathscr{F}$, define *freq*$_0(\alpha)$ to be the number of occurrences of α's in $\sigma_1\sigma_0$, and define *freq*$_1(\alpha) =$ *freq*$(\alpha) -$ *freq*$_0(\alpha)$. Note—since σ_1 is of length at most k and each $\alpha \in \mathscr{A}$ satisfies *freq*$(\alpha) \geqslant k^2 + k$—that for every $\alpha \in \mathscr{A}$ it holds that *freq*$_1(\alpha) \geqslant k^2$. For each $\alpha \in \mathscr{A}$, define *ind*(α) to be the smallest $i > 0$ such that $\alpha^i(1) = 1$ if such an i exists, and undefined otherwise. Note that for any α, if *ind*(α) is defined then it is at most k. We claim (under the supposition that (3) holds) that if $x \in L$, then for every $\alpha \in \mathscr{A}$, *ind*(α) is defined. This can be seen as follows. Let α be a member of $\mathscr{A}$. Let ξ represent the entire chain of $2^{t(|x|)}$ tasks run on our input x (recall we are assuming $x \in L$). Let ξ' denote ξ with enough instances of the mappings in $\sigma_1\sigma_0$ removed to form $\sigma_1\sigma_0$, and also with all *freq*$_1(\alpha) \geqslant k^2$ remaining instances of α removed. Consider $\bigodot_{\xi'\alpha^{freq_1(\alpha)}\sigma_1\sigma_0}(0)$. Since $x \in L$ and our computation is universally serializable, this equals 1. Consider the tokens output by each of the α's in the $\alpha^{freq_1(\alpha)}$ portion of this computation. Since there are at least k^2 such outputs and only k tokens, some token must be output more than once. Thus, there exists an integer h, $0 < h \leqslant k^2$, such that $\bigodot_{\alpha^{freq_1(\alpha)-h}}(1) = \bigodot_{\alpha^{freq_1(\alpha)}}(1)$, and in particular $\bigodot_{\alpha^{freq_1(\alpha)-h}\sigma_1\sigma_0}(0) = \bigodot_{\alpha^{freq_1(\alpha)}\sigma_1\sigma_0}(0)$. So $\bigodot_{\xi'\alpha^{freq_1(\alpha)-h}\sigma_1\sigma_0}(0) = \bigodot_{\xi'\alpha^{freq_1(\alpha)}\sigma_1\sigma_0}(0)$, but we already know that this is 1. However, since our computation is universally serializable it must hold that $\bigodot_{\xi'\alpha^{freq_1(\alpha)}\sigma_1\sigma_0}(0) = \bigodot_{\alpha^h\xi'\alpha^{freq_1(\alpha)-h}\sigma_1\sigma_0}(0)$. That is, $1 = \bigodot_{\alpha^h\xi'\alpha^{freq_1(\alpha)-h}\sigma_1\sigma_0}(0) = \bigodot_{\alpha^h}(1)$. So there does exist some $h > 0$ such that $\alpha^h(1) = 1$, and thus there is some minimum such value, and *ind*(α) is defined.

We claim (still under the supposition that (3) holds) that if $x \in L$, then the following condition is satisfied:

Let σ be a sequence consisting only of mapping in $\mathscr{A}$ and for every $\alpha \in \mathscr{A}$, α appears at most k^2 times in σ. If $\bigodot_\sigma(1) = 1$, then for any σ' constructed from σ by permuting the order of mappings, $\bigodot_{\sigma'}(1) = 1$. (4)

Our claim is established as follows. Let ξ be as above. Since each mapping in $\mathscr{A}$ appears in σ at most k^2 times, each $\alpha \in \mathscr{A}$ appears in σ less than $freq_1(\alpha)$ times. Thus, after removing from ξ the tasks making up $\sigma_1\sigma_0$, what is left of ξ retains the right mix of tasks to make up σ. Let ξ'' represent the mapping multiset (ordered in any way) remaining after removing from ξ the mappings (in their proper multiplicities) to form σ_1, σ_0, and σ. Note that (aa) $\odot_{\xi''\sigma\sigma_1\sigma_0}(0) = 1$ since $x \in L$. By assumption, $\odot_\sigma(1) = 1$, and we know from earlier that $\odot_{\sigma_1\sigma_0}(0) = 1$, so (bb) $\odot_{\sigma\sigma_1\sigma_0}(0) = 1$. From (aa) and (bb) it follows that $\odot_{\xi''}(1) = 1$. Let σ' be some permuted version of σ. Then since our computation is universally serializable, and from the above discussion, and the fact that $x \in L$, we have $1 = \odot_{\xi''\sigma\sigma_1\sigma_0}(0) = \odot_{\sigma\xi''\sigma_1\sigma_0}(0) = \odot_{\sigma'}(\odot_{\xi''}(\odot_{\sigma_1\sigma_0}(0))) = \odot_{\sigma'}(\odot_{\xi''}(1)) = \odot_{\sigma'}(1)$. Thus (4) holds. Taking the contrapositive, we may state that if either $ind(\alpha)$ is undefined for some $\alpha \in \mathscr{A}$ or (4) does not hold, then $x \notin L$. Similarly to the case with condition (3), since there are only a constant (for k fixed) number of possible values $\mathscr{A}$ can take on, whether or not condition (4) holds for a given $\mathscr{A}$ can be checked by simple look-up in a table of constant (in k) size.

So, suppose that $ind(\alpha)$ is defined for every $\alpha \in \mathscr{A}$ and that (4) holds. We are still also under the supposition that (3) holds. We claim that the k^2 upper bound in (4) can be eliminated; that is, we claim that the following condition is satisfied:

Let σ be a sequence consisting only of mappings in $\mathscr{A}$. If $\odot_\sigma(1) = 1$, then for any σ' constructed from σ by permuting the order of mappings, $\odot_{\sigma'}(1) = 1$. (5)

The proof is by contradiction. Suppose, to the contrary, that there exist two sequences, τ_1 and τ_2, each consisting only of mappings in $\mathscr{A}$, such that

for each i, $1 \leqslant i \leqslant p$, the number of α_i's appearing in τ_1 is the same as the number of α_i's appearing in τ_2, and $\odot_{\tau_1}(1) = 1 \neq \odot_{\tau_2}(1)$. (6)

Choose τ_1 and τ_2 so that the length of these sequences is the smallest amongst all sequence pairs satisfying the two conditions in (6). Since we are assuming that (4) holds, it must be the case that for some i, $1 \leqslant i \leqslant p$, α_i appears in τ_1 (and thus, in τ_2) more than k^2 times. Without loss of generality, assume that α_1 appears more than k^2 times. Let S be the set of all sequences τ constructed from τ_1 by permuting the order of mappings. Both τ_1 and τ_2 are members of S. Note that for any σ and σ' in S, σ' can be obtained from σ by repeatedly swapping the order of two consecutive mappings. Since $\odot_{\tau_1}(1) = 1 \neq \odot_{\tau_2}(1)$, there exist two sequences ρ_1 and ρ_2 in S and two mappings β and γ from $\mathscr{A}$ such that

1. $\odot_{\rho_1}(1) = 1 \neq \odot_{\rho_2}(1)$, and
2. for some sequences θ and ξ, $\rho_1 = \xi\beta\gamma\theta$ and $\rho_2 = \xi\gamma\beta\theta$.

Among all such ρ_1, ρ_2, β, and γ, pick a combination that minimizes the length of ξ (therefore, maximizes the length of θ).

Let m' denote the length of sequence θ. We claim that $m' < k$. Let $\theta = [c_{m'}, ..., c_1]$, with $(\forall i: 1 \leqslant i \leqslant m')[c_i \in \mathscr{A}]$ holding, and assume that $m' \geqslant k$. Then there exist i and j, $0 \leqslant i < j \leqslant k$, such that $c_j(c_{j-1}(\cdots(c_1(1))\cdots)) = c_i(c_{i-1}(\cdots(c_1(1))\cdots))$. Pick one such pair (i, j) and let $\theta' = [c_{m'}, ..., c_{j+1}, c_i, ..., c_1]$. Then $\odot_\theta(1) = \odot_{\theta'}(1)$. Let $\tau_1' = \xi\beta\gamma\theta'$ and $\tau_2' = \xi\gamma\beta\theta'$. The sequences τ_1' and τ_2' satisfy the two conditions in (6) and are shorter than τ_1 and τ_2, which contradicts the minimality of the length of τ_1 and τ_2. Thus, $m' < k$. Also note that $\beta \neq \gamma$, as otherwise ρ_1 and ρ_2 would be identical, and thus could not produce different images under the swapping of β and γ. Thus, we can conclude that the number of α_1's in $\beta\gamma\theta$ is at most $1 + |\theta| \leqslant 1 + (k-1) = k$. Let m be the number of α_1's in ξ. Then, since we have assumed that α_1 appears at least $k^2 + 1$ times in τ_1, we have

$$m \geqslant k^2 + 1 - k = k(k-1) + 1. \qquad (7)$$

As we have chosen ξ, θ, β, and γ so that ξ is of minimum length, for any ξ' constructed from ξ by permuting the order of mappings it holds that:

$$\odot_{\xi'\beta\gamma\theta}(1) = 1 \quad \text{and} \quad \odot_{\xi'\gamma\beta\theta}(1) \neq 1. \qquad (8)$$

The proofs of the two claims in (8) are similar, so we prove only the first, $\odot_{\xi'\beta\gamma\theta}(1) = 1$. The proof is by contradiction. Suppose, by way of contradiction, that for some ξ' constructed from ξ by permuting the order of the mappings we have $\odot_{\xi'\beta\gamma\theta}(1) \neq 1$. Recall, $\odot_{\xi\beta\gamma\theta}(1) = 1$. Consider some sequence of mapping sequences, each formed from the previous one by exchanges of two adjacent elements, that starts with ξ and ends with ξ'. Let ξ_2 be the first mapping sequence in this sequence such that $\odot_{\xi_2\beta\gamma\theta}(1) \neq 1$, and let ξ_1 be the mapping sequence just before ξ_2 (note that $\odot_{\xi_1\beta\gamma\theta}(1) = 1$). We may write $\xi_1 = uvwz$ and $\xi_2 = uwvz$, where v and w are elements of $\mathscr{A}$ and u and z are sequences of zero or more elements of $\mathscr{A}$. Note that we assumed above that θ was of the longest length possible. However, $z\beta\gamma\theta$ is an even longer sequence that has the properties required of "θ" (with u serving as the corresponding "ξ"), and this yields a contradiction. This concludes the proof by contradiction of (the first claim of) (8). So, let ξ_0 be the sequence constructed from ξ by simply taking out all m α_1's and let ξ_1 be the length m sequence consisting only of α_1's. Then we have

$$\odot_{\xi_1\xi_0\beta\gamma\theta}(1) = 1 \quad \text{and} \quad \odot_{\xi_1\xi_0\gamma\beta\theta}(1) \neq 1.$$

Let $d_1 = \odot_{\xi_0\beta\gamma\theta}(1)$ and $d_2 = \odot_{\xi_0\gamma\beta\theta}(1)$. Then we have

$$\alpha_1^m(d_1) = 1 \quad \text{and} \quad \alpha_1^m(d_2) \neq 1.$$

Since $m \geqslant k(k-1)+1$, there exist i and j, $0 \leqslant i < j \leqslant k$, such that $\alpha_1^j(d_1) = \alpha_1^i(d_1)$. Let (i_1, j_1) be the smallest such i and j and let $r_1 = j_1 - i_1$. Similarly, let (i_2, j_2) be the smallest i and j such that $0 \leqslant i < j \leqslant k$ and $\alpha_1^j(d_2) = \alpha_1^i(d_2)$ and let $r_2 = j_2 - i_2$. Let $\mathrm{lcm}(a, b)$ represent the least common multiple of a and b. We claim that

$$\text{both } i_1 + \mathrm{lcm}(r_1, r_2) \text{ and } i_2 + \mathrm{lcm}(r_1, r_2) \text{ are less than or equal to } k(k-1)+1. \tag{9}$$

(9) is seen as follows. First, consider the case that $r_1 = r_2 = k$. Then, since $j_1 = r_1 + i_1 \leqslant k$, we have $i_1 = 0$. Similarly, $i_2 = 0$. Thus, $i_1 + \mathrm{lcm}(r_1, r_2) = i_2 + \mathrm{lcm}(r_1, r_2) = k \leqslant k(k-1)+1$. Next, consider the case that $r_1 = k$ and $r_2 \leqslant k-1$. Again, we have $i_1 = 0$. Since $\mathrm{lcm}(r_1, r_2) \leqslant k(k-1)$, we have $i_1 + \mathrm{lcm}(r_1, r_2) \leqslant k(k-1) < k(k-1)+1$. Also, $i_2 + \mathrm{lcm}(r_1, r_2) \leqslant (k - r_2) + kr_2$. The right-hand side is maximized when r_2 is the largest. Since $r_2 \leqslant k-1$, we have $i_2 + \mathrm{lcm}(r_1, r_2) \leqslant k(k-1)+1$. The case $r_2 = k$ and $r_1 \leqslant k-1$ is symmetric to the previous one. Finally, consider the case that $r_1 \leqslant k-1$ and $r_2 \leqslant k-1$. We have $\mathrm{lcm}(r_1, r_2) \leqslant (k-1)^2$, $i_1 \leqslant k-1$, and $i_2 \leqslant k-1$. So we have $i_1 + \mathrm{lcm}(r_1, r_2) \leqslant k(k-1) < k(k-1)+1$ and $i_2 + \mathrm{lcm}(r_1, r_2) \leqslant k(k-1) < k(k-1)+1$. Thus, we have established claim (9).

Now let $m' = m - \mathrm{lcm}(r_1, r_2)$. Then, by (7) and (9), we have $m' \geqslant 0$, and since r_1 and r_2 are the lengths of the two cycles, we have $\alpha_1^{m'}(d_1) = \alpha_1^m(d_1)$ and $\alpha_1^{m'}(d_2) = \alpha_1^m(d_2)$. So, let ξ_2 be the sequence of m' many α_1's and let $\tau_1' = \xi_2\xi_0\beta\gamma\theta$ and $\tau_2' = \xi_2\xi_0\gamma\beta\theta$. Then these sequences are shorter than τ_1 and τ_2 and satisfy the two conditions in (6), which contradicts the minimality of the length of τ_1 and τ_2. This concludes our proof by contradiction of (5).

Let $\mathscr{S}$ be the set of sequences consisting only of mappings in $\mathscr{A}$. For each $\sigma \in \mathscr{S}$, let $\mu(\sigma)$ denote $(b_1, ..., b_p)$ such that for each i, $1 \leqslant i \leqslant p$, α_i appears in σ exactly b_i many times. Define K_0 to be the set of all i for which there exists a sequence $\sigma \in \mathscr{S}$ such that $\odot_\sigma(1) = i$. Let $i \in K_0$ be witnessed by a sequence σ_i with $\mu(\sigma_i) = (b_{i,1}, ..., b_{i,p})$. Choose $b'_{i,1}, ..., b'_{i,p}$ so that for every j, $1 \leqslant j \leqslant p$, $b_{i,j} + b'_{i,j}$ is a multiple of $ind(\alpha_j)$ and let σ_i' be a sequence such that $\mu(\sigma_i') = (b'_{i,1}, ..., b'_{i,p})$. There exist some integers $c_{i,1}, ..., c_{i,p}$ such that $\mu(\sigma_i'\sigma_i) = (c_{i,1}\, ind(\alpha_1), ..., c_{i,p}\, ind(\alpha_p))$. So, for some permutation, the mappings in $\sigma_i'\sigma_i$ take token 1 to 1. Then, by (5), no matter what ordering is used, the mappings in $\sigma_i'\sigma_i$ take 1 to 1. So, $\odot_{\sigma_i'}(\odot_{\sigma_i}(1)) = 1$, and so $\odot_{\sigma_i'}(i) = 1$. Thus,

$$\text{for each } i \in K_0 \text{, there exists a sequence } \sigma_i \in \mathscr{S} \text{ such that } \odot_{\sigma_i}(i) = 1. \tag{10}$$

For $\sigma, \sigma' \in \mathscr{S}$, write $\sigma =_c \sigma'$ if $\mu(\sigma) = \mu(\sigma')$. For $i, j \in K_0$, write $i \equiv j$ if there exist $\sigma, \sigma' \in \mathscr{S}$ such that $\sigma =_c \sigma'$, $\odot_\sigma(1) = i$, and $\odot_{\sigma'}(1) = j$. The relation $\equiv$ is reflexive by the definition of K_0 and symmetric. We claim that $\equiv$ is transitive. This is seen as follows: Suppose $i_1 \equiv i_2$ and $i_2 \equiv i_3$ and this is witnessed by sequences σ and σ' such that $\sigma =_c \sigma'$ and sequences τ and τ' such that $\tau =_c \tau'$, respectively; that is, $\odot_\sigma(1) = i_1$, $\odot_{\sigma'}(1) = i_2$, $\odot_\tau(1) = i_2$, and $\odot_{\tau'}(1) = i_3$. Let θ be a sequence such that $\odot_\theta(i_2) = 1$. Then $\odot_{\theta\sigma'}(1) = 1$ and $\odot_{\theta\tau}(1) = 1$. So, by (5), we have $\odot_{\theta\sigma}(1) = 1$ and $\odot_{\theta\tau'}(1) = 1$. Now let $\xi = \sigma\theta\tau$ and $\xi' = \tau'\theta\sigma$. Then $\xi =_c \xi'$, $\odot_\xi(1) = i_1$, and $\odot_{\xi'}(1) = i_3$. Thus, $i_1 \equiv i_3$. Hence, $\equiv$ is an equivalence relation over K_0. Let $\mathscr{W} = \{W_1, ..., W_d\}$ be the equivalence classes over K_0, induced by $\equiv$. Without loss of generality, assume that 1 is in W_1. Then, by (5), W_1 consists only of 1. For each sequence $\sigma \in \mathscr{S}$ and each i, $1 \leqslant i \leqslant d$, if $\odot_\sigma(1) \in W_i$, then for each σ' such that $\sigma' =_c \sigma$, we have $\odot_{\sigma'}(1) \in W_i$; of course, for each σ there is a unique i such that $\odot_\sigma(1) \in W_i$. More precisely, for any $(b_1, ..., b_p)$, there exists a unique i such that for any σ with $\mu(\sigma) = (b_1, ..., b_p)$, $\odot_\sigma(1) \in W_i$.

For each i, let $\alpha(W_i) =_{\text{def}} \{\alpha(w) \mid w \in W_i\}$. Moreover, for any $\alpha \in \mathscr{A}$ and any i and j, $1 \leqslant i < j \leqslant d$, it holds that $\alpha(W_i) \cap \alpha(W_j) = \emptyset$. This is proven by contradiction. Assume for some $\alpha \in \mathscr{A}$ and some i and j, $1 \leqslant i < j \leqslant d$, that $\ell \in \alpha(W_i) \cap \alpha(W_j)$. Let b_i be an element of W_i such that $\alpha(b_i) = \ell$, and let b_j be an element of W_j such that $\alpha(b_j) = \ell$. Let θ be a sequence that takes ℓ to 1. (Such a sequence exists as some element of $\mathscr{S}$ takes 1 to i as $i \in K_0$, so as $\alpha(i) = \ell$, some sequence $\mathscr{S}$ takes 1 to ℓ, and so $\ell \in K_0$, and so by (10) there is an element of $\mathscr{S}$ mapping ℓ to 1.) Let τ_1 and τ_2 be sequences that take 1 to b_i and b_j, respectively. Define $\xi_1 = \tau_1\theta\alpha\tau_2$ and $\xi_2 = \tau_2\theta\alpha\tau_1$. Then $\odot_{\xi_1}(1) = b_i$ and $\odot_{\xi_2}(1) = b_j$. Since $\xi_1 =_c \xi_2$, we have $b_i \equiv b_j$, which contradicts our assumption that W_i and W_j are distinct equivalence classes induced by $\equiv$. Thus, the claim holds. We can conclude from this that each sequence $\sigma \in \mathscr{S}$ acts as a bijection of $\mathscr{W}$ to $\mathscr{W}$.

Now for each $\sigma, \sigma' \in \mathscr{S}$, define $\sigma \equiv \sigma'$ if for every i, $1 \leqslant i \leqslant d$, the image of W_i by σ is the same as that of σ'. Then, $\mathscr{S}/\equiv$ forms an Abelian group acting on $\mathscr{W}$. Note for any $\sigma \in \mathscr{S}/\equiv$, the order of σ is a divisor of $ind(\alpha_1) \cdots ind(\alpha_p)$. So the order of the Abelian group is a product of primes each no greater than k. It is well-known that any finite Abelian group is a direct product of cyclic groups of prime power orders (and that these orders and their multiplicities are unique and determine the group up to isomorphism) [Jac74, Theorem 3.13]. So there exist integer e, cyclic groups $G_1, ..., G_e$—with generators $g_1, ..., g_e$ and prime power orders $r_1, ..., r_e$—such that $\mathscr{S}/\equiv$ can be viewed as the direct product $G_1 \times \cdots \times G_e$ and such that $r_1, ..., r_e$ are powers of primes $\leqslant k$. In particular, for each i, $1 \leqslant i \leqslant p$, the action of α_i on $\mathscr{W}$ is represented as $(g_1^{E(i,1)}, ..., g_e^{E(i,e)})$ for some $E(i,1), ..., E(i,e)$. So, given a sequence σ with $\mu(\sigma) = (b_1, ..., b_p)$, $\odot_\sigma(1) = 1$ if and only if for every j, $1 \leqslant j \leqslant e$, $\sum_{i=1}^p E(i,j) \cdot b_i$ is a multiple of r_j. Thus, $x \in L$ if and only if for every j, $1 \leqslant j \leqslant e$, $\sum_{i=1}^{p} E(i,j) \cdot freq_1(\alpha_i)$ is a multiple of r_j. Since $r_j \leqslant k$ for all j, it holds that, for each j, whether $\sum_{i=1}^{p} E(i,j) \cdot freq_1(\alpha_i)$ is a multiple of r_j can be tested by conjunctive queries to at most e of the classes $\mathrm{MOD}_q\mathrm{P}$ with q being a prime power no greater than k. It is known that $\mathrm{MOD}_{q^h}\mathrm{P} = \mathrm{MOD}_q\mathrm{P}$ for any prime q and $h \geqslant 1$ [BGH90]. So whether $\sum_{i=1}^{p} E(i,j) \cdot freq_1(\alpha_i)$ is a multiple of r_j can be tested by conjunctive queries to at most e of the classes $\mathrm{MOD}_q\mathrm{P}$ with q being prime and no greater than k. Now, whether for every j, $1 \leqslant j \leqslant e$, $\sum_{i=1}^{p} E(i,j) \cdot freq_1(\alpha_i)$ is a multiple of r_j can be tested by conjunctive queries to at most e of the classes $\mathrm{MOD}_q\mathrm{P}$ with q being prime and no greater than k. Since $\mathrm{MOD}_q\mathrm{P}$ with q prime is closed under $\leqslant_T^p$-reductions [PZ83, BGH90], we can certainly test whether $x \in L$ by conjunction of at most one query to each of the classes $\mathrm{MOD}_q\mathrm{P}$ with q prime and at most k. So, this can be tested by just one query to $\mathrm{coMOD}_{k!}\mathrm{P}$. This proves the theorem. ▮

We note in passing that the approximation made just after (2) was quite conservative. In fact, from (2) the k^{k+2} in the theorem statement can be replaced, e.g., by $\mathcal{O}(k^k \log k)$, and this comment applies equally well to the result mentioned in the text immediately after the proof of Theorem 3.6.

COROLLARY 3.2. $\mathrm{SSF}_2 \subseteq \mathrm{NP} \vee \oplus\mathrm{P}$.

What does Theorem 3.1 tell us about the relationship between SF_k and SSF_k? Recall that SF_2 equals $R_m^p(\mathrm{NNT})$, the many-one reducibility closure of the near-testable sets. So, $\mathrm{SSF}_2 \neq \mathrm{SF}_2$ unless every coNP language (indeed, every language in the Boolean hierarchy, or indeed in $R_m^p(\mathrm{NNT})$) is the union of an NP and a $\oplus$P set. Part 1 of Theorem 3.3 notes that there is a relativized world in which $\mathrm{SSF}_2 \neq \mathrm{SF}_2$, via this reasoning. Similarly (as $\mathrm{SF}_3 \supseteq \Sigma_2^p$), we have that unless $\Sigma_2^p \subseteq \bigcup_k \mathrm{SSF}_k$, for all $k \geqslant 3$, $\mathrm{SSF}_k \neq \mathrm{SF}_k$. Part 2 of Theorem 3.3 shows that indeed there is, via the same reasoning, a relativized world in which, for all $k \geqslant 3$, $\mathrm{SSF}_k \neq \mathrm{SF}_k$.

THEOREM 3.3. 1. $(\exists A)[\mathrm{coNP}^A \not\subseteq (\mathrm{NP} \vee \oplus\mathrm{P})^A]$. *Thus, in particular,* $\mathrm{SSF}_2^A \neq \mathrm{SF}_2^A$.

2. $(\exists A)[\mathrm{NP}^{\mathrm{NP}^A} \not\subseteq \bigcup_k \mathrm{SSF}_k^A]$. *Thus, in particular, for each* $k \geqslant 3$, $\mathrm{SF}_k^A \neq \mathrm{SSF}_k^A$, *and* $\bigcup_k \mathrm{SSF}_k^A \subsetneq \mathrm{PSPACE}^A$.

In fact, the second part of the theorem even implies that $\mathrm{SF}_3^A \not\subseteq \bigcup_k \mathrm{SSF}_k^A$. The proof of Theorem 3.3 relies on Torán's [Tor88; Tor91] basic work, revisited and extended by Beigel [Bei91], on the relativized relationship between nondeterministic computation and parity-like computation. The proof also relies on the fact that $\mathrm{SF}_2 = R_m^p(\mathrm{NNT})$, $\Sigma_2^p \subseteq \mathrm{SF}_3$, and Theorem 3.1 all relativize.

Proof. (1) We wish to show $(\exists A)[\mathrm{NP}^A \not\subseteq \mathrm{coNP}^A \wedge \oplus\mathrm{P}^A]$, which is equivalent to Part 1 of the theorem. We merely sketch the appropriate modification of Torán's proof [Tor91, Theorem 5.9] that $(\exists B)[\mathrm{NP}^B \not\subseteq \oplus\mathrm{P}^B]$. His proof is a stage construction. The ith stage diagonalizes to ensure that the (tally) language accepted by a certain nondeterministic polynomial-time oracle Turing machine is not the same as the language accepted by the ith nondeterministic polynomial-time oracle Turing machine viewed as a $\oplus$P machine. We use the same test language as Torán. We ensure such wide spacing between the lengths at which the stage diagonalizations take place that every oracle string queried at stage $i-1$ is shorter than the stage i input and that no machine we are diagonalizing against at stage i can, when operating on our stage i diagonalization string, 0^i, query on a single path all length i oracle strings. To ensure this, we may also have to "slow down" the enumerations of coNP and $\oplus$P machines about to be mentioned. At our stage i, we interpret i as a pair, e.g., $i = \langle j, k \rangle$, encoding the jth coNP machine and the kth $\oplus$P machine, from some fixed suitable enumeration of such machines. We seek to diagonalize to ensure that our test language differs from the intersection of the languages accepted by these two machines. First, we ask whether there is any extension of the current oracle that will cause the coNP machine to reject. If so, freeze one rejecting path of the coNP machine (and any new strings added to the oracle along that path), and then add a string not on that path to our oracle in such a way as to make the test input accept. If not, then no action taken later during the stage can possibly cause the coNP machine to reject, so if we proceed with Torán's procedure at this stage, his diagonalization against the $\oplus$P machine will have no effect on the coNP machine's acceptance, and thus by diagonalizing against the $\oplus$P machine via Torán's procedure, we in this case also diagonalize against the language accepted by the intersection of the coNP machine and the $\oplus$P machine.

(2) The proof is based on the construction in [Bei91] of an oracle A relative to which $\mathrm{NP} \not\subseteq \mathrm{coMOD}_k\mathrm{P}$. Define

$$L(A) = \{0^n \mid (\exists x\colon |x| = n)(\forall y\colon |y| = n)[xy \in A]\}.$$

Then $L(A) \in \mathrm{NP}^{\mathrm{NP}^A}$. We will construct A so that $L(A)$ is not in SSF_k for any $k \geqslant 3$. Let $F_1, F_2, \ldots$ be an enumeration of all polynomial time-bounded oracle Turing transducers and let $M_1, M_2, \ldots$ be an enumeration of all polynomial-time nondeterministic Turing machines. We assume that F_i and M_i both run in time $p_i(n) = n^i + i$. We construct A in stages. At stage $s = \langle i, j, k, l \rangle$, we extend A so that the following requirement is fulfilled:

If F_i makes at most k^{k+2} queries to its oracle for every input, then there is some n such that $0^n \in A$ if and only if relative to A, $F_i^{L(M_j)}(0^n)$ outputs a string y such that $acc_{M_l}(y)$ is not a multiple of $k!$. (11)

Let A' be the set of all strings put into A prior to stage s. At stage s, we do the following:

We first choose n so that no strings of length $2n$ have been touched so far and $2^{\sqrt{n}} > 2^{k^{k+2}} p_i(n)\, p_j(p_i(n))$. We first check whether the following condition holds:

$$\text{There is a subset } S \text{ of } \Sigma^{2n} \text{ such that with oracle } A' \cup S,\ F_i(0^n) \text{ makes more than } k^{k+2} \text{ queries.} \tag{12}$$

If (12) holds, then we let $A = A' \cup S$, thereby fulfilling (11). So, suppose (12) does not hold. We then freeze computation of F_i on 0^n in phases 1, ..., $m \leqslant k^{k+2}$ as follows.

Phase t. Freeze computation of F_i's tth query. If F_i will make no more queries, then quit the loop. Otherwise, let q_t be the tth query along with the frozen computation. If A can be extended so that M_j accepts q_t, then freeze one accepting computation path of M_j on q_t. Proceed to Phase $t+1$.

Let w be the output of $F_i(0^n)$ with the oracle we have just constructed. Note that the number of strings whose membership is fixed in this process is less than $2^{\sqrt{n}}$. So there are at least $2^n - 2^{\sqrt{n}}$ many x's such that for every y, $|y| = n$, xy has not been touched. Let $K = k! - 1$ and $D = p_j(p_i(n)) + 1$, and choose KD such x's. Let $x_0, ..., x_{KD-1}$ be an enumeration of the chosen x's. For each i, $0 \leqslant i \leqslant KD - 1$, put every $x_i y$ with $y \in \Sigma^n - \{0^n\}$ into A. Let U be the family of all sets $S \subseteq \{x_0 0^n, ..., x_{k!d-1} 0^n\}$ such that for every i, $0 \leqslant i \leqslant d$, S contains at most one $x_{Ki+j} 0^n$, $0 \leqslant j \leqslant K-1$. There are $(K+1)^D = (k!)^D$ possible S's. Let B_0 be the set of strings put into A so far in this stage and $B = A' \cup B_0$. Note for any set $S \in U$, that $F_i^{B \cup S}$ outputs w. We claim that

$$\text{There is some set } S \in U \text{ such that } S = \varnothing \text{ if and only if } acc_{M_l}^{B \cup S}(0^n) \text{ is not a multiple of } k!. \tag{13}$$

Suppose, by way of contradiction, that (13) does not hold; that is, for every $S \in U$, $S = \varnothing$ if and only if $acc_{M_l}^{A_0 \cup S}(0^n)$ is a multiple of $k!$. Consider the sum of all accepting computation paths of M_l on 0^n for some oracle $A_0 \cup S$ with $S \in U$. By our assumption, the sum is $((K+1)^D - 1$ many multiples of $k')$ plus (exactly one nonmultiple of $k!$), so it is not a multiple of $k!$. Now let us look at one accepting computation path. Since M_l queries at most $D-1$ strings, for some i, $x_{Ki}, ..., x_{Ki+K-1}$, are not queried at all along the path. Thus, the computation path contributes to the sum a multiple of K, and so the grand total should be a multiple of K, yielding a contradiction. Thus, (13) always holds. Choose one S satisfying (13) and put all strings in S to A in order to establish (11). This proves part (2) of the theorem. ▌

Regarding Part 1 of Theorem 3.3, one might well hope not for a relativized result, but rather for a structural consequence. For example, can one prove

$$\mathrm{SSF}_2 = \mathrm{SF}_2 \Rightarrow \mathrm{PH} \text{ collapses?} \tag{14}$$

Note, however, that (since $\mathrm{NP} \subseteq \oplus\mathrm{P} \Rightarrow \oplus\mathrm{OptP} = \oplus\mathrm{P} = \mathrm{NP} \overset{+}{\forall}_z \oplus\mathrm{P}$, as OptP is in $\mathrm{FP}^{\mathrm{NP}}$ and $\oplus\mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$ [PZ83]):

Proposition 3.4. $\mathrm{NP} \subseteq \oplus\mathrm{P} \Rightarrow \mathrm{SF}_2 = \mathrm{SSF}_2$.

But this says that even easier than establishing (14) would be establishing

$$\mathrm{NP} \subseteq \oplus\mathrm{P} \Rightarrow \mathrm{PH} \text{ collapses.} \tag{15}$$

However (15) itself is very open. In fact, although $\mathrm{NP} \subseteq \oplus\mathrm{P}$ trivially implies a few consequences (e.g., $\oplus\mathrm{P}^{\mathrm{NP}} \subseteq \oplus\mathrm{P}$), it is a notorious open problem to prove from this assumption that the entire polynomial hierarchy falls within $\oplus\mathrm{P}$, the seemingly helpful known fact that $\oplus\mathrm{P}^{\oplus\mathrm{P}} = \oplus\mathrm{P}$ notwithstanding (see [Hem94]; the problem is that it is not at all clear that $\mathrm{NP} \subseteq \oplus\mathrm{P} \Rightarrow \mathrm{NP}^{\mathrm{NP}} \subseteq \oplus\mathrm{P}^{\oplus\mathrm{P}}$).

Part 2 of Theorem 3.3 displayed a relativized world in which the second level of the polynomial hierarchy was not contained in the SSF hierarchy. In fact, we conjecture that the weakness of the SSF classes is far more dramatic. In particular, we conjecture that the third level of the Boolean hierarchy is not contained in the SSF hierarchy, that is, we conjecture that $\mathrm{BH}(3) \nsubseteq \bigcup_k \mathrm{SSF}_k$. If this conjecture is true, then it is tight, as it is not hard to see that $\mathrm{DP} =_{\mathrm{def}} \mathrm{BH}(2) \subseteq \mathrm{SSF}_3$. However, note that there is something quite curious in our conjecture, given the result of Cai and Furst that $\mathrm{BH} \subseteq \mathrm{SF}_2$. In fact, the entire Boolean hierarchy is contained in $\bigcup_k \mathrm{SSF}_k$, given a certain change in the acceptance notion used to define of the SSF_k classes. We conclude this section with a brief discussion of alternate acceptance notions for the SSF classes.

Our definition of SSF_k adopts the basic acceptance mechanism—acceptance if some particular token is the final output—of Cai and Furst. However, the situation is a bit murkier than that would suggest. In particular, it is not hard to see that their definition is unchanged under a wide variety of acceptance mechanisms, including the following three mechanisms, each defined by redefining the function *output* from Section 2. Recall by way of comparison that our standard definition of *output* is

$$output(x, \pi) = \begin{cases} accept, & \text{if } h_{2^{t(|x|)}}(x, \pi) = 1, \\ reject, & \text{otherwise.} \end{cases}$$

Definition 3.5. 1. Acceptance mechanism ZERO (Z):

$$output(x, \pi) = \begin{cases} accept, & \text{if } h_{2^{t(|x|)}}(x, \pi) = 0, \\ reject, & \text{otherwise.} \end{cases}$$

2. Acceptance mechanism FIXED INTERPRETATION (FI): For some set $A \subseteq \{0, ..., k-1\}$ depending on L but not on x,

$$output(x, \pi) = \begin{cases} accept, & \text{if } h_{2^{t(|x|)}}(x, \pi) \in A, \\ reject, & \text{otherwise.} \end{cases}$$

3. Acceptance mechanism POLYNOMIAL-TIME INTERPRETATION (PI): For some polynomial-time function $g\colon \Sigma^* \times \{0, ..., k-1\} \to \{accept, reject\}$,

$$output(x, \pi) = g(x, h_{2^{t(|x|)}}(x, \pi)).$$

4. Let $MECH \in \{Z, FI, PI\}$. For each $k \geqslant 2$, ${}_{MECH}\mathrm{SSF}_k$ (respectively, ${}_{MECH}\mathrm{SF}_k$) denotes SSF_k (respectively, SF_k) redefined with the acceptance mechanism *MECH*. For $k=1$, we retain the special $k=1$ acceptance mechanism described in Section 2.

Informally, Z changes the mechanism to accept on the token 0 rather than the token 1, FI uses a fixed table based on the token, and PI allows a polynomial-time function to decide the outcome, given as its input both the input string and the final output token. Note, for each k, $\mathrm{SF}_k = {}_Z\mathrm{SF}_k = {}_{FI}\mathrm{SF}_k = {}_{PI}\mathrm{SF}_k$. For our symmetric safe storage classes, we have, for each k, $\mathrm{SSF}_k \subseteq {}_{FI}\mathrm{SF}_k \subseteq {}_{PI}\mathrm{SF}_k$. In contrast with $\mathrm{SF}_k = {}_Z\mathrm{SF}_k$, it is not clear that $\mathrm{SSF}_2 \subseteq \bigcup_k {}_Z\mathrm{SSF}_k$. On the other hand, it is immediate that $(\forall k)\ [{}_Z\mathrm{SSF}_k \subseteq \mathrm{SSF}_{k+1}]$. Note also that ${}_Z\mathrm{SSF}_2 = \mathrm{coSSF}_2$, ${}_{FI}\mathrm{SSF}_2 = \mathrm{SSF}_2 \cup \mathrm{coSSF}_2$, and ${}_{PI}\mathrm{SSF}_2 = R^p_{1\text{-truth-table}}(\mathrm{SSF}_2)$.

Regarding the other models, we note below that FI acceptance allows the Boolean hierarchy to be accepted via symmetric safe storage.

THEOREM 3.6. $(\forall k)\ [\mathrm{BH}(k) \cup \mathrm{coBH}(k) \subseteq {}_{FI}\mathrm{SSF}_{k+1}]$.

Proof. Each set in BH(k) can be expressed as $L_1 - (L_2 - (\cdots (L_{k-1} - L_k) \cdots))$, with $L_1 \supseteq L_2 \supseteq \cdots \supseteq L_{k-1} \subseteq L_k$ [CGH+88]. Our SSF machine keeps track of the largest k for which some witness has been seen certifying that the input is in L_k. The zero token is used to mean that for no k has a certificate been seen. Our fixed-interpretation acceptance set is $A = \{1, 3, ...\}$. Thus, $\mathrm{BH}(k) \subseteq {}_{FI}\mathrm{SSF}_{k+1}$. ${}_{FI}\mathrm{SSF}_{k+1}$ is closed under complementation, thus it also contains coBH(k). ∎

In fact, even the most general notion yields classes whose complexity is bounded above by the combined power of polynomial-time nondeterministic and parity-like computation. That is, one can also prove:

$$(\forall k \geqslant 3)\ [{}_{PI}\mathrm{SSF}_k \subseteq \{L \mid L \leqslant^{\mathrm{FP}^{\mathrm{NP}}_{(k^{k+2})\text{-truth-table}}}_{k^{k^k}\text{-disjunctive-truth-table}} \mathrm{coMOD}_{k!}\mathrm{P}\}$$

(the proof can be found in [HO96]). We conjecture that $\mathrm{SSF}_k \not\supseteq \mathrm{coMOD}_{k+1}\mathrm{P}$, although we concede that the upper bound provided by the result just mentioned is not sufficiently tight to provide structural evidence in support of our conjecture. Note that our conjecture would imply that the SSF hierarchy does not collapse: $\mathrm{SSF}_1 \subsetneq \mathrm{SSF}_2 \subsetneq \cdots$. Can one find an upper bound on SSF_k sufficiently tight as to support our conjecture? Or, keeping in mind that the SF hierarchy does collapse, can one prove that the SSF hierarchy also collapses?

Finally, we mention the open issue of closure under complementation and under Boolean operations. Since $\mathrm{SSF}_1 = \mathrm{SF}_1 = \mathrm{P}$ and $\mathrm{SF}_k = \mathrm{PSPACE}$ for all $k \geqslant 5$, these classes are closed under all Boolean operations. Since $\mathrm{SF}_2 = \oplus\mathrm{OptP}$ and $\oplus\mathrm{OptP}$ is closed under all Boolean operations [HH91], so is SF_2. Also, SF_3 and SF_4 are both closed under complementation [Ogi94]—we have only to append a new task at the end, which moves tokens at position 1 to position 0 and tokens at any position $\neq 1$ to 1. Furthermore, the ${}_{FI}\mathrm{SSF}_k$ classes and the ${}_{PI}\mathrm{SSF}_k$ classes are closed under complementation—for the former classes, take the complement of the finite interpretation set A and for the latter, swap *accept* and *reject*. On the other hand, from our characterization of SSF_2, SSF_2 seems unlikely to be closed under complementation or other Boolean operations (even union, as the union operation could destroy the strong-disjointedness property). In fact, note that Part 1 of Theorem 3.3 implies that there is a relativized world in which SSF_2 is not closed under complementation. As ${}_Z\mathrm{SSF}_2 = \mathrm{coSSF}_2$, there thus is also a relativized world in which ${}_Z\mathrm{SSF}_2$ is not closed under complementation. Can one resolve the remaining issues regarding closure under complementation and other Boolean operations? For example, can one prove that SF_3, SF_4, and SSF_k, $k \geqslant 3$, are closed under Boolean operations? Very recently, Hertrampf [Her97] has shown that SF_3 and SF_4 are indeed closed under Boolean operations, thus answering two parts of this question.

4. UNIVERSALLY SERIALIZABLE PROBABILISTIC COMPUTATION

As discussed earlier, Cai and Furst [CF91] showed that all languages in PSPACE can be accepted via many simple tasks scheduled in a particular way. The previous section proves upper bounds suggesting that if one removes the fixed scheduling (that is, if one requires the tasks to be universally serializable), then the class of languages so accepted may not even contain $\mathrm{NP}^{\mathrm{NP}}$. In this section, we consider the case in which the universally serialized tasks are probabilistic polynomial-time tasks, rather than deterministic polynomial-time tasks, and we prove that this suffices to recapture the polynomial hierarchy.

To be more specific, consider SSF_2 with the following changes. Each task is allowed to be a probabilistic polynomial-time Turing machine. We will say that the machine accepts if the probability that 0 is mapped to 1 (in the probabilistic mapping created by the running of all tasks in sequence) is exactly $\frac{1}{2}$. We require the scheduling to be universally serializable in the sense that the machine must accept on all scheduling orders if and only if it accepts on some scheduling order. Denote the class of languages so accepted by $\text{ProbabilisticSSF}_2$.

We will prove that $\text{ProbabilisticSSF}_2 = \text{NP}^{\text{PP}}$. This class is relatively robust, as it is well-known that $\text{NP}^{\text{PP}} = \text{NP}^{\text{C}_=\text{P}} = \text{NP}^{\#\text{P}}$ [Tor91]. Furthermore, it follows from Toda's Theorem [Tod91] that this class is relatively inclusive, as $\text{NP}^{\text{PP}^{\text{PH}}} = \text{NP}^{\text{PP}}$.

Of course, just as Cai and Furst drew on Barrington's [Bar89] work on branching programs, one might hope that here we can draw on work on probabilistic branching programs. Indeed, probabilistic branching programs have been studied by Kilian [Kil90]. However, we cannot obtain our desired results from this work for two reasons. First, we require universal serializability, and his work is in the (analog of the) "fixed schedule" model. Second, Kilian's approach to probabilistic computation is somewhat different than our approach; Kilian considers inserting, randomly, mappings among the tokens.

We note that our acceptance notion is not original to this paper, though ours is the first use of the mechanism outside of the setting of polynomially depth-bounded computation trees. In such a setting, it is in fact the notion that has come to be known as the "exact counting" acceptance mechanism. This mechanism was introduced by Simon [Sim75, p. 94], who proved that exact counting can be simulated by general probabilistic computation (in current terminology, $\text{C}_=\text{P} \subseteq \text{PP}$). The exact counting model and its power, in Simon's setting, have been studied in many papers (e.g., [Wag86; Tor91; TO92; KSTT92; OL93]).

THEOREM 4.1. *Universally serializable probabilistic polynomial-time tasks passing only one bit between tasks and using the exact counting acceptance mechanism accept exactly those languages in* NP^{PP} (*that is*, $\text{ProbabilisticSSF}_2 = \text{NP}^{\text{PP}}$).

The idea behind the $\text{ProbabilisticSSF}_2 \subseteq \text{NP}^{\text{PP}}$ direction of our proof is as follows. If something is accepted in this model, the question of whether the probability that 0 is mapped to 1 is $\frac{1}{2}$ must be unchanged by any shuffling of the tasks. We use this fact to obtain equations constraining the probabilistic mappings enacted by subblocks of tasks, and we conclude that an NP machine with a PP oracle can indeed check whether a $\text{ProbabilisticSSF}_2$ machine accepts.

Proof. First, we prove that $\text{ProbabilisticSSF}_2 \supseteq \text{NP}^{\text{PP}}$. $\text{C}_=\text{P}$ [Sim75; Wag86] is the "exact counting" class defined as follows (this is more commonly stated as a normal form, but we equivalently use it here as the definition): $L \in \text{C}_=\text{P}$ if and only if there is a polynomial $q(\cdot)$ and a polynomial-time predicate $R(\cdot, \cdot)$ such that for all strings x it holds that

$$x \in L \Leftrightarrow \|\{y \mid q(|x|) = |y| \text{ and } R(x, y)\}\| = \|\{y \mid q(|x|) = |y| \text{ and } \neg R(x, y)\}\|. \quad (16)$$

As is standard, $\exists \cdot \text{C}_=\text{P}$ denotes the class of all languages L such that there is a $\text{C}_=\text{P}$ language L' and a polynomial $r(\cdot)$ such that for all x it holds that

$$x \in L \Leftrightarrow (\exists y)[\,|y| = r(|x|) \text{ and } \langle x, y\rangle \in L'].$$

Assume that the pairing function respects lengths, that is, if $|x_1| = |x_2|$ and $|y_1| = |y_2|$, then $|\langle x_1, y_1\rangle| = |\langle x_2, y_2\rangle|$.

It is well known (simply by having the NP machine guess the exact probability of acceptance of the PP queries and then if the current nondeterministic path is one that would accept given the PP oracle answers implicit in the guesses then we accept if and only if a $\text{C}_=\text{P}$ check that all those guesses are correct says that they are correct) that $\text{NP}^{\text{PP}} = \text{NP}^{\text{C}_=\text{P}} = \exists \cdot \text{C}_=\text{P}$ [Tor91].

So, let our arbitrary NP^{PP} language, B, be specified via the polynomial $r(\cdot)$ and $q(\cdot)$ and the predicate $R(\cdot, \cdot)$ described above, with L' denoting the implied $\text{C}_=\text{P}$ language. On input x, our $\text{ProbabilisticSSF}_2$ machine will have $2^{r(|x|)}$ tasks, each corresponding to one possible value of the existential quantification of our $\exists \cdot L'$ representation. Each task will have $2^{q(|\langle x, 0^{r(|x|)}\rangle|)}$ equal-probability paths, each corresponding to one y string from Eq. (16). Along a given probabilistic path α of the task corresponding to the existential guess β, the task implements the following mapping between inputs and outputs: if $R(\langle x, \beta\rangle, \alpha)$ holds, then implement the identity mapping (0 maps to 0 and 1 maps to 1), otherwise, implement the transpose mapping (0 maps to 1 and 1 maps to 0).

We claim this $\text{ProbabilisticSSF}_2$ task set accepts exactly the language B. Why? Suppose $x \in B$. Then for some existential guess, the pairing of the input with the guess is in L', and so the task corresponding to that guess implements each of the transpose and the identity with probability $\frac{1}{2}$. In such a case, no matter when the probabilities (from the interval $[0, 1]$ and summing to one) of 0 and 1 were before the task, the probability, immediately after that task, of the zero token being output equals the probability of the one token being output equals $\frac{1}{2}$. Further, as all tasks (and thus in particular all subsequent tasks) in the construction divide their probability weight between the identity and the transpose, after all subsequence tasks the probability weight of 0 and the probability weight of 1 will remain tied, as for any h, $h/2 + (1-h)/2 = \frac{1}{2}$. So if $x \in B$, our $\text{ProbabilisticSSF}_2$ machine accepts.

On the other hand, if $x \notin B$, we claim our $\text{ProbabilisticSSF}_2$ machine will not accept. Certainly, before the first task, the probability of token 0 is not $\frac{1}{2}$ (in fact, it is 1, as that is the starting token). But note that if a task that divides all its probability weight between the identity and the transpose (as do the tasks of our constructed machine) outputs 0 with probability $\frac{1}{2}$, this can only be because either the probability that 0 was input to that task was $\frac{1}{2}$, or because the query to L' associated with the task is indeed in L' (that is, exactly half the paths of the $\mathrm{C_{=}P}$ predicate accept). We can see this as follows. Denote by p the probability that 0 is the token input to the task and denote by a the probability that a given task implements the identity. Then, for $\frac{1}{2}$ to be the probability that 0 is output, we must have

$$\tfrac{1}{2} = ap + (1-a)(1-p).$$

Solving, we see that this is satisfied only if $a = \frac{1}{2}$ or $p = \frac{1}{2}$. But if $x \notin B$, then for no task will a be $\frac{1}{2}$. But since the initial p is not $\frac{1}{2}$, it follows that, inductively, for no task's output will p (of that task's output) be $\frac{1}{2}$. And thus the final output is zero with some probability other than $\frac{1}{2}$. Thus x is not accepted by our ProbabilisticSSF$_2$ machine. This concludes the proof that ProbabilisticSSF$_2 \supseteq \mathrm{NP^{PP}}$.

We now turn to proving that ProbabilisticSSF$_2 \subseteq \mathrm{NP^{PP}}$. Let L be an arbitrary ProbabilisticSSF$_2$ language, and fix some scheme of tasks witnessing its membership in ProbabilisticSSF$_2$. Let x be a string of length n whose membership in L we are testing. There exists a polynomial p such that the tasks on input x are numbered $1, ..., h = 2^{p(n)}$. Also, without loss of generality, by normalizing the probabilistic computation of the tasks, we may assume that there is a polynomial q such that each task flips exactly $q(n)$ coins on each computation path. Over the two tokens $\{0, 1\}$, let ξ_1, ξ_2, ξ_3, and ξ_4 denote, respectively, the identity mapping, the transpose, the constant 0 mapping (i.e., both 0 and 1 are mapped to 0), and the constant 1 mapping. For each task t and j, $1 \leqslant j \leqslant 4$, let γ_t^j denote the probability that task t acts as mapping ξ_j. Note that for every j and t, γ_t^j is of the form $a/2^{q(n)}$ for some natural number a computable by a #P function.

For each task t, let $\sigma_t = \gamma_t^1 - \gamma_t^2$ and $\tau_t = \gamma_t^3 - \gamma_t^4$. For any permutation π of $\{1, ..., h\}$, let

$$P_\pi = \tau_{\pi(h)} + \sigma_{\pi(h)}(\tau_{\pi(h-1)} + \sigma_{\pi(h-1)} \times (\cdots(\tau_{\pi(2)} + \sigma_{\pi(2)}(\tau_{\pi(1)} + \sigma_{\pi(1)}))\cdots)). \qquad (17)$$

Note that the token 0 is mapped to 0 with probability $\frac{1}{2}$ when the tasks are executed in the order $\pi(1), ..., \pi(h)$ if and only if $P_\pi = 0$. So, as we are discussing universally serializable computation, we may even state the following.

$$[x \in L] \Leftrightarrow [\text{for some } \pi,\ P_\pi = 0] \Leftrightarrow [\text{for every } \pi,\ P_\pi = 0]. \qquad (18)$$

Let T be the set of all tasks t such that either $\gamma_t^3 > 0$ or $\gamma_t^4 > 0$. We claim that

$$x \in L \text{ if and only if either} \qquad (19)$$

$$\text{for some } t,\ 1 \leqslant t \leqslant h, \quad \sigma_t = \tau_t = 0, \qquad (19c1)$$

or

$\|T\| \leqslant q(n)+1$ and for any contiguous order of execution, the task set T maps 0 to 0 with probability $\frac{1}{2}$. (19c2)

Suppose (19c1) holds, and, thus, let t be such that $\sigma_t = \tau_t = 0$. Then any permutation π with $\pi(h) = t$ gives $P_\pi = 0$; so by (18), $x \in L$. Thus, we have only to show that

if (19c1) does not hold, then $x \in L$ if and only if (19c2) holds.

So, suppose that (19c1) does not hold. Then for every t, $1 \leqslant t \leqslant h$, $\sigma_t \neq 0$ or $\tau_t \neq 0$. Let $t \notin T$. By definition of T we have $\gamma_t^3 = \gamma_t^4 = 0$; so $\tau_t = 0$, and thus $\sigma_t \neq 0$. Now let $\ell = \|T\|$ and let π be a permutation that places the tasks in T first in the execution order, i.e., such that $\{\pi(1), ..., \pi(\ell)\} = T$. Then, by (17),

$$P_\pi = \left(\prod_{t \notin T} \sigma_t\right) \cdot (\tau_{\pi(\ell)} + \sigma_{\pi(\ell)}(\tau_{\pi(\ell-1)} + \sigma_{\pi(\ell-1)}(\cdots(\tau_{\pi(2)} + \sigma_{\pi(2)}(\tau_{\pi(1)} + \sigma_{\pi(1)}))\cdots))).$$

Since for every $t \notin T$, $\sigma_t \neq 0$, $P_\pi = 0$ if and only if

$$\tau_{\pi(\ell)} + \sigma_{\pi(\ell)}(\tau_{\pi(\ell-1)} + \sigma_{\pi(\ell-1)}(\cdots(\tau_{\pi(2)} + \sigma_{\pi(2)}(\tau_{\pi(1)} + \sigma_{\pi(1)}))\cdots)) = 0. \qquad (20)$$

So (still, of course, under our assumption that (19c1) fails to hold), $x \in L$ if and only if for every permutation π such that $\{\pi(1), ..., \pi(\ell)\} = T$, (20) holds. Thus, in light of the analogs of (17) and (18) for the task set T, we have only to show that in this case (19c2) holds if and only if for any permutation π with $\{\pi(1), ..., \pi(\ell)\} = T$, (20) holds. The equivalence trivially holds when $\|T\| \leqslant q(n)+1$. So, suppose $\|T\| = \ell > q(n)+1$. We will show that for some permutation with $\{\pi(1), ..., \pi(\ell)\} = T$, (20) does not hold.

Assume otherwise; that is, for any permutation with $\{\pi(1), ..., \pi(\ell)\} = T$, (20) holds even if (19c1) does not hold and $\|T\| > q(n)+1$. Note that for any $t \in T$ it holds that $|\sigma_t|$ is neither 0 nor 1. This is seen as follows. Suppose $\sigma_t = 0$ and π is a permutation with $\pi(\ell) = t$. Then the left-hand side of (20) is τ_t, which cannot be 0 since (19c1) does not hold. So, $P_\pi \neq 0$, which yields a contradiction. Suppose $|\sigma_t| = 1$. Then either γ_t^1 or γ_t^2 is 1, which implies $\gamma_t^3 = \gamma_t^4 = 0$, so $t \notin T$, a contradiction. Thus, $|\sigma_t|$ is neither 0 nor 1 if $t \in T$.

Let π and π' be two permutations such that $\{\pi(1), ..., \pi(\ell)\} = \{\pi'(1), ..., \pi'(\ell)\} = T$, $\pi(\ell) = \pi'(\ell-1)$, $\pi(\ell-1) = \pi'(\ell)$, and for every i, $1 \leqslant i \leqslant \ell-2$, $\pi(i) = \pi'(i)$. Then by our assumption, it holds that

$$\begin{aligned}\tau_{\pi(\ell)} + \sigma_{\pi(\ell)}(\tau_{\pi(\ell-1)} + \sigma_{\pi(\ell-1)} Z) \\ = \tau_{\pi(\ell-1)} + \sigma_{\pi(\ell-1)}(\tau_{\pi(\ell)} + \sigma_{\pi(\ell)} Z) = 0,\end{aligned}$$

where $Z = \tau_{\pi(\ell-2)} + \sigma_{\pi(\ell-2)}(\tau_{\pi(\ell-3)} + \sigma_{\pi(\ell-3)}(\cdots(\tau_{\pi(1)} + \sigma_{\pi(1)})\cdots))$. This implies that

$$\tau_{\pi(\ell)} + \sigma_{\pi(\ell)}\tau_{\pi(\ell-1)} = \tau_{\pi(\ell-1)} + \sigma_{\pi(\ell-1)}\tau_{\pi(\ell)}.$$

Note that $\pi(\ell)$ and $\pi(\ell-1)$ can take on two arbitrary distinct values in T. So, for every i, j, $1 \leqslant i < j \leqslant \ell$, it holds that

$$\tau_{\pi(i)} + \sigma_{\pi(i)}\tau_{\pi(j)} = \tau_{\pi(j)} + \sigma_{\pi(j)}\tau_{\pi(i)}.$$

In particular, for every i, $2 \leqslant i \leqslant \ell$, by letting $j = 1$ and $\alpha = \tau_{\pi(1)}/(1 - \sigma_{\pi(1)})$, we have

$$\tau_{\pi(i)} = \alpha(1 - \sigma_{\pi(i)}).$$

Note that α is well-defined, as we know from above that $\sigma_{\pi(1)}$ cannot be 1, as $\pi(1) \in T$. Now, by our assumption that (20) holds for any permutation, we have

$$\begin{aligned}\tau_{\pi(\ell)} + \sigma_{\pi(\ell)}(\tau_{\pi(\ell-1)} + \sigma_{\pi(\ell-1)}(\cdots(\tau_{\pi(2)} \\ + \sigma_{\pi(2)}(\tau_{\pi(1)} + \sigma_{\pi(1)}))\cdots)) = 0.\end{aligned}$$

By replacing $\tau_{\pi(i)}$ with $\alpha(1 - \sigma_{\pi(i)})$ for every $i > 1$, expanding, and cancelling terms, we have

$$\alpha + \left(\prod_{i=2}^{\ell} \sigma_{\pi(i)}\right)(\sigma_{\pi(1)} + \tau_{\pi(1)} - \alpha) = 0,$$

so

$$\frac{\tau_{\pi(1)}}{1 - \sigma_{\pi(1)}} + \left(\prod_{i=2}^{\ell} \sigma_{\pi(i)}\right)\left(\sigma_{\pi(1)} - \frac{\tau_{\pi(1)}\sigma_{\pi(1)}}{1 - \sigma_{\pi(1)}}\right) = 0,$$

and thus

$$\tau_{\pi(1)} + \left(\prod_{i=1}^{\ell} \sigma_{\pi(i)}\right)(1 - \sigma_{\pi(1)} - \tau_{\pi(1)}) = 0. \tag{21}$$

Let $\tau_{\pi(1)} = b_1/2^{q(n)}$, and for each i, $1 \leqslant i \leqslant \ell$, let $\sigma_{\pi(i)} = a_i/2^{q(n)}$. Then, by (21), we have

$$\frac{b_1}{2^{q(n)}} = \frac{(\prod_{i=1}^{\ell} a_i)}{2^{\ell \cdot q(n)}} \cdot \frac{a_1 + b_1 - 2^{q(n)}}{2^{q(n)}}.$$

Multiplying both sides by $2^{\ell \cdot q(n)} \cdot 2^{q(n)}$, we get

$$b_1 \cdot 2^{\ell \cdot q(n)} = \left(\prod_{i=1}^{\ell} a_i\right)(a_1 + b_1 - 2^{q(n)}). \tag{22}$$

It must be the case that $b_1 \cdot 2^{\ell \cdot q(n)} \neq 0$. Why? Suppose both sides of the above equation are 0. Then since each $a_t \neq 0$ by our assumption $\sigma_{\pi(t)} \neq 0$, $a_1 + b_1 - 2^{q(n)} = 0$ and $b_1 = 0$, so $a_1 = 2^{q(n)}$. This implies $\gamma_{\pi(1)}^3 = \gamma_{\pi(1)}^4 = 0$ and contradicts $\pi(1) \in T$. Thus, the value of both sides of (22) is nonzero. So, $0 < |a_1 + b_1 - 2^{q(n)}| \leqslant 2^{q(n)+1}$ and $0 < |b_1| \leqslant 2^{q(n)}$. Moreover, since for every t, $1 \leqslant t \leqslant \ell$, $|\sigma_{\pi(t)}| \notin \{0, 1\}$, it holds that $0 < |a_i| < 2^{q(n)}$. Now let 2^d and 2^e be the largest powers of 2 dividing b_1 and $a_1 + b_1 - 2^{q(n)}$, respectively, and for each i, let 2^{c_i} be the largest power of 2 dividing a_i. Then, we have $0 \leqslant d \leqslant q(n)$, $0 \leqslant e \leqslant q(n) + 1$, and for every i, $c_i \leqslant q(n) - 1$. By (22), we have

$$d + \ell \cdot q(n) = e + \sum c_i,$$

where the left-hand side is at least $\ell \cdot q(n)$ and the right-hand side is at most

$$q(n) + 1 + \ell \cdot q(n) - \ell.$$

But, our assumption $\ell > q(n) + 1$ implies $\ell \cdot q(n) > q(n) + 1 + \ell \cdot q(n) - \ell$; thus (22) cannot hold. Thus, we have a contradiction. This proves the claim.

Now it remains to show that the membership of x in L can be tested by a predicate in $\mathrm{NP}^{\mathrm{PP}}$. We use the "if and only if" equivalence stated as (19)/(19c1)/(19c2) to establish this. Condition (19c1) can be tested by a predicate in $\mathrm{NP}^{\mathrm{PP}}$: our nondeterministic Turing machine guesses task t, uses queries to a #P (or PP) oracle to compute τ_i and σ_i, and accepts x if and only if $\tau_i = \sigma_i = 0$. On the other hand, condition (19c2) can be tested in P^{PP}. Membership in T can be checked by a predicate in NP because t belongs to T if and only if there exists some computation path along which (i.e., some set of coin flips on which) task t acts as a constant function. So, "$\|T\| > q(n) + 1$?" can be tested by an NP predicate. If $\|T\| \leqslant q(n) + 1$, then, using binary search, we can enumerate all elements of T in polynomial time with queries to an NP oracle. After enumerating all elements of T, using a #P (or PP) oracle we can compute in polynomial time σ_t and τ_t for every $t \in T$. Now, the "if and only if" equivalence stated as (19)/(19c1)/(19c2) holds if and only if for some permutation of tasks in T, (20) holds. Since all τ_t and σ_t are known, the condition can be tested by a coNP predicate. So, since $\mathrm{NP} \cup \mathrm{coNP} \subseteq \mathrm{PP}$, the second condition can be tested by a P^{PP} predicate. Hence, the "if and only if" equivalence stated as (19)/(19c1)/(19c2) can be tested by an NP^{PP} predicate. This proves the theorem. ∎

COROLLARY 4.2. $\text{ProbabilisticSSF}_2 = NP^{PH^{PP}}$, *and thus, in particular*, $\text{ProbabilisticSSF}_2$ *contains the polynomial hierarchy*.

ACKNOWLEDGMENTS

We are very grateful to Joel Seiferas for asking the question that motivated this paper, to Edith Hemaspaandra for generous and useful suggestions, to Rajesh Rao for enjoyable conversations, and to an anonymous referee for helpful comments.

REFERENCES

[Bar89] D. Barrington, Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1, *J. Comput. System Sci.* **38**, No. 1 (1989), 150–164.

[BC93] D. Bovet and P. Crescenzi, "Introduction to the Theory of Complexity," Prentice–Hall, Englewood Cliffs, NJ, 1993.

[Bei91] R. Beigel, Relativized counting classes: Relations among thresholds, parity, and mods, *J. Comput. System Sci.* **42**, No. 1 (1991), 76–96.

[BGH90] R. Beigel, J. Gill, and U. Hertrampf, Counting classes: Thresholds, party, mods, and fewness, *in* "Proceedings, 7th Annual Symposium on Theoretical Aspects of Computer Science," Lecture Notes in Computer Science, Vol. 415, pp. 49–57, Springer-Verlag, New York/Berlin, 1990.

[BJY90] D. Bruschi, D. Joseph, and P. Young, Strong separations for the Boolean hierarchy over RP, *Int. J. Found. Comput. Sci.* **1**, No. 3 (1990), 201–218.

[BS95] R. Beigel and H. Straubing, The power of local self-reductions, *in* "Proceedings, 10th Structure in Complexity Theory Conference," pp. 277–285, IEEE Comput. Soc., Los Alamitos, CA, 1995.

[CF87] J. Cai and M. Furst, PSPACE survives three-bit bottlenecks, *in* "Proceedings, 2nd Structure in Complexity Theory Conference," pp. 94–102, IEEE Comput. Soc., Los Alamitos, CA, 1987.

[CF91] J. Cai and M. Furst, PSPACE survives constant-width bottlenecks, *Int. J. Found. Comput. Sci.* **2**, No. 1 (1991), 67–76.

[CGH+88] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung, The Boolean hierarchy I: Structural properties, *SIAM J. Comput.* **17**, No. 6 (1988), 1232–1252.

[CGH+89] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung, The Boolean hierarchy II: Applications, *SIAM J. Comput.* **18**, No. 1 (1989), 95–111.

[CH90] J. Cai and L. Hemachandra, On the power of parity polynomial time, *Math. Systems Theory* **23**, No. 2 (1990), 95–106.

[GNW90] T. Gundermann, N. Nasser, and G. Wechsung, A survey on counting classes, *in* "Proceedings, 5th Structure in Complexity Theory Conference," pp. 140–153, IEEE Comput. Soc., Los Alamitos, CA, 1990.

[GP86] L. Goldschlager and I. Parberry, On the construction of parallel computers from various bases of Boolean functions, *Theor. Comput. Sci.* **43** (1986), 43–58.

[Hem93] L. Hemachandra, Fault-tolerance and complexity, *in* "Proceedings, 20th International Colloquium on Automata, Languages, and Programming," Lecture Notes in Computer Science, Vol. 700, pp. 189–202, Springer-Verlag, New York/Berlin, 1993.

[Hem94] L. Hemaspaandra, The not-ready-for-prime-time conjectures, *SIGACT News* **25**, No. 2 (1994), 5–10.

[Her97] U. Hertrampf, Acceptance by transformation monoids (with an application to local self-reductions), *in* "Proceedings, 12th Annual IEEE Conference on Computational Complexity," pp. 213–224, IEEE Comput. Soc., Los Alamitos, CA, 1997.

[HH91] L. Hemachandra and A. Hoene, On sets with efficient implicit membership tests, *SIAM J. Comput.* **20**, No. 6 (1991), 1148–1156.

[HHH97] E. Hemaspaandra, L. Hemaspaandra, and H. Hempel, An introduction to query order, *Bulletin of the EATCS*, 1997, to appear.

[HHW] L. Hemaspaandra, H. Hempel, and G. Wechsung, Query order, *SIAM J. Comput.*, to appear.

[HO96] L. Hemaspaandra and M. Ogihara, "Universally Serializable Computation," Technical Report TR-638, Department of Computer Science, University of Rochester, Rochester, NY, September 1996.

[HR97] L. Hemaspaandra and J. Rothe, Unambiguous computation: Boolean hierarchies and sparse Turing-complete sets, *SIAM J. Comput.* **26**, No. 3 (1997), 634–653.

[Jac74] N. Jacobson, "Basic Algebra I," Freeman, San Francisco, 1974.

[Kil90] J. Kilian, "Uses of Randomness in Algorithms and Protocols," MIT Press, Cambridge, MA, 1990.

[Kre88] M. Krentel, The complexity of optimization problems, *J. Comput. System Sci.* **36** (1988), 490–509.

[KSTT92] J. Köbler, U. Schöning, S. Toda, and J. Torán, Turing machines with few accepting computations and low sets for PP, *J. Comput. System Sci.* **44**, No. 2 (1992), 272–286.

[LLS75] R. Ladner, N. Lynch, and A. Selman, A comparison of polynomial time reducibilities, *Theor. Comput. Sci.* **1**, No. 2 (1975), 103–124.

[Ogi94] M. Ogihara, On serializable languages, *Intern. J. Found. Comput. Sci.* **5**, Nos. 3–4 (1994), 303–318.

[OL93] M. Ogihara and A. Lozano, On sparse hard sets for counting classes, *Theor. Comput. Sci.* **112**, No. 2 (1993), 255–275.

[Pap94] C. Papadimitriou, "Computational Complexity," Addison–Wesley, Reading, MA, 1994.

[PZ83] C. Papadimitriou and S. Zachos, Two remarks on the power of counting, *in* "Proceedings, 6th GI Conference on Theoretical Computer Science," Lecture Notes in Computer Science, Vol. 145, pp. 269–276, Springer-Verlag, 1983.

[Sim75] J. Simon, "On Some Central Problems in Computational Complexity," Ph.D. thesis, Cornell University, Ithaca, NY, January 1975; Available as Cornell Department of Computer Science Technical Report TR75-224.

[TO92] S. Toda and M. Ogiwara, Counting classes are at least as hard as the polynomial-time hierarchy, *SIAM J. Comput.* **21**, No. 2 (1992), 316–328.

[Tod91] S. Toda, PP is as hard as the polynomial-time hierarchy, *SIAM J. Comput.* **20**, No. 5 (1991), 865–877.

[Tor88] J. Torán, "Structural Properties of the Counting Hierarchies," Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona, Spain, 1988.

[Tor91] J. Torán, Complexity classes defined by counting quantifiers, *J. of the ACM* **38** (1991), 753–774.

[Wag86] K. Wagner, The complexity of combinatorial problems with succinct input representations, *Acta Informatica* **23** (1986), 325–356.