A survey of space complexity

Pascal Michel
Université Paris 7, 59 Rue du Cardinal Lemoine, 75005 Paris, France

Abstract

Michel, P., A survey of space complexity, Theoretical Computer Science 101 (1992) 99–132.

The main properties of deterministic and nondeterministic space complexity classes are given, with emphasis on closure under complementation. Various limitations and generalizations of these classes are studied: weakly space-bounded classes, classes defined by one-way, alternating, or probabilistic machines, and nonuniform classes. In each case intrinsic properties of these classes and relationship between these classes are given. Then three ways of relativizing complexity classes are examined. Finally, the space complexity of RAMs is defined and its relation to usual classes is given.

1. Introduction

Space complexity was introduced by Stearns et al. [79] in 1965. It is the second in importance of the two most widespread ways of measuring the complexity of a computation, the first one being time complexity.

Intuitively, if you make a computation on a paper, then the complexity of this computation can be measured by the time you spend or by the amount of paper you use. If you write in ink there is no difference between both. But if you write with a pencil and an eraser, trying to save place, then the amount of paper you need can be far less clearly related to the time you spend. Space complexity formalizes this way of measuring the complexity of a computation. In an actual computation by a computer, the feature formalized by space complexity is the amount of memory used by the computer.

Computational complexity introduced many abstract models of computation, the most widespread being the Turing machine, but many of these models give rise to essentially equivalent formalizations of such fundamental concepts as time or space complexity.

In Section 2 we define Turing machines and space complexity classes, giving their basic properties and some definitions, notations and lemmas. Section 3 is devoted to the space constructibility of functions, a property which occurs in the hypotheses of many theorems in this paper. In Sections 4 and 5 we study closure and hierarchy theorems for deterministic and nondeterministic space complexity classes. Relationship between these last two kinds of classes is studied in Section 6. Next, the impact of various limitations or generalizations on machines are surveyed: weak acceptance in Section 7, one-way machines in Section 8, alternating machines in Section 9, probabilistic machines in Section 10, and nonuniform complexity in Section 11. In each of these last sections, we consider the intrinsic properties of the complexity classes and the relationship of these classes with the usual ones. In Section 12 we consider various ways of relativizing a space complexity class. Finally, we present in Section 13 the space complexity of RAMs, illustrating the relative independence of the concept of space from the model of computation that is chosen.

Among the topics which are omitted, the major ones are space-bounded reducibilities and complete problems for space complexity classes. Relationship between space complexity and other measures of complexity is only cursorily treated. To avoid inflation, proofs are often sketched or omitted, especially if they are available in many textbooks, or related to minor results, or too long. In each case, detailed references are provided.

2. Preliminaries

In this section, we describe the model of computation we use, we give some basic results about this model and the space complexity classes it defines, and state some definitions and useful lemmas.

If f and g are functions from the set $\mathbb{N}$ of natural numbers into itself, then $f=o(g)$ if $\lim f/g=0$, and $f=O(g)$ if there is a constant c such that for any large enough n, $f(n)\leqslant cg(n)$. The function log denotes the base-two logarithm. Frequently, n will denote the length $|x|$ of the input x.

Turing machines

The model we use is the *Turing machine* as it is defined in standard textbooks such as [8, 35] or [90]. A Turing machine consists of

- a control unit, with a finite number of *states*,
- an *input tape* on which the input is written between two endmarkers, with a tape head which can move right and left on the input (*two-way* machine), and cannot modify the input (*read-only* head),
- a fixed finite number k of semi-infinite *work tapes* (with an endmarker on the leftmost cell), each with a tape head which can move right and left on the tape and can write on it (*read–write* head).

Initially, the control unit is in the *initial state* q_0, the cells of the work tapes, other than the leftmost cells, are blank, and each head is on the leftmost cell. In one step of

computation, the Turing machine, according to the state of the control unit and the symbols scanned by the heads,
- changes the state,
- prints symbols which replace symbols scanned, and
- moves heads.

Formally, a Turing machine is $M=(Q,\Sigma,\delta,q_0,F)$, where Q is the finite set of states, Σ is the *tape alphabet*, i.e. the finite set of allowable symbols, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of *final* states (with *accepting* and *rejecting* states), and δ is the *transition function*. For a *deterministic Turing machine* (DTM), δ is a partial function

$$\delta : Q \times \Sigma^{k+1} \rightarrow Q \times \Sigma^k \times \{-1,0,1\}^{k+1}.$$

If $\delta(q,a_0,a_1,\ldots,a_k)=(q',a'_1,\ldots,a'_k,\varepsilon_0,\varepsilon_1,\ldots,\varepsilon_k)$, then the machine M, in state q, reading a_0 on the input tape and $a_1,\ldots,a_k$ on the work tapes,
- enters state q',
- writes $a'_1,\ldots,a'_k$ in place of $a_1,\ldots,a_k$,
- moves the input tape head by ε_0, and the work tapes heads by $\varepsilon_1,\ldots,\varepsilon_k$.

For a *nondeterministic Turing machine* (NTM),

$$\delta \subseteq (Q \times \Sigma^{k+1}) \times (Q \times \Sigma^k \times \{-1,0,1\}^{k+1}),$$

i.e. at each step, the machine may have many possibilities for the next move.

Given an input $x \in \Sigma^*$, a *configuration* of M consists of
- the state of M,
- the positions of its input and work tapes heads,
- the contents of its work tapes.

A *computation* of M on the input x is a sequence of configurations of M beginning with the initial configuration, and such that one goes from a configuration to the next one according to δ. This allows to define, for a DTM, the computation of M on x, and for a NTM, the tree of possible computations of M on x. A computation is *accepting* if it ends in an accepting state. An input x is accepted by a DTM M if the computation of M on x is accepting. An input x is accepted by a NTM M if there exists an accepting computation of M on x. The *language accepted* by M is the set $L(M)$ of words x accepted by M.

A Turing machine M *works in space* $S(n)$ (or is $S(n)$ space-bounded) if for every n, every input x of length n, and every computation of M on x, at most $\max(S(n),1)$ cells are scanned on each work tape of M. DSPACE($S(n)$) (resp. NSPACE($S(n)$)) is the class of languages accepted by the deterministic (resp. nondeterministic) Turing machines working in space $S(n)$.

A Turing machine M works in time $T(n)$ (or is $T(n)$ time-bounded) if for every n and every input x of length n, all computations of M on x end in less than $T(n)$ steps. DTIME($T(n)$) (resp. NTIME($T(n)$)) is the class of languages accepted by the deterministic (resp. nondeterministic) Turing machines working in time $T(n)$.

Classes such as DSPACE($S(n)$), NSPACE($S(n)$), etc. are called *complexity classes*. Ways of quantifying the complexity of a language, such as deterministic space, nondeterministic space, etc., are called *complexity measures*.

If $\mathscr{C}$ is a complexity class, then

$$\text{co-}\mathscr{C}=\{\Sigma^*-L\colon \Sigma \text{ finite alphabet, } L\in\mathscr{C}\}.$$

An *output tape* can be added to a Turing machine, allowing it to compute a function. Such a tape is one-way and write-only, and, when the machine stops, contains the value of the computed function. If $\mathscr{C}$ is a complexity class, $F\mathscr{C}$ denotes the class of functions computed by Turing machines which are bounded according to $\mathscr{C}$.

Influence of the number of tapes and the number of heads [90, pp. 341–342]. Space complexity classes remain unchanged if they are defined by Turing machines with only one work tape. A Turing machine M with k work tapes can be simulated by a Turing machine M' with one work tape divided into $2k$ parallel tracks, numbered $1,2,\ldots,2k$. The odd-numbered tracks contain the symbols of the tapes of M, and the even-numbered tracks are blank, except for a marker pointing the place of the head of the corresponding tape. The space used by M' is the same as the space used by M.

From now on, we assume that Turing machines have only one work tape.

Space complexity classes remain unchanged if many heads are allowed on each work tape of Turing machines. But if both the number of heads and the number of work tapes symbols are limited, then space complexity classes are lessened; see [36, 69, 70].

Linear speed-up [90, pp. 330–331]. Let

$$\mathrm{DSPACE}(\mathrm{O}(S(n)))=\bigcup_{c>0}\mathrm{DSPACE}(cS(n)).$$

Theorem 2.1. *For any function S and any $c>0$,*

$$\mathrm{DSPACE}(S(n))=\mathrm{DSPACE}(cS(n))=\mathrm{DSPACE}(\mathrm{O}(S(n))),$$

and

$$\mathrm{NSPACE}(S(n))=\mathrm{NSPACE}(cS(n))=\mathrm{NSPACE}(\mathrm{O}(S(n))).$$

Proof (*sketch*). By increasing the work tape alphabet, a block of m cells can be coded by one symbol. □

Relationship between space and time. The detailed study of the relationship between space and time measures are beyond the scope of this paper. See for example [90, pp. 451–473]. Roughly, a Turing machine works in a time exponential in its space bound. Formally, we have the following theorem.

Theorem 2.2. *If $S(n) \geqslant \log n$, then*

$$\mathrm{DSPACE}(S(n)) \subseteq \mathrm{DTIME}(2^{O(S(n))}),$$

and

$$\mathrm{NSPACE}(S(n)) \subseteq \mathrm{NTIME}(2^{O(S(n))}).$$

It can be noted that the concept of alternation, introduced in Section 9, allows a neater view of the relationship between space and time. Theorem 9.2 states a relation between alternating space and deterministic time which implies both results of Theorem 2.2. On the other hand, alternating time is closely related to deterministic space (see Theorem 9.11). Theorems 6.2 and 10.6 are two examples where a result of inclusion in a deterministic space complexity class is achieved with no more labor by an inclusion in an alternating time complexity class.

Definitions 2.3. A *finite automaton* is a Turing machine without work tape. A *regular set* is a set accepted by a finite automaton. A *generalized sequential machine* is a one-way finite automaton with an output tape allowing the computation of a function. A *homomorphism* is a function $h: \Sigma^* \rightarrow \Sigma^*$ satisfying $h(uv) = h(u)h(v)$ for every $u, v \in \Sigma^*$. A function is *e-free* if the empty word does not belong to its range. An *abstract family of languages* (AFL) is a family of languages closed under e-free homomorphism, inverse homomorphism, intersection with regular sets, union, concatenation and Kleene star. x^{-1} denotes the reverse of the word x.

Lemma 2.4. *For all positive integers a, b, $a \neq b$, there is a prime number p, $p \leqslant 2\log_2(a+b)$, such that $a \not\equiv b \pmod p$.*

Proof (*sketch*). The relevant reference for this result is Rosser and Schoenfeld [62]. Let $\theta(x) = \ln(\prod_{p \leqslant x} p)$. By Theorem 10 of [62, p. 71], for all $x \geqslant 101$, $\theta(x) \geqslant 0.84x$. An inspection of numbers below 101 shows that for all $x \geqslant 3$, $\theta(x) \geqslant 4x/10$.

We prove in a first step that for any $x \geqslant 3$ there is a prime number $p \leqslant 2\log x$, such that $x \not\equiv 0 \pmod p$. Else, $x \equiv 0 \pmod p$ for every prime number $p \leqslant 2\log x$; thus, $x \equiv 0$ $(\mathrm{mod} \prod\{p\colon p \text{ prime}, p \leqslant 2\log x\})$. But then $x \geqslant \prod\{p\colon p \text{ prime}, p \leqslant 2\log x\}$, for $x \neq 0$. Thus, $\ln x \geqslant \ln(\prod\{p\colon p \text{ prime}, p \leqslant 2\log x\}) = \theta(2\log x)$. Then $x \geqslant 3$ implies $2\log x \geqslant 3$ and by the result above, $\theta(2\log x) \geqslant 8\log x/10$. Thus, $\ln x \geqslant 8\log x/10$, or $\ln 2 \geqslant 8/10$, which is false.

Now let a, b be positive integers, $a \neq b$. Then $a + b \geqslant 1 + 2 = 3$. If $|a-b| = 1$, then $p = 2 \leqslant 2\log 3$ suits. If $|a-b| = 2$, then $p = 3 \leqslant 2\log 3$ suits. If $|a-b| \geqslant 3$, then, by the result of the preceding paragraph, there is a prime number $p \leqslant 2\log(|a-b|) \leqslant 2\log(a+b)$ such that $|a-b| \not\equiv 0 \pmod p$, i.e., $a \not\equiv b \pmod p$.

3. Fully space constructible functions

As we shall see in the sequel, constructibility hypotheses occur in the statement of many theorems. When it can be done without it, it is often at the expense of a lengthier proof.

Definition 3.1. A function $S:\mathbb{N}\to\mathbb{N}$ is *fully space constructible* (f.s.c.) if there is a DTM M working in space $S(n)$ which marks off $S(n)$ cells on the work tape on all inputs of length n.

Theorem 3.2 (Freedman and Ladner [19], Szepietowski [82]). *If* $\lim S(n)=+\infty$ *and* $S(n)=\mathrm{o}(\log n)$, *then* $S(n)$ *is not fully space constructible.*

Proof (*Freedman and Ladner* [19, p. 124], *Hartmanis and Ranjan* [33, p. 51]). Suppose that $S(n)$ is f.s.c., and let M be a DTM working in space $S(n)$, which marks off $S(n)$ cells on the work tape, on the input 1^n. Let q be the number of states and r the number of tape symbols of M.

A partial configuration of M is given by
- the state of M,
- the position of its work tape head,
- the contents of its work tape.

There are less than $qS(n)r^{S(n)}$ partial configurations of M on 1^n. Since $S(n)=\mathrm{o}(\log n)$, this number is less than n for large enough n. Thus M, while reading the input, comes back to the same partial configuration, and enters a cycle of length $m\leqslant n$. As m divides $n!$, M reaches the end of the input in the same partial configuration on input 1^n as on input $1^{n+n!}$. The same argument repeats when M comes back to the beginning of the input. For all k, M has the same behavior on 1^n and $1^{n+kn!}$, and uses the same space: $\exists n_0\ \forall n\geqslant n_0\ \forall k\ \ S(n)=S(n+kn!)$. This contradicts the hypothesis: $\lim S(n)=+\infty$. □

Remark 3.3. The proof method $n\to n+n!$ was used for the first time by Stearns et al. [79].

Szepietowski [82] proves that if $\liminf S(n)/\log n=0$ and $\exists c\ S(n)\geqslant c\log\log n$, then $S(n)$ is not f.s.c., by another method, using Lemma 2.4.

Freedman and Ladner [19] prove that if $\limsup S(n)/\log n=0$ and $\liminf S(n)=+\infty$, then $S(n)$ is not f.s.c., by the $n\to n+n!$ method.

Both these results imply Theorem 3.2.

Remark 3.4. Theorem 3.2 does not preclude the existence of f.s.c. functions $S(n)$ satisfying $S(n)\leqslant\log\log n$ for all n. For example, $S(n)=\log(\min\{i\colon i \text{ does not divide } n\})$ is such a function. More precisely, $\forall n\geqslant 2\ S(n)\leqslant\log\log n+2$. The function $S(n)$ is f.s.c. by a DTM M which writes $2,3,4,\ldots$ in binary on its work tape and tests whether they divide n, till a number that does not divide n is attained.

Definition 3.5. A function $S: \mathbb{N} \to \mathbb{N}$ is *space constructible* if there is a DTM M working in space $S(n)$ such that for all n there is an input of length n on which M uses exactly $S(n)$ cells.

Of course, f.s.c. functions are space constructible.

Definition 3.6 (*Hartmanis and Ranjan* [33, p. 56], *Szepietowski* [86, p. 110]). A function $S: \mathbb{N} \to \mathbb{N}$ is *nondeterministically fully space constructible* if there is a NTM M working in space $S(n)$ (for all inputs and all computations) such that for any input $x \in \Sigma^*$ there is a computation of M on x using exactly $S(n)$ cells.

Theorem 3.7 (Geffert [24]). *If* $\lim S(n) = +\infty$ *and* $S(n) = o(\log n)$, *then* $S(n)$ *is not nondeterministically fully space constructible.*

Remarks 3.8. For more definitions of constructibility and their relations, see Seiferas [69, pp. 75–76].

Usual functions, such as $\log n, n^k, 2^n$, are f.s.c. Sums and products of f.s.c. functions are f.s.c.

4. Properties of deterministic space classes: DSPACE($S(n)$)

Closure under complementation

Theorem 4.1 (Sipser [75, 76]). *For any function* $S(n)$, DSPACE($S(n)$) = co-DSPACE($S(n)$), *i.e.*, DSPACE($S(n)$) *is closed under complementation.*

Proof. By definition of DSPACE($S(n)$), the work tape head of a DTM M working in space $S(n)$ never moves away by more than $S(n)$ cells from the beginning of the tape. On an input that M does not accept, either M attains a rejecting state or M enters a loop and never halts. In the first case, recognizing the complement is easy: accepting and rejecting states are swapped. The problem is to eliminate loops. Therefore, it is sufficient to show the following proposition.

Proposition 4.2. *For any function S and any DTM M working in space $S(n)$, there is a DTM N working in space $S(n)$, accepting the same language as M, and halting on every input.*

Proof. It can be supposed that M, when it accepts an input, erases the work tape, moves back the input and work tapes on the initial endmarkers, and has only one accepting state. Then M has a unique accepting configuration. Remember that a configuration consists of

- the state of M,
- the position of its input tape head,
- the position of its work tape head,
- the contents of its work tape.

Let $x \in \Sigma^*$ be a fixed input. For a given natural number k, the machine N uses a procedure to determine if the machine M can reach the accepting configuration from the initial configuration in space no more than k. Consider the directed graph of all possible configurations of M, with an arc between two configurations if the transition function maps the first one to the second one. This graph may have many connected components, and may contain loops. As the computation is deterministic, each vertex has fan-out at most one.

The connected component containing the accepting configuration is a tree. The machine N determines whether this tree contains the initial configuration by a depth-first search. N simulates M from the accepting configuration, to the past and to the future within this tree.

The machine N does not know the space k used by M. N executes the procedure for $k = 1, 2, 3, \ldots$ If M accepts x, then N reaches the initial configuration. Else, N might never stop. So, before executing the procedure for $k+1$, N makes sure that M uses space at least $k+1$. N does so by enumerating the configurations using space k and selecting those which use space $k+1$ at the next step. For each of these, N searches in its past for the initial configuration. If no such search succeeds, then M never uses space $k+1$, and N rejects. Else, N executes the procedure for space $k+1$, from the accepting configuration, as described above. $\square$

Remarks 4.3. Hopcroft and Ullman [34, p. 170, 35, p. 297] give a simpler proof if $S(n) \geq \log n$.

An even simpler proof can be given if $S(n)$ is f.s.c. and $S(n) \geq \log n$, by marking $S(n)$ cells and counting the number of steps, in base c, where $c^{S(n)}$ is greater than the number of possible configurations. See [8, p. 48].

Hartmanis and Berman [29, 30, pp. 218–219] give a proof for DSPACE$(S(n)) \cap \mathscr{P}(\{1\}^*)$, without any hypothesis on $S(n)$.

Other closure properties

Theorem 4.4 (Hopcroft and Ullman [34, p. 175], Wagner and Wechsung [90, p. 196]). *For any function $S(n)$,* DSPACE$(S(n))$ *is closed under union, intersection and intersection with regular sets.*

The proofs are easy.

Theorem 4.5 (Wagner and Wechsung [90, pp. 196–197]). *Let $S(n)$ be an increasing function. Then* DSPACE$(\mathrm{O}(S(n)))$ *is closed*

(i) *under inverse homomorphism,*

(ii) *if $S(n) \geq \log n$, under inverse generalized sequential machine.*

Theorem 4.6 (Wagner and Wechsung [90, pp. 196–197]). *Let $S(n)$ be an increasing function. Then* DSPACE($S(n)$) *is closed*

(i) *if $S(n) \geqslant \log n$, under concatenation,*

(ii) *if $S(n) \geqslant n$, under Kleene star, e-free homomorphism, and e-free generalized sequential machine.*

Theorem 4.7 (Ibarra and Ravikumar [37, p. 4]). *If* $\log\log n \leqslant S(n) = o(\log n)$, *then* DSPACE($S(n)$) *is not closed under the following operations*:
- *concatenation,*
- *Kleene star,*
- *homomorphism,*
- *suffix* ($\mathrm{SUFFIX}(L) = \{y\colon \exists x\ xy \in L\}$),
- *permutation* ($\mathrm{PERM}(L) = \{y\colon$ *some permutation of* $y \in L\}$).

Theorem 4.8 (Wagner and Wechsung [90, p. 197] and Ibarra and Ravikumar [37, pp. 8–9]).

(i) DSPACE($\log n$) *is closed under Kleene star*
  $\Leftrightarrow$ DSPACE($\log n$) = NSPACE($\log n$),

(ii) DSPACE($\log n$) *is closed under e-free homomorphism*
  $\Leftrightarrow$ DSPACE($\log n$) *is closed under e-free generalized sequential machine*
  $\Leftrightarrow$ DSPACE($\log n$) *is an AFL*
  $\Leftrightarrow$ DSPACE($\log n$) = ATIME-ALT($O(n), O(1)$) (= RUD) (see Section 9)
  $\Leftrightarrow$ DSPACE($\log n$) = NP,

(iii) DSPACE($\log n$) *is closed under suffix*
  $\Rightarrow$ DSPACE($\log n$) = NSPACE($\log n$),

(iv) DSPACE($\log n$) *is closed under permutation*
  $\Rightarrow$ DSPACE(n) = NSPACE(n).

Hierarchy theorems

Theorem 4.9. *If S_2 is space constructible and* $\liminf S_1(n)/S_2(n) = 0$, *then*

$$\mathrm{DSPACE}(S_2(n)) - \mathrm{DSPACE}(S_1(n)) \neq \emptyset.$$

The proof is by diagonalization, using Theorem 4.1. See, for example, [35, p. 298], [8, pp. 52–53] or [90, p. 311].

It is possible to find a subset of $\{1\}^*$ in DSPACE($S_2(n)$) − DSPACE($S_1(n)$): see [90, p. 312].

Theorem 4.10 (Low-end hierarchy theorem) (Stearns et al. [79]). *If* $S(n) = o(\log\log n)$, *then*

$$\mathrm{Reg} = \mathrm{DSPACE}(S(n)) \subsetneq \mathrm{DSPACE}(\log\log n).$$

Proof (*sketch*). A simple counting argument on crossing sequences shows that

$$\mathrm{Reg} = \mathrm{DSPACE}(S(n)) \text{ if } S(n) = o(\log\log n).$$

See [90, pp. 131, 126].

We show that $\{\mathrm{bin}(0)\#\mathrm{bin}(1)\#\cdots\#\mathrm{bin}(k)\colon k\in\mathbb{N}\}\in\mathrm{DSPACE}(\log\log n)$, where $\mathrm{bin}(i)$ is the number i written in binary [79, 90, p. 131]. A machine can test whether b is $a+1$ in binary, by counting the places of 0's and 1's in a and b. Such a machine works in space $\log|b| = \log\log b \leqslant \log\log k$. But

$$n = |\mathrm{bin}(0)\#\mathrm{bin}(1)\#\cdots\#\mathrm{bin}(k)| \geqslant k.$$

This language is not regular. □

5. Properties of nondeterministic space classes: NSPACE($S(n)$)

Closure under complementation

Theorem 5.1 (Immerman [39] and Szelepcsényi [81]). *For any function* $S(n)\geqslant\log n$ $\mathrm{NSPACE}(S(n)) = \text{co-}\mathrm{NSPACE}(S(n))$, *i.e.*, $\mathrm{NSPACE}(S(n))$ *is closed under complementation.*

Proof (Szelepcsényi [81]). Let M be a NTM working is space $S(n)\geqslant\log n$. Remember that a configuration of M consists of
- the state of M,
- the position of its input tape head,
- the position of its work tape head,
- the contents of its work tape.

The length of a configuration is $\leqslant O(1)+\log n+\log S(n)+S(n) = O(S(n))$, so the number of configurations is at most $c^{S(n)}$ for a constant c.

Let $x\in\Sigma^*$ be a fixed input.

Let K be the set of the configurations of M, ordered by increasing length, and lexicographically for a fixed length. Let IC be the initial configuration of M on x. Let $J\subseteq K$ be the set of the configurations which are reachable by M on x from IC. Let $N=\mathrm{card}(J)$, and $MC=\max(J)$. Then MC uses space $O(S(n))$.

Fact 1. *There is a NTM M' working in space $O(S(n))$ which, given IC, MC and N on its work tape, accepts x iff M rejects x.*

Proof of Fact 1. By definition of nondeterminism,

M rejects $x \Leftrightarrow$ none of the configurations which are reachable from IC are accepting.

And by definition of N,

$$M \text{ rejects } x \Leftrightarrow \text{there are at least } N \text{ nonaccepting configurations which are reachable from } IC.$$

Let M' be the NTM which

- enumerates K up to MC and, while doing that,
  - guesses which are the configurations in J,
  - verifies that each of these configurations in J is nonaccepting and reachable from IC,
  - counts these configurations in J,
- verifies that N configurations have been counted,
- accepts if all that precedes can be done.

Then M' works in space $O(S(n))$ and M' accepts x iff there are at least N nonaccepting configurations which are reachable from IC, i.e., iff M rejects x. □

Fact 2. *There is a NTM M' working in space* $O(S(n))$ *which, given IC on its work tape, computes the number $N = \text{card}(J)$ of configurations reachable by M from IC and the greatest of these configurations,* $\max(J)$.

Proof of Fact 2. Let J_d be the set of the configurations which are reachable from IC in at most d steps. Let $N_d = \text{card}(J_d)$. Then there is an increasing sequence of sets: $J_0 \subseteq J_1 \subseteq \cdots$. This sequence is eventually stationary. J is the first J_d with $J_d = J_{d+1}$. Then we get $N = \text{card}(J)$ and $MC = \max(J)$.

We prove by induction on d that N_d and $\max(J_d)$ are computable. This is evident for $d = 0, 1$. Suppose that d, N_d, and $\max(J_d)$ are given on the work tape of M'. Then M' can enumerate J_d by the following procedure: M'

- enumerates K up to $\max(J_d)$ and, while doing that
  - guesses the members of J_d,
  - verifies they are in J_d,
  - counts them,
- verifies that N_d configurations have been counted.

To compute $\max(J_{d+1})$, M' enumerates J_d and, for each of these configurations, enumerates the configurations which are reachable in 0 or 1 steps. Thus, M' enumerates J_{d+1}, and can get $\max(J_{d+1})$.

To compute N_{d+1}, M' enumerates J_d and enumerates the configurations reachable from them in one step. M' must avoid counting a configuration twice. So, for each configuration, M' enumerates again from the beginning, comparing the configuration to the preceding ones, and finds out whether it is new and may be counted.

M' computes N_d and $\max(J_d)$ for $d = 0, 1, \ldots$, in space $O(S(n))$; M' stops when $N_{d+1} = N_d$ and gets $N = \text{card}(J)$ and $\max(J)$. □

Proof of Theorem 5.1 (*conclusion*). Let M' be the NTM which computes IC from the input x. Next, by Fact 2, M' computes $N = \text{card}(J)$ and $MC = \max(J)$. At last, by

Fact 1, M' accepts x iff M rejects x. Then M' works in space $O(S(n))$ and accepts the complement of the language accepted by M. □

Remarks 5.2. The proof above is from Szelepcsényi [81]. The proof of Immerman [39] (see also [9, pp. 257–262]) is only slightly different. Immerman enumerates the configurations of K which use space $S(n)$, avoiding the knowledge of MC in Fact 1 and the computation of $\max(J_d)$ and $\max(J)$ in Fact 2. This can be done directly if $S(n)$ is f.s.c. Else, M' tries to enumerate for $S(n)=1,2,\ldots$ till a space is attained within which all computations can be done.

The problem of the closure under complementation of NSPACE$(S(n))$ for $S(n)=o(\log n)$ is open.

Corollary 5.3. *The class of context-sensitive languages is closed under complementation.*

Proof. Kuroda [51] showed that CLS = NSPACE(n). □

Other closure properties

Theorem 5.4 (Wagner and Wechsung [90, pp. 196–197]). *For any function $S(n)$,* NSPACE$(S(n))$ *is closed under union, intersection, and intersection with regular sets.*

Theorem 5.5 (Wagner and Wechsung [90, pp. 196–197]). *Let $S(n)$ be an increasing function. Then* NSPACE$(O(S(n)))$ *is closed*

(i) *under inverse homomorphism,*

(ii) *if $S(n)\geq \log n$, under inverse generalized sequential machine.*

Theorem 5.6 (Wagner and Wechsung [90, pp. 196–197]). *Let $S(n)$ be an increasing function. Then* NSPACE$(S(n))$ *is closed*

(i) *if $S(n)\geq \log n$, under concatenation and Kleene star,*

(ii) *if $S(n)\geq n$, under e-free homomorphism and e-free generalized sequential machine.*

Theorem 5.7 (Wagner and Wechsung [90, p. 197]).

NSPACE$(\log n)$ *is closed under e-free homomorphism*

$\Leftrightarrow$ NSPACE$(\log n)$ *is closed under e-free generalized sequential machine*

$\Leftrightarrow$ NSPACE$(\log n)$ *is an AFL*

$\Leftrightarrow$ NSPACE$(\log n)$ = NP.

Hierarchy theorem

Theorem 5.8. *If $S_2(n)$ is f.s.c., $S_2(n)\geq\log(n)$ and* $\liminf S_1(n)/S_2(n)=0$, *then*

$$\text{NSPACE}(S_2(n))-\text{NSPACE}(S_1(n))\neq\emptyset.$$

This theorem is stated by Seiferas [70] with stronger hypotheses. See also [90, p. 314]. Theorem 5.1 allows this statement and a simpler proof by diagonalization [39, p. 937].

Theorem 5.9 (Hopcroft and Ullman [34], Wagner and Wechsung [90, p. 317]). *If* $S_2(n) \leqslant \log n$, S_2 *f.s.c. and* $\liminf S_1(n)/S_2(n) = 0$, *then*

$$\mathrm{NSPACE}(S_2(n)) - \mathrm{NSPACE}(S_1(n)) \neq \emptyset.$$

The proof depends on showing that

$$\{u2^k u^{-1} : u \in \{0, 1\}^*, |u| = 2^{S_2(|u2^k u^{-1}|)}\} \in \mathrm{NSPACE}(S_2(n)) - \mathrm{NSPACE}(S_1(n)).$$

Theorem 5.10 (low-end hierarchy). *If* $S(n) = o(\log\log n)$, *then* $\mathrm{Reg} = \mathrm{NSPACE}(S(n)) \subsetneq \mathrm{NSPACE}(\log\log n)$.

Proof (*sketch*). For $\mathrm{Reg} = \mathrm{NSPACE}(S(n))$ if $S(n) = o(\log\log n)$, see [90, p. 131]. The languages exhibited in the proof of Theorem 4.10 show that $\mathrm{Reg} \subsetneq \mathrm{NSPACE}(\log\log n)$. $\square$

6. Relationship between deterministic and nondeterministic space classes

Remark 6.1. Evidently, for any $S(n)$, $\mathrm{DSPACE}(S(n)) \subseteq \mathrm{NSPACE}(S(n))$. If $S(n) = o(\log\log n)$, then $\mathrm{Reg} = \mathrm{DSPACE}(S(n)) = \mathrm{NSPACE}(S(n))$. Else, the problem of the strict inclusion is open. Kannan [43] proves that if $\log\log n \leqslant S(n) = o(\log n)$, then there is a NTM working in space $S(n)$ which cannot be simulated, in a special sense, by a DTM working in space $S(n)$. But this does not imply that $\mathrm{DSPACE}(S(n)) \subsetneq \mathrm{NSPACE}(S(n))$, despite what is written in [90, p. 419].

Theorem 6.2 ("Savitch theorem"). *For any function* $S(n)$, $\mathrm{NSPACE}(S(n)) \subseteq \mathrm{DSPACE}(S(n)(S(n) + \log n))$.

Savitch [66] shows that if $S(n)$ is f.s.c. and $S(n) \geqslant \log n$, then

$$\mathrm{NSPACE}(S(n)) \subseteq \mathrm{DSPACE}(S(n)^2).$$

Monien and Sudborough [57, 58] show that if $S(n)$ is f.s.c. and $\log\log n \leqslant S(n) \leqslant \log n$, then

$$\mathrm{NSPACE}(S(n)) \subseteq \mathrm{DSPACE}(S(n)\log n)$$

(see also [90, p. 416]). The statement above is from Tompa [88], who shows in fact that

$$\mathrm{NSPACE}(S(n)) \subseteq \mathrm{ATIME}(S(n)(S(n) + \log n))$$

(see Theorem 9.11). For a proof when $S(n)$ f.s.c., $S(n) \geqslant \log n$, see for example [35, pp. 301–302] or [8, pp. 50–52].

Corollary 6.3. NSPACE(poly) = DSPACE(poly).

This class is denoted by PSPACE.

Theorem 6.4 (Litow [54]). *If* $S(n) = o(\log n)$, *L is a bounded language* (*i.e.*, $\exists a_1, \ldots, a_k$ $L \subseteq a_1^* \ldots a_k^*$), *and* $L \in \mathrm{NSPACE}(S(n))$, *then* $L \in \mathrm{DSPACE}(\max(S(n)^2, \log n))$.

Translational theorems

We give only two examples (see [90, p. 418]).

Theorem 6.5. *If* $\mathrm{NSPACE}(\log n) = \mathrm{DSPACE}(\log n)$, *then* $\mathrm{NSPACE}(n) = \mathrm{DSPACE}(n)$.

Proof (*sketch*). If $A \in \mathrm{NSPACE}(n)$, then it is easy to see that

$$B = \{x01^k : x \in A \quad \text{and} \quad k = 2^{|x|} - |x| - 1\} \in \mathrm{NSPACE}(\log n).$$

Thus, by hypothesis $B \in \mathrm{DSPACE}(\log n)$, and from that it follows easily that $A \in \mathrm{DSPACE}(n)$. □

This proof generalizes to $S(n) \geqslant \log n$. Another method of proof is needed for the following theorem of Szepietowski [84, 87].

Theorem 6.6. *If* $\mathrm{NSPACE}(\log\log n) = \mathrm{DSPACE}(\log\log n)$, *then*

$$\mathrm{NSPACE}(\log n) = \mathrm{DSPACE}(\log n).$$

Remark 6.7. The question whether $\mathrm{DSPACE}(\log n) = \mathrm{NSPACE}(\log n)$ is probably the most important open problem concerning space complexity (see a discussion in [90, pp. 420–423]). Efforts to solve it often lead to equivalent formulations of the problem within a different framework.

As an example, complexity classes can be characterized by a first-order logic together with an operator. Immerman [38] gives such a characterization for DSPACE($\log n$) and NSPACE($\log n$). This approach led him to prove the closure of NSPACE($\log n$) under complementation [39] (see Theorem 5.1) at first within this logical framework.

7. Weakly space-bounded classes

Definition 7.1. $L \in \mathrm{DSPACE}_w(S(n))$ (resp. $\mathrm{NSPACE}_w(S(n))$) if there is a DTM (resp. NTM) M such that $L(M) = L$ and for all $x \in L(M)$ there is an accepting computation of M on x during which M works in space $S(n)$.

Remarks 7.2. These space-bounded classes are called weakly space-bounded classes. By contrast, usual space bounds are also called strong space bounds.

There are two differences here with the definition of strongly space-bounded classes. Firstly, the space bound is required here for all $x \in L(M)$, and not for all $x \in \Sigma^*$. Secondly, it is required for an accepting computation, and not for every accepting computation.

For any complexity measure $\mathscr{C}$, the inclusion $\mathscr{C}(S(n)) \subseteq \mathscr{C}_w(S(n))$ is evident.

If $S(n)$ is f.s.c., then weakly and strongly space-bounded classes are the same. $S(n)$ cells can be marked on the work tape before any computation, and the machine rejects if this marked space is overstepped.

Lemma 7.3 (Alt and Mehlhorn [3]). $\{0^l10^m : l \neq m\} \in \text{DSPACE}_w(\log\log n)$.

Proof. Let M be the DTM which on input x verifies that $x = 0^l10^m$, l, m natural numbers, and then computes l and m modulo k for $k = 1, 2, \ldots$ until it finds a natural number k such that $l \not\equiv m (\bmod k)$. By Lemma 2.4, if $l \neq m$, such a k satisfies $k \leqslant c \log(l+m)$ for a constant c. The DTM M writes successively $1, 2, \ldots, k$ in binary on its work tape, and uses space $O(\log k)$. So, on an accepted input, M uses space $O(\log\log(l+m)) = O(\log\log|x|)$.

Theorem 7.4 (Chang et al. [16]). *If* $\log\log n \leqslant S(n) = o(\log n)$, *then* $\text{DSPACE}_w(S(n))$ *and* $\text{NSPACE}_w(S(n))$ *are not closed under complementation.*

Proof (*sketch*). By Lemma 7.3, $\{0^l10^m : l \neq m\} \in \text{DSPACE}_w(\log\log n)$. But $\{0^l10^l : l \geqslant 1\} \notin \text{NSPACE}_w(\log\log n)$, and $\text{DSPACE}_w(S(n))$ and $\text{NSPACE}_w(S(n))$ are closed under intersection with regular sets. □

Theorem 7.5. *If* $S(n) = o(\log\log n)$, *then*

(i) $\text{Reg} = \text{DSPACE}_w(S(n)) \subsetneq \text{DSPACE}_w(\log\log n)$,

(ii) $\text{Reg} = \text{NSPACE}_w(S(n)) \subsetneq \text{NSPACE}_w(\log\log n)$.

Proof (*sketch*). Alberts [2] proves that $\text{Reg} = \text{NSPACE}_w(S(n))$. The theorem follows from Lemma 7.3. □

Theorem 7.6 (Alt and Mehlhorn [3], Chang et al. [16]). *If* $\log\log n \leqslant S(n) = o(\log n)$, *then* $\text{DSPACE}(S(n)) \subsetneq \text{DSPACE}_w(S(n))$.

Proof (*sketch*). From Lemma 7.3 and $\{0^l10^m : l \neq m\} \notin \text{DSPACE}(o(\log n))$. □

Theorem 7.7 (Geffert [24]). *If* $\log\log n \leqslant S(n) = o(\log n)$, *then* $\text{NSPACE}(S(n)) \subsetneq \text{NSPACE}_w(S(n))$.

This theorem is a consequence of Theorem 3.7, as is the fact that $\{0^l10^m : l \neq m\} \notin \text{NSPACE}(o(\log n))$. See also Szepietowski [86], who discussed implications between these assertions before they were proved.

Remarks 7.8. The theorems of this section leave many problems open for the classes $\mathrm{DSPACE}_w(S(n))$ and $\mathrm{NSPACE}_w(S(n))$ when $S(n)$ is not f.s.c.: closure under complementation if $S(n) \geq \log n$, hierarchy theorems, relationship between determinism and nondeterminism, relationship between strong and weak classes if $S(n) \geq \log n$.

8. One-way machines

Up to this point, Turing machines have been supposed to be two-way, i.e., the input tape head could move in both directions. If we require that the input tape head reads the input from left to right only, we get *one-way* Turing machines (1-DTM and 1-NTM). The one-way complexity class corresponding to the two-way class $\mathscr{C}$ is denoted by 1-$\mathscr{C}$.

Evidently 1-$\mathscr{C} \subseteq \mathscr{C}$. Furthermore, if $S(n) \geq n$, then one-way and two-way classes are the same, for the space is sufficient to copy the entire input once for all on the work tape and then read it in both directions.

One-way classes have the same properties as two-way classes with regard to the number of tapes, the number of heads and linear speed-up (see [90, pp. 70, 330]).

Properties of classes 1-DSPACE$(S(n))$ *and* 1-NSPACE$(S(n))$

Theorem 8.1 (Hopcroft and Ullman [34, p. 170]). *For any function $S(n)$ and any 1-DTM (resp. 1-NTM) M working in space $S(n)$, there is a 1-DTM (resp. 1-NTM) M' working in space $S(n)$, accepting the same language as M, and halting on every input.*

Lemma 8.2. *Let $L = \{u\#v\colon u, v \in \{0,1\}^*, u \neq v\}$. Then*

(i) $L \in$ 1-NSPACE$(\log n)$,

(ii) $L \in$ DSPACE$(\log n)$,

(iii) $L \notin$ 1-DSPACE$(o(n))$.

Proof (*sketch*). (i) L is accepted by a 1-NTM which guesses in binary the position of the first letter where u and v differ, and then verifies its guess by reading the input.

(ii) L is accepted by a DTM which compares one by one the letters of u and v, noting their positions in binary on the work tape.

(iii) By an elementary counting argument. If $S(n) = o(n)$, then for any q and any sufficiently large n, $q^{S(n)} < 2^n$. Therefore, there are strings u_1, u_2, $u_1 \neq u_2$, such that a 1-DTM M working in space $S(n)$ is in the same configuration after reading $u_1\#$ and $u_2\#$. But then $u_1\#u_1$ and $u_2\#u_1$ are either both accepted or both rejected, and M does not accept L. See [32, p. 384] or [90, p. 121]. □

Theorem 8.3 (Hopcroft and Ullman [34, p. 175]). (i) *For any function $S(n)$,* 1-DSPACE($S(n)$) *is closed under union, intersection and complementation.*

(ii) *For any function $S(n)$,* 1-NSPACE($S(n)$) *is closed under union and intersection.*

(iii) *If $\log n \leqslant S(n) = \mathrm{o}(n)$, then* 1-NSPACE($S(n)$) *is not closed under complementation.*

From Lemma 8.2 and the closure of 1-DSPACE($S(n)$) by complementation, it follows that

$$\{u \# v\colon u, v \in \{0,1\}^*,\ u \neq v\} \in \text{1-NSPACE}(\log n) - \text{co-1-NSPACE}(\mathrm{o}(n)).$$

Hence the third part of Theorem 8.3.

Theorem 8.4 (Stearns et al. [79], Wagner and Wechsung [90, p. 313]). *If S_2 is f.s.c., $S_2(n) \geqslant \log n$, and $S_1(n) = \mathrm{o}(S_2(n))$, then*

$$\text{1-DSPACE}(S_1(n)) \subsetneq \text{1-DSPACE}(S_2(n)).$$

No such result is known for 1-NSPACE.

Theorem 8.5 (Stearns et al. [79], Hopcroft and Ullman [34, p. 172], Wagner and Wechsung [90, p. 313]). *If $S(n) = \mathrm{o}(\log n)$, then*

(i) Reg = 1-DSPACE($S(n)$) $\subsetneq$ 1-DSPACE($\log n$),

(ii) Reg = 1-NSPACE($S(n)$) $\subsetneq$ 1-NSPACE($\log n$).

The strict inclusions follow from $\{0^n 1^n\colon n \in \mathbb{N}\} \in$ 1-DSPACE($\log n$) − Reg.

Relationship between determinism and nondeterminism

For any function $S(n)$, 1-DSPACE($S(n)$) $\subseteq$ 1-NSPACE($S(n)$). The situation is clearer than for two-way classes, as is shown by the following theorem.

Theorem 8.6. (i) *If $\log n \leqslant S(n) = \mathrm{o}(n)$, then* 1-DSPACE($S(n)$) $\subsetneq$ 1-NSPACE($S(n)$).

(ii) *If $\log n \leqslant S(n) = \mathrm{o}(\sqrt{n})$, then* 1-NSPACE($S(n)$) $\nsubseteq$ 1-DSPACE($S(n)^2$).

Both parts of this theorem are direct consequences of Lemma 8.2: $\{u \# v\colon u, v \in \{0,1\}^*,\ u \neq v\} \in$ 1-NSPACE($\log n$) − 1-DSPACE($\mathrm{o}(n)$). The Savitch theorem (Theorem 6.2) is therefore false for one-way classes if $S(n) = \mathrm{o}(\sqrt{n})$. It becomes true if $S(n) \geqslant \sqrt{n}$, because

$$\text{1-NSPACE}(S(n)) \subseteq \text{NSPACE}(S(n)) \subseteq \text{DSPACE}(S(n)^2)$$
$$= \text{1-DSPACE}(S(n)^2).$$

Relationship between one-way and two-way machines

Theorem 8.7. *If* $\log n \leqslant S(n) = o(n)$, *then*
(i) $1\text{-DSPACE}(S(n)) \subsetneq \text{DSPACE}(S(n))$,
(ii) $1\text{-NSPACE}(S(n)) \subsetneq \text{NSPACE}(S(n))$.

This follows from

$$\{w \# w : w \in \{0,1\}^*\} \in \text{DSPACE}(\log n) - 1\text{-NSPACE}(o(n))$$

(see [90, pp. 107, 121]). We recall that:

- If $S(n) = o(\log\log n)$, then

$$\text{Reg} = 1\text{-DSPACE}(S(n)) = 1\text{-NSPACE}(S(n)) = \text{DSPACE}(S(n)) = \text{NSPACE}(S(n)).$$

- If $\log\log n \leqslant S(n) = o(\log n)$, then

$$\text{Reg} = 1\text{-DSPACE}(S(n)) = 1\text{-NSPACE}(S(n)) \subsetneq \text{DSPACE}(S(n)) \subseteq \text{NSPACE}(S(n)).$$

- If $S(n) \geqslant n$, then

$$1\text{-DSPACE}(S(n)) = \text{DSPACE}(S(n)) \subseteq 1\text{-NSPACE}(S(n)) = \text{NSPACE}(S(n)).$$

Weakly space-bounded one-way machines

Lemma 8.8 (Freivalds [20]). $\{0^l 1 0^m : l \neq m\} \in 1\text{-NSPACE}_w(\log\log n)$.

The proof uses Lemma 2.4, similarly to Lemma 7.3.

Theorem 8.9 (Chang et al. [16, p. 7]). *If* $\log\log n \leqslant S(n) = o(\log n)$, *then* $1\text{-NSPACE}_w(S(n))$ *is not closed under complementation.*

Proof (*sketch*). From Lemma 8.8 and $\{0^l 1 0^l : l \geqslant 1\} \notin 1\text{-NSPACE}_w(o(\log n))$. □

The analogous problem for 1-DSPACE_w is open.

Theorem 8.10 (Alberts [2]). *If* $S(n) = o(\log\log n)$, *then*

$$\text{Reg} = 1\text{-NSPACE}_w(S(n)) \subsetneq 1\text{-NSPACE}_w(\log\log n).$$

The analogous problem for 1-DSPACE_w is open.

Theorem 8.11. *If* $\log\log n \leqslant S(n) = o(\log n)$, *then*
(i) $1\text{-DSPACE}_w(S(n)) \subsetneq \text{DSPACE}_w(S(n))$ (*cf.* [3]),
(ii) $1\text{-NSPACE}_w(S(n)) \subsetneq \text{NSPACE}_w(S(n))$ (*cf.* [16]),
(iii) $1\text{-NSPACE}(S(n)) (= \text{Reg}) \subsetneq 1\text{-NSPACE}_w(S(n))$ (*cf.* [20]).

The inclusions are strict by considering $\{\text{bin}(0)\#\text{bin}(1)\#\cdots\#\text{bin}(k)\colon k\in\mathbb{N}\}$, which is in DSPACE$(\log\log n)$ ([79, 90, p. 131]; see Theorem 4.10), but not in 1-NSPACE$_w(\text{o}(\log n))$ [16, p. 6].

The problem (Reg $=$) 1-DSPACE$(S(n)) \subsetneq ?$ 1-DSPACE$(S(n))$, for $S(n)=\text{o}(\log n)$, is open.

Remark 8.12. For lower-bound results for the measure 1-DSPACE, see [55].

9. Alternating machines

Definitions 9.1 (*Chandra et al.* [15]). An *alternating Turing machine* (ATM) is a generalization of a nondeterministic Turing machine.

The set of states is partitioned in the four subsets of *universal* states, *existential* states, *accepting* states and *rejecting* states. A configuration is universal (resp. existential, accepting, rejecting) if the state associated to this configuration is universal (resp. existential, accepting, rejecting). Given an input x, we can consider the tree of all possible computations of an ATM M on x. The nodes of this tree are labeled by configurations according to the (multivalued) transition function δ. The root is the initial configuration.

A *computation tree* of M on x is a subtree of this tree with the same root, such that the children of a node labeled by a universal (resp. existential) configuration consist of all (resp. one) of the immediate successors of that configuration. A computation tree is *accepting* if it is finite and all the leaves are accepting configurations. An ATM M *accepts* x if there is an accepting computation tree of M on x. $L(M)$ denotes the set of words accepted by M.

M works in time $T(n)$ if for every input x, all computation trees of M on x have height at most $T(n)$. M works in (strong) space $S(n)$ if for every input x all the computation trees of M on x use space at most $S(n)$. M works in weak space $S(n)$ if for every $x\in L(M)$ there is a computation tree of M on x using space at most $S(n)$. Szepietowski [83] introduces an intermediary type: for every $x\in L(M)$, all the computation trees of M on x use space at most $S(n)$.

Alternating Turing machines can be one-way or two-way. Therefore, four new measures of complexity can be defined: ASPACE, ASPACE$_w$, 1-ASPACE and 1-ASPACE$_w$. We now survey their properties and their relationship with measures introduced in the previous sections.

Properties of alternating complexity classes

Theorem 9.2 (Chandra et al. [15]). *If* $S(n)\geq\log n$, *then* ASPACE$_w(S(n))=$ DTIME$(2^{\text{O}(S(n))})$.

For a proof, see [15] or [9, pp. 74–76]. In particular, $\mathrm{ASPACE_w}(\log n) = \mathrm{DTIME}(n^{\mathrm{O}(1)}) = \mathrm{P}$. Therefore, alternating space complexity class $\mathrm{ASPACE_w}(S(n))$, for $S(n) \geqslant \log n$, amounts to deterministic time complexity classes and its full study leads us beyond the scope of this paper.

Theorem 9.3 (Wagner and Wechsung [90, p. 317]). *If S_2 is f.s.c., $S_2 \geqslant \log n$ and $S_1(n) = \mathrm{o}(S_2(n))$, then*

$$\mathrm{ASPACE_w}(S_1(n)) \subsetneq \mathrm{ASPACE_w}(S_2(n)).$$

Theorem 9.4. (i) *If $S(n) = \mathrm{o}(\log\log n)$, then* [80]

$$\mathrm{Reg} = \mathrm{ASPACE}(S(n)) \subsetneq \mathrm{ASPACE}(\log\log n).$$

(ii) *If $S(n) = \mathrm{o}(\log\log n)$, then* [2]

$$\mathrm{Reg} = \mathrm{ASPACE_w}(S(n)).$$

(iii) *If $S(n) = \mathrm{o}(\log n)$, then* [16]

$$\mathrm{Reg} = \text{1-}\mathrm{ASPACE}(S(n)) \subsetneq \text{1-}\mathrm{ASPACE}(\log n).$$

(iv) *If $S(n) = \mathrm{o}(\log\log n)$, then* [2]

$$\mathrm{Reg} = \text{1-}\mathrm{ASPACE_w}(S(n)) \subsetneq \text{1-}\mathrm{ASPACE_w}(\log\log n).$$

Relationship between complexity classes

The following theorem completes Theorems 7.6 and 8.11.

Theorem 9.5. *If $\log\log n \leqslant S(n) = \mathrm{o}(\log n)$, then*

(i) $\mathrm{NSPACE}(S(n)) \subsetneq \text{1-}\mathrm{ASPACE_w}(S(n))$,
(ii) $\mathrm{NSPACE}(S(n)) \subsetneq \mathrm{ASPACE}(S(n))$,
(iii) $\text{1-}\mathrm{NSPACE_w}(S(n)) \subsetneq \text{1-}\mathrm{ASPACE_w}(S(n))$,
(iv) $\mathrm{NSPACE_w}(S(n)) \subsetneq \mathrm{ASPACE_w}(S(n))$,
(v) $\text{1-}\mathrm{ASPACE_w}(S(n)) \subsetneq \mathrm{ASPACE_w}(S(n))$.

Proof (*sketch*). All inclusions are trivial, except the first one: see [16, erratum]. The fact that the first four inclusions are strict comes from considering the language $\{1^n : \forall m < n \; F(m) < F(n)\}$, where $F(n)$ is the smallest positive integer which does not divide n. This language is in $\mathrm{ASPACE}(\log\log n)$ and $\text{1-}\mathrm{ASPACE_w}(\log\log n)$, but not in $\mathrm{NSPACE_w}(\mathrm{o}(\log n))$ (see [16, p. 3]). The fact that the last inclusion is strict is due to [40] (see also [16, erratum]). □

It is open whether the inclusion of $\mathrm{ASPACE}(S(n))$ in $\mathrm{ASPACE_w}(S(n))$ is strict for $\log\log n \leqslant S(n) = \mathrm{o}(\log n)$.

Theorem 9.6 (Ladner et al. [52]). *If $S(n) \geqslant \log n$, then*

$$\mathrm{ASPACE}(S(n)) = 1\text{-}\mathrm{ASPACE}(S(n)),$$

$$\mathrm{ASPACE}_w(S(n)) = 1\text{-}\mathrm{ASPACE}_w(S(n)).$$

Proof. If L is accepted by an ATM M working in space $S(n)$, then L is accepted by the 1-ATM M' which records in binary on the work tape the position of the input tape head, simulates M step by step, and for each step

- existentially guesses the symbol read by the input tape head of M, and then
- universally,
  - goes on the simulation with the guessed symbol as symbol read by the input tape head of M,
  - verifies that the guessed symbol is the right one by moving its input tape head on the recorded position of the input tape head.

Such an M' moves the input tape head only on this last verification, so is one-way. M' uses for the recording of the position of the input tape head a space $\log n \leqslant S(n)$, so works in space $S(n)$.

Remark 9.7. If $S(n)$ is f.s.c., then $\mathrm{ASPACE}(S(n)) = \mathrm{ASPACE}_w(S(n))$, as in Remark 7.2.

Definition 9.8. The *number of alternations* of a computation tree of an ATM is the maximum number of changes of type of configuration (i.e., universal or existential) along a branch of the tree. $\mathrm{ASPACE\text{-}ALT}(S(n), A(n))$ denotes the class of languages accepted by ATMs working in space $S(n)$, with at most $A(n)$ alternations. $\Pi_k\mathrm{SPACE}(S(n))$ denotes the class of languages accepted by ATMs working in space $S(n)$, with at most $k-1$ alternations, and beginning in a universal state.

Remarks 9.9. It is proved in [15] that if $S(n) \geqslant \log n$, then

$$\mathrm{ASPACE\text{-}ALT}(S(n), A(n)) \subseteq \mathrm{DSPACE}(S(n)(S(n) + A(n))).$$

If $S(n) \geqslant \log n$, then it follows from the Immerman–Szelepcsényi theorem (Theorem 5.1) that

$$\mathrm{ASPACE\text{-}ALT}(S(n), \mathrm{O}(1)) = \bigcup_{k \in \mathbb{N}} \Pi_k\mathrm{SPACE}(S(n)) = \mathrm{NSPACE}(S(n)).$$

Theorem 9.10 (Szepietowski [85]). *If $\log\log n \leqslant S(n) = \mathrm{o}(\log n)$, then*

(i) $\mathrm{NSPACE}(S(n)) \subsetneq \Pi_2\mathrm{SPACE}(S(n))$,

(ii) $\mathrm{NSPACE}_w(S(n)) \subsetneq \Pi_2\mathrm{SPACE}_w(S(n))$.

In fact, if $S(n)$ is f.s.c., $\limsup S(n) = \infty$ and $S(n) = \mathrm{o}(\log n)$, then there is a language $L \subseteq \{1\}^*$ such that $L \in \Pi_2\mathrm{SPACE}(S(n)) - \mathrm{NSPACE}_w(\mathrm{o}(\log n))$.

The following result from Chandra et al. [15] is used in the remarks following Theorems 6.2 and 10.6.

Theorem 9.11. *If* $S(n) \geqslant \log n$, *then*

(i) $\text{ATIME}(S(n)) \subseteq \text{DSPACE}(S(n))$,

(ii) $\text{NSPACE}(S(n)) \subseteq \text{ATIME}(\text{O}(S(n)^2))$.

For a proof, see [15, 9, pp. 70–73].

Remark 9.12. Together with Theorem 9.2, Theorem 9.11 provides a nice relationship between deterministic and alternating time and space complexity classes. The deterministic hierarchy shifts by one level when we move to alternating complexity classes, according to the following inclusions:

$$\begin{aligned}&\text{DSPACE}(\log n)\\ &\quad\subseteq \text{DTIME}(n^{\text{O}(1)}) = \text{ASPACE}(\log n)\\ &\quad\subseteq \text{DSPACE}(n^{\text{O}(1)}) = \text{ATIME}(n^{\text{O}(1)})\\ &\quad\subseteq \text{DTIME}(2^{n^{\text{O}(1)}}) = \text{ASPACE}(n^{\text{O}(1)})\\ &\quad\subseteq \text{DSPACE}(2^{n^{\text{O}(1)}}) = \text{ATIME}(2^{n^{\text{O}(1)}}).\end{aligned}$$

10. Probabilistic machines

Definitions 10.1 (*Gill* [25]). A *probabilistic Turing machine* (*PTM*) is a NTM with only binary branching, which, at each step, equiprobably chooses between these two possibilities. $p_M(x)$ denotes the probability that the PTM M reaches an accepting state on input x. At least three types of acceptance of a language L by a PTM can be defined.

Let M be a PTM. The language accepted by M according to the *probabilistic acceptance* is $L \subseteq \Sigma^*$ such that $\forall x \in \Sigma^*\ x \in L \Leftrightarrow p_M(x) > 1/2$. $\text{PrSPACE}(S(n))$ denotes the class of languages accepted, according to the probabilistic acceptance, by a PTM working in space $S(n)$.

The PTM M accepts L according to the *bounded error probabilistic acceptance* if there is a constant $c > 0$ such that

$$\begin{aligned}\forall x \in \Sigma^* \quad & x \in L \Leftrightarrow p_M(x) > \tfrac{1}{2} + c,\\ & x \notin L \Leftrightarrow p_M(x) < \tfrac{1}{2} - c.\end{aligned}$$

$\text{BPrSPACE}(S(n))$ denotes the class of languages accepted according to the bounded error probabilistic acceptance, by a PTM working in space $S(n)$.

If the *error* of a PTM M on a language L is defined by

$$e_{M,L} = \sup(\{p_M(x) : x \notin L\} \cup \{1 - p_M(x) : x \in L\}),$$

then M accepts L according to the bounded error acceptance iff $e_{M,L} < 1/2$.

The PTM M accepts L according to the *random acceptance* if

$$\forall x \in \Sigma^* \quad x \in L \Leftrightarrow p_M(x) > \tfrac{1}{2},$$
$$x \notin L \Leftrightarrow p_M(x) = 0.$$

RSPACE($S(n)$) denotes the class of languages accepted, according to the random acceptance, by a PTM working in space $S(n)$.

Remarks 10.2. It is not assumed that the computations of a PTM stop on a given input. We shall see (Theorem 10.3) that a PTM can always be modified in order that its computations on a given input stop with probability one.

If we require in the definitions that every computation stops, then other classes are defined: see [91, 48, 49].

A PTM can also be defined as a DTM with a tape on which is written before any computation a random infinite sequence of 0's and 1's, together with a one-way head. This definition, introduced by Borodin et al. [12], is equivalent to the definition above. Karpinski and Verbeek [46, 47] have studied what happens if the tape with the random sequence of 0's and 1's is two-way. Then the machine is more powerful. For example, with obvious notations, if $S(n) \geq \log n$, then

$$\mathrm{Pr_2SPACE}(S(n)) = \mathrm{PrSPACE}(2^{O(S(n))}).$$

Theorem 10.3 (Simon [72], Ruzzo et al. [65]). *If $S(n)$ is space constructible, $S(n) \geq \log n$, and if M is a PTM working in space $S(n)$, then there is a PTM M' which accepts the same language as M, according to the same type of acceptance (probabilistic, bounded-error probabilistic, random), and halts on all inputs with probability one, working in expected time $2^{2^{O(S(n))}}$.*

Proof (*sketch*). By a result of Gill [25, pp. 692–693], there is a constant c such that for every input x, if $p_M(x) > 1/2$, then $p_M(x) > 1/2 + 2^{-c^{S(n)}}$. There is a constant d such that the number of configurations of M on inputs of length n is at most $d^{S(n)}$.

Then let M' be the PTM which, on an input x of length n, simulates M on x, accepting if M enters an accepting configuration, and rejecting with probability $2^{-(c+d)^{S(n)}}$ at every interval of $d^{S(n)}$ steps. Then it is easy to see that M' works in space $O(S(n))$ and expected time $2^{2^{O(S(n))}}$, halts with probability one and does not accept any input that M does not accept. A deeper analysis shows that M' accepts any input that M accepts, according to the same type of acceptance.

Theorem 10.4 (Simon [72], Ruzzo et al. [65]). *If $S(n)$ is space constructible and $S(n) \geq \log n$, then* PrSPACE($S(n)$) *and* BPrSPACE($S(n)$) *are closed under complementation.*

Proof (*sketch*). Given a PTM M, a PTM M' that accepts the complement of the language accepted by M is obtained by swapping accepting and rejecting states, granted that the two following conditions are satisfied. Firstly, M halts with probability one, which can be achieved by Theorem 10.3. Secondly, for any input x, $p_M(x) \neq 1/2$, which can be achieved by the following technique of Gill [25, p. 686]. M is replaced by a PTM which runs with equal probability the PTM M and the PTM accepting each input with probability $1/2 - 2^{-c^{S(n)}}$, where c is the constant from Theorem 10.3. □

Theorem 10.5 (Gill [25]). *If $S(n)$ is f.s.c. and $S(n) \geq \log n$, then* RSPACE($S(n)$) = NSPACE($S(n)$).

Theorem 10.6 (Borodin et al. [12]). *If $S(n)$ is f.s.c. and $S(n) \geq \log n$, then* PrSPACE($S(n)$) ⊆ DSPACE($S(n)^2$).

Remarks 10.7. Gill [25] first proved that PrSPACE($S(n)$) ⊆ DSPACE($2^{O(S(n))}$). Then it was asserted in [71] that PrSPACE($S(n)$) ⊆ DSPACE($S(n)^{O(1)}$), in [73] that PrSPACE($S(n)$) ⊆ DSPACE($S(n)^6$) and, finally, in [41] that PrSPACE($S(n)$) ⊆ DSPACE($S(n)^2$), but these last three papers used incorrectly a formula of Csanky [18].

Regarding probabilistic transducers (i.e., Turing machines computing functions), it was asserted in [74, 26] that FPrSPACE($S(n)$) ⊆ FDSPACE($S(n)^{36}$), and in [41] that FPrSPACE($S(n)$) ⊆ FDSPACE($S(n)^4$). Borodin et al. [12] prove that FPrSPACE($S(n)$) ⊆ FDSPACE($S(n)^2$).

Borodin et al. [12] actually prove that if $S(n)$ is f.s.c. and $S(n) \geq \log n$, then PrSPACE($S(n)$) ⊆ ASPACE-TIME($S(n), S(n)^2$) (and see Theorem 9.11).

Remarks 10.8. It is not known whether the inclusion DSPACE($S(n)$) ⊆ PrSPACE($S(n)$) is strict. Ajtai et al. [1] give a condition for equality when $S(n) = \log n$. Jung [42] proves that PrSPACE($\log n$) = PrTIME-SPACE($n^{O(1)}, \log n$). Furst et al. [22] give a link between, on the one hand, some relations between probabilistic complexity classes and, on the other hand, the existence of pseudorandom number generator satisfying specific conditions. Freivalds [21] scrutinizes one-way probabilistic machines working in space $\log\log n$ or $o(\log\log n)$. Karpinski and Verbeek [48, 49] study another type of acceptance and the measure MSPACE it defines.

11. Nonuniform space-bounded classes

Introduction

Up to this point, algorithms, i.e., Turing machines, work the same way on every input: they are uniform algorithms with respect to input length. We can consider a family $\varphi = (\varphi_n)_{n \in \mathbb{N}}$ of algorithms such that φ_n is defined on the set Σ^n of words with length n. Then the complexity of φ is defined from the complexity of the φ_n.

For example, let φ_n be a Boolean circuit computing a Boolean function from $\{0,1\}^n$ to $\{0,1\}$. φ_n consists in a directed acyclic graph whose nodes are labeled by n Boolean variables (input nodes, with indegree 0), or by $\wedge$, $\vee$ or $\neg$. For different input lengths n, m, φ_n and φ_m can be very different. The size of φ_n is the number of nodes. A family of circuits $\varphi = (\varphi_n)_{n\in\mathbb{N}}$ accepts a language $L \subseteq \{0,1\}^*$ if for every n the Boolean function computed by φ_n is the characteristic function of $L \cap \{0,1\}^n$. The size complexity of a language $L \subseteq \{0,1\}^*$ is the size of a family of circuits with smallest sizes that accepts L.

Then there are sets with very small size complexity that are not recursive. For example, let A be a nonrecursive subset of $\mathbb{N}$, and $L = \{1^n\colon n \in A\}$. L is accepted by $(\varphi_n)_{n\in\mathbb{N}}$ such that $\forall x \in \Sigma^n$ $\varphi_n(x) = 1$ if $n \in A$, and $\varphi_n(x) = 0$ if $n \notin A$. φ_n is constant on Σ^n, so L has a very small nonuniform complexity, and yet L is not recursive.

Now, how to make possible the comparison between uniform and nonuniform complexity measures? One way is to uniformize circuits, for instance by requiring the function $1^n \to \text{code}(\varphi_n)$ to be computable in logarithmic space. Another way is to define nonuniform space complexity for Turing machines. This can be done in many ways. We shall present three of them.

Definition of nonuniform classes by Turing machines with advice

Definition 11.1. Let $\mathscr{C}$ be a class of languages, and let F be a class of functions from $\mathbb{N}$ to Σ^*. Let $A \subseteq \Sigma^*$. Then $A \in \mathscr{C}/F$ if there is a language B in $\mathscr{C}$ and a function h in F such that $A = \{x \in \Sigma^*\colon (x, h(|x|)) \in B\}$.

This definition amounts to allowing, next to the input, an "advice", word depending only on the length of the input, and satisfying some conditions. For example, if $\mathscr{C} = \text{PSPACE}$ and $F = \text{poly} =_{\text{def}} \{h\colon \mathbb{N} \to \Sigma^*\colon \exists p \text{ polynomial } \forall n \in \mathbb{N}\ |h(n)| \leqslant p(n)\}$, we obtain the nonuniform class PSPACE/poly.

Remarks 11.2. Schnorr [68, p. 100] seems to be the first to introduce nonuniform complexity classes, by defining "relative" Turing machines. These machines have a supplementary tape with a word $w \in \{0,1\}^*$ which is called oracle and can be modified during the computation.

Pippenger [59] cites and modifies Schnorr [68]: on a read-only tape which is called a reference tape there is a word $g(x)$ depending only on the length of the input x.

Karp and Lipton [44, 45] define as above the complexity classes $\mathscr{C}/F$, attributing them to Pippenger [59] and Plaisted [60]. This definition is adopted by many authors (for example Meinel [56], Balcázar et al. [8]).

Cook [17, p. 82], considering again the definition of Pippenger [59], adds that the space $S(n)$ must verify $S(n) \geqslant \log(|g(x)|)$, in order that the position of the head of the reference tape can be written in binary on the work tape. For spaces $S(n) \geqslant \log n$ and functions g such that $|g(x)| \leqslant p(|x|)$, p polynomial, this condition is already fulfilled.

Definition of nonuniform classes by automata

Definition 11.3. Let $S(n)$ be a function and let $L \in \Sigma^*$ be a language. Then $L \in$NUDSPACE($S(n)$) (resp. NUNSPACE($S(n)$)) if there is a constant c and a sequence $(M_n)_{n \geq 1}$ of two-way deterministic (resp. nondeterministic) automata such that for every $n \geq 1$,

(i) M_n has at most $c^{S(n)}$ states,
(ii) $L(M_n) \cap \Sigma^n = L \cap \Sigma^n$.

Remarks 11.4. Cook [17, p. 82] seems to be the first to give this definition. Balcázar et al. [5, 6, 8, p. 123] consider one-way deterministic and nondeterministic automata.

Definition of nonuniform classes by oracle Turing machines

Definition 11.5. Let $S(n)$ be a function on $\mathbb{N}$, and let A and L be languages. Then $A \in \mathrm{DSPACE}^A(S(n))$ if there is a DTM M working in space $S(n)$ and having a $S(n)$ space-bounded oracle tape, and three states $q_?, q_{\text{yes}}, q_{\text{no}}$ such that, when M enters state $q_?$ with y on the oracle tape, then M goes to the state q_{yes} if $y \in A$, and q_{no} if $y \notin A$. It is sometimes said that M accepts L modulo A. Then $\bigcup_{A \subseteq \Sigma^*} \mathrm{DSPACE}^A(S(n))$ is a nonuniform complexity class. An analogous definition can be stated for nondeterministic classes.

Actually, this definition is too powerful, so additional conditions are given. Ruzzo [63, p. 381], inspired by Pippenger [59], seems to be the first to give a similar definition, but allows only one final call to the oracle. Balcázar et al. [5, 6] consider only sparse oracles, i.e., sets A such that there is a polynomial p such that for every n, $\mathrm{card}(A \cap \Sigma^n) \leq p(n)$. Furthermore, the oracle tape is not space-bounded.

Relationship between the preceding definitions

Little has been published on this matter. Nonetheless, the following theorems are known.

Theorem 11.6. *For classes of languages on the alphabet* $\{0, 1\}$,

$$\mathrm{PSPACE}/\{2^n\} = \mathrm{NUDSPACE}(2^n) = \bigcup_{A \text{ sparse}} \mathrm{DSPACE}^A(2^n) = \mathscr{P}(\{0, 1\}^*).$$

Theorem 11.7 (Balcázar et al. [6, p. 68]).

$$\mathrm{DSPACE}(\log n)/\mathrm{poly} = \mathrm{NUDSPACE}(\log n) = \bigcup_{A \text{ sparse}} \mathrm{DSPACE}^A(\log n),$$

where the oracle tape is not *space-bounded.*

Theorem 11.8 (Balcázar et al. [7]).

$$\text{PSPACE/poly} = \bigcup_{A\ sparse} \text{PSPACE}^A,$$

where the oracle tape is *space-bounded.*

Properties of nonuniform classes

Nonuniform classes inherit some properties of uniform classes.

Theorem 11.9 (Ibarra and Ravikumar [37, p. 10]). *For any function* $S(n)$, NUDSPACE($S(n)$) *is closed under complementation.*

Theorem 11.10 (Cook [17, p. 83]). *For any function* $S(n) \geqslant \log n$, NUNSPACE($S(n)$) $\subseteq$ NUDSPACE($S(n)^2$).

Relationship between uniform and nonuniform classes

We have

$$\text{DSPACE}(S(n)) \subseteq \text{NUDSPACE}(S(n)).$$

Theorem 11.11 (Ibarra and Ravikumar [37, pp. 12, 15]).
(i) $\{0^n1^n\colon n \geqslant 1\} \in \text{NUDSPACE}(\log\log n) - \text{DSPACE}(o(\log n))$.
(ii) $\text{DSPACE}(2^{n^{O(1)}}) - \bigcup_{\varepsilon<1} \text{NUDSPACE}(n^\varepsilon) \neq \emptyset$.

12. Relativizations

Definitions 12.1. An oracle Turing machine M has an additional tape, the oracle tape, and three states $q_?, q_{\text{yes}}, q_{\text{no}}$. When M enters state $q_?$ (M is said to query the oracle), then M goes to the state q_{yes} or to the state q_{no} according to whether the word written on the oracle tape belongs or does not belong to a set called the oracle. Then the oracle tape is erased. The language accepted by the oracle Turing machine M with oracle A is denoted by $L^A(M)$. The class of languages accepted by a deterministic oracle Turing machine working in space $S(n)$, with oracle A, is denoted by DSPACE^A($S(n)$), and similarly for nondeterministic oracle Turing machines and probabilistic oracle Turing machines.

This definition is satisfactory for time complexity measures, but a major problem arises for space complexity measures: do we bound the space of the oracle tape? Various answers have been suggested. For a thorough discussion see Ruzzo et al. [64, 65], Buss [13, 14], and Hartmanis [27, 28].

First definition: no bound on the oracle tape

Ladner and Lynch [53] do not bound the space of the oracle tape. It amounts to bounding it by a space $2^{O(S(n))}$ (see the definition of Savitch [67, p. 230]). A nondeterministic oracle Turing machine can query $2^{2^{O(S(n))}}$ words in the tree of all possible computations. With this definition, neither Theorem 5.1, nor Theorem 6.2, nor Theorem 10.6 relativizes. Precisely, we have the following theorem.

Theorem 12.2. *If $S(n)$ is f.s.c., $S(n) \geqslant \log n$, then*
(i) $\exists A$ NSPACE$^A(S(n))$ *is not closed under complementation,*
(ii) $\exists A$ NSPACE$^A(S(n)) \nsubseteq$ DSPACE$^A(S(n)^2)$,
(iii) $\exists A$ PrSPACE$^A(S(n)) \nsubseteq$ DSPACE$^A(S(n)^2)$.

See for (i) Hartmanis [28, p. 87], for (ii) Ladner and Lynch [53, p. 28], for (iii) Buss [14, p. 356].

Second definition: same bound for every tape

This definition has been chosen by Simon and, subsequently, by Book and his followers (see, for example, Book [10], Book et al. [11]).

With this definition, the three results of Theorem 12.2 do relativize. The problem is that if $A \in$ DSPACE$(n) -$ DSPACE$(\log n)$, then $A \notin$ DSPACE$^A(\log n)$, which is counter-intuitive of what should be a good definition of relativization (see [65, p. 225]).

Third definition

Ruzzo et al. [64, 65] give a definition which is intermediate between the two previous ones:
– the oracle tape is not space-bounded,
– the machine works deterministically from the time it begins to write on the oracle tape, up to it queries the oracle, after which the oracle tape is erased.

Let $\mathscr{C}^{\langle A\rangle}$ denote the complexity class $\mathscr{C}$ with oracle A according to this third definition. Then DSPACE$^{\langle A\rangle}(S(n)) =$ DSPACE$^A(S(n))$ (without space bound), and $A \in$ DSPACE$^{\langle A\rangle}(S(n))$. The three results of Theorem 12.2 relativize (see, respectively, Hartmanis [28], Ruzzo et al. [65, p. 225], and Buss [14, p. 354]).

This definition has been followed for example by Gasarch [23], Kirsig and Lange [50], and Wilson [93]. As an example, we state a theorem of Hartmanis [28] (see also [31]), which generalizes results of Rackoff and Seiferas [61], Kirsig and Lange [50], and Wilson [93].

Theorem 12.3. *For space constructible $S(n) \geqslant \log n$,*

$$\mathrm{DSPACE}(\log n) = \mathrm{NSPACE}(\log n)$$

$$\Leftrightarrow \forall S(n)\ \forall A\ \mathrm{DSPACE}^{A}(S(n)) = \mathrm{NSPACE}^{A}(S(n))$$

(*with the second definition*)

$$\Leftrightarrow \forall S(n)\ \forall A\ \mathrm{DSPACE}^{\langle A\rangle}(S(n)) = \mathrm{NSPACE}^{\langle A\rangle}(S(n)).$$

Remark 12.4. Angluin [4] considers Turing machines working in space $S(n)$, with an auxiliary (unbounded) stack, and with an oracle tape which can be $S(n)$ space-bounded, $2^{O(S(n))}$ space-bounded, or unbounded.

13. Space complexity of RAMs

The RAM (random access machine) is a model of computation more similar to a real computer than a Turing machine. As we shall see, the space complexity defined by RAMs is closely related to the one defined by Turing machines. This shows the robustness of the notion of space complexity.

Definition 13.1 (*Wagner and Wechsung* [90, pp. 35–38]). A RAM consists of
- a two-way input tape as for a Turing machine,
- an infinite number of registers, numbered $0, 1, 2, \ldots$, containing natural numbers.
- an instruction counter containing a natural number, IC,
- a program operating on the registers and the instruction counter. A program consists in a finite sequence of instructions, numbered $1, 2, \ldots, m$. An instruction is one of the following, the meaning of which is clear (R_i denotes the contents of the register numbered i):
  - instructions inducing $\mathrm{IC} \leftarrow \mathrm{IC} + 1$:

$$\mathrm{READ}\ R_i$$

$$R_i \leftarrow w$$

$$R_i \leftarrow R_j$$

$$R_i \leftarrow R_{R_j}$$

$$R_{R_i} \leftarrow R_j$$

$$R_i \leftarrow R_j + R_k$$

$$R_i \leftarrow R_j \dot{-} R_k$$

- jump instruction:

  IF $R_i = 0$ THEN GOTO k

  inducing IC$\leftarrow k$ if $R_i = 0$ and IC$\leftarrow$IC$+1$ otherwise.
- instructions leaving IC unchanged:

  STOP
  ACCEPT
  REJECT

A nondeterministic RAM is defined by allowing two instructions to have the same number.

As for a Turing machine, are defined a computation, an accepting computation, and the language accepted by a RAM.

Definition 13.2 (*Space complexity of a RAM*). Given a RAM and an input, let $R_{j,t}$ be the content of the register j at time t of the computation. Let $|n|$ be the length of the positive number n written in binary, and $|0| = 0$. Let $\mathrm{sgn}(a) = 1$ if $a > 0$ and $\mathrm{sgn}(0) = 0$. Then the space used by the RAM on the input is

$$\sum_{j=0}^{\infty} (\max_{t \in \mathbb{N}} |R_{j,t}| + |j| \,\mathrm{sgn}(\max_{t \in \mathbb{N}} |R_{j,t}|)).$$

In other words, it is the sum of the lengths of maximum contents and of the lengths of the addresses of the registers used during the computation.

This allows to define the complexity measures RAM-DSPACE and RAM-NSPACE: RAM-DSPACE($S(n)$) (resp. RAM-NSPACE($S(n)$)) is the class of languages accepted by a deterministic (resp. nondeterministic) RAM working in space $S(n)$ on all inputs of length n.

Theorem 13.3. *For* $S(n) \geq n$, RAM-DSPACE(O($S(n)$))$=$DSPACE(O($S(n)$)), *and* RAM-NSPACE(O($S(n)$))$=$NSPACE(O($S(n)$)).

Proof (*sketch*) (*Wagner and Wechsung* [90, pp. 67–69]). In both directions, the simulation is done step by step.

A RAM can be simulated by a Turing machine that writes on its work tape the addresses and contents of the registers used during the computation of the RAM.

A Turing machine can be simulated by a RAM that uses a constant number of registers. A register contains a natural number encoding the word on the left of the work tape head. Another register contains a natural number encoding the word on the right of the work tape head. Then changing a symbol on the tape and moving the head is simulated by easy operations on the numbers contained in these two registers. These operations can be done with a constant number of registers. □

Theorem 13.3 can be formulated by saying that the space measures of RAMs and Turing machines are linearly related.

Remarks 13.4. If the space used by a RAM is defined by $\sum_{j=0}^{\infty} \max_{t \in \mathbb{N}} |R_{j,t}|$, i.e., the sum of the lengths of maximum contents of the registers used during the computation, then the deterministic measure is still linearly related to DSPACE, but the proof is intricate: see Slot and van Emde Boas [77, 78]. The same problem for the nondeterministic measure is open [78, p. 99]. For more discussions on space measures for RAMs, see also Wiedermann [92].

One-way Turing machines and RAMs are not linearly related (Slot and van Emde Boas [78]).

For space measures of other types of RAMs see van Emde Boas [89].

Acknowledgment

Many thanks to Etienne Grandjean for his enthusiasm, without which this paper would not have been possible.

References

[1] M. Ajtai, J. Komlós and E. Szemerédi, Deterministic simulation in LOGSPACE, in: *Proc. 19th Ann. ACM Symp. on Theory of Computing* (1987) 132–140.

[2] M. Alberts, Space complexity of alternating Turing machines, in: *FCT '85*, Lecture Notes in Computer Science, Vol. 199 (Springer, Berlin, 1985) 1–7.

[3] H. Alt and K. Mehlhorn, Lower bounds for the space complexity of context-free recognition, in: S. Michaelson and R. Milner, eds., *Automata, Languages and Programming, 3rd Int. Coll.* (Edinburgh Univ. Press, Edinburgh, 1976) 339–354.

[4] D. Angluin, On relativizing auxiliary pushdown machines, *Math. Systems Theory* **13** (1980) 283–299.

[5] J.L. Balcázar, J. Díaz and J. Gabarró, On some "non-uniform" complexity measures, in: *FCT '85*, Lecture Notes in Computer Science, Vol. 199 (Springer, Berlin, 1985) 18–27.

[6] J.L. Balcázar, J. Díaz and J. Gabarró, Uniform characterizations of non-uniform complexity measures, *Inform. and Control* **67** (1985) 53–69.

[7] J.L. Balcázar, J. Díaz and J. Gabarró, On non-uniform polynomial space, in: *Structure in Complexity Theory*, Lecture Notes in Computer Science, Vol. 223 (Springer, Berlin, 1986) 35–50.

[8] J.L. Balcázar, J. Díaz and J. Gabarró, *Structural Complexity I* (Springer, Berlin, 1988).

[9] J.L. Balcázar, J. Díaz and J. Gabarró, *Structural Complexity II* (Springer, Berlin, 1990).

[10] R.V. Book, On languages accepted by space-bounded oracle machines, *Acta Inform.* **12** (1979) 177–185.

[11] R.V. Book, C.B. Wilson and M.-R. Xu, Relativizing time, space, and time-space, *SIAM J. Comput.* **11** (1982) 571–581.

[12] A. Borodin, S. Cook and N. Pippenger, Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Inform. and Control* **58** (1983) 113–136.

[13] J.F. Buss, A theory of oracle machines, in: *Proc. 2nd Structure in Complexity Theory Conf.* (IEEE Computer Society Press, Los Alamitos, CA, 1987) 175–181.

[14] J.F. Buss, Relativized alternation and space-bounded computations, *J. Comput. System Sci.* **36** (1988) 351–378.

[15] A.K. Chandra, D.C. Kozen and L.J. Stockmeyer, Alternation, *J. ACM* **28** (1981) 114–133.

[16] J.H. Chang, O.H. Ibarra, B. Ravikumar and L. Berman, Some observations concerning alternating Turing machines using small space, *Inform. Process. Lett.* **25** (1987) 1-9. Erratum: *Inform. Process. Lett.* **27** (1988) 53.

[17] S.A. Cook, Towards a complexity theory of synchronous parallel computation, *Monographie de l'Enseignement Mathematique* **30** (1982) 75–100.

[18] L. Csanky, Fast parallel matrix inversion algorithms, *SIAM J. Comput.* **5** (1976) 618–623.

[19] A.R. Freedman and R.E. Ladner, Space bounds for processing contentless inputs, *J. Comput. System Sci.* **11** (1975) 118–128.

[20] R. Freivalds, On time complexity of deterministic and nondeterministic Turing machines, *Latvian Math.* **23** (1979) 158–165.

[21] R. Freivalds, Space and reversal complexity of probabilistic one-way Turing machines, in: *FCT '83*, Lecture Notes in Computer Science, Vol. 158 (Springer, Berlin, 1983) 159–170.

[22] M. Furst, R. Lipton and L. Stockmeyer, Pseudorandom number generation and space complexity, in: *FCT '83*, Lecture Notes in Computer Science, Vol. 158 (Springer, Berlin, 1983) 171–176.

[23] W. Gasarch, Oracles: three new results, Lecture Notes in Pure and Applied Math., Vol. 106 (Dekker, New York, 1987) 219–251.

[24] V. Geffert, Nondeterministic computations in sublogarithmic space and space constructibility, in: *ICALP '90*, Lecture Notes in Computer Science, Vol. 443 (Springer, Berlin, 1990) 111–124.

[25] J. Gill, Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6** (1977) 675–695.

[26] J. Gill, J. Hunt and J. Simon, Deterministic simulation of tape-bounded probabilistic Turing machine transducers, *Theoret. Comput. Sci.* **12** (1980) 333–338.

[27] J. Hartmanis, New developments in structural complexity theory, in: *ICALP '88*, Lecture Notes in Computer Science, Vol. 317 (Springer, Berlin, 1988) 271–286.

[28] J. Hartmanis, New developments in structural complexity theory, *Theoret. Comput. Sci.* **71** (1990) 79–93.

[29] J. Hartmanis and L. Berman, A note on tape bounds for SLA language processing, in: *Proc. 16th IEEE Symp. on Foundations of Computer Science* (1975) 65–70.

[30] J. Hartmanis and L. Berman, On tape bounds for single letter alphabet language processing, *Theoret. Comput. Sci.* **3** (1976) 213–224.

[31] J. Hartmanis, R. Chang, J. Kadin and J. Mitchell, The structural complexity column: some observations about relativization of space-bounded computations, *Bull. EATCS* **35** (1988) 82–92.

[32] J. Hartmanis and S. Mahaney, Languages simultaneously complete for one-way and two-way log-tape automata, *SIAM J. Comput.* **10** (1981) 383–390.

[33] J. Hartmanis and D. Ranjan, Space-bounded computations: review and new separation results, in: *MFCS '89*, Lecture Notes in Computer Science, Vol. 379 (Springer, Berlin, 1989) 49–66.

[34] J.E. Hopcroft and J.D. Ullman, Some results on tape-bounded Turing machines, *J. ACM* **16** (1969) 168–177.

[35] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation* (Addison-Wesley, Reading, MA, 1979).

[36] O.H. Ibarra, A hierarchy theorem for polynomial-space recognition, *SIAM J. Comput.* **3** (1974) 184–187.

[37] O.H. Ibarra and B. Ravikumar, Sublogarithmic-space Turing machines, nonuniform space complexity, and closure properties, *Math. Systems Theory* **21** (1988) 1–17.

[38] N. Immerman, Languages that capture complexity classes, *SIAM J. Comput.* **16** (1987) 760–778.

[39] N. Immerman, Nondeterministic space is closed under complementation, *SIAM J. Comput.* **17** (1988) 935–938.

[40] K. Inoue, A. Ito and I. Takanami, A note on alternating Turing machines using small space, *Trans. IEICE, Section E* **70** (1987) 990–996.

[41] H. Jung, Relationships between probabilistic and deterministic tape complexity, in: *MFCS '81*, Lecture Notes in Computer Science, Vol. 118 (Springer, Berlin, 1981) 339–346.

[42] H. Jung, On probabilistic time and space, in: *ICALP '85*, Lecture Notes in Computer Science, Vol. 194 (Springer, Berlin, 1985) 310–317.

[43] R. Kannan, Alternation and the power of nondeterminism, in: *Proc. 15th Ann. ACM Symp. on Theory of Computing* (1983) 344–346.

[44] R.M. Karp and R.J. Lipton, Some connections between nonuniform and uniform complexity classes, in: *Proc. 12th Ann. ACM Symp. on Theory of Computing* (1980) 302–309.

[45] R.M. Karp and R.J. Lipton, Turing machines that take advice, *Monographie de l'Enseignement Mathematique* **30** (1982) 255–273.

[46] M. Karpinski and R. Verbeek, There is no polynomial deterministic space simulation of probabilistic space with a two-way random-tape generator, *Inform. and Control* **67** (1985) 158–162.

[47] M. Karpinski and R. Verbeek, On the power of two-way random generators and the impossibility of deterministic poly-space simulation, *Inform. and Control* **71** (1986) 131–142.

[48] M. Karpinski and R. Verbeek, Randomness, provability, and the separation of Monte Carlo time and space in: *Computation Theory and Logic*, Lecture Notes in Computer Science, Vol. 270 (Springer, Berlin, 1987) 189–207.

[49] M. Karpinski and R. Verbeek, On the Monte Carlo space constructible functions and separation results for probabilistic complexity classes, *Inform. and Comput.* **75** (1987) 178–189.

[50] B. Kirsig and K.-J. Lange, Separation with the Ruzzo, Simon and Tompa relativization implies DSPACE($\log n$)$\neq$NSPACE($\log n$), *Inform. Process. Lett.* **25** (1987) 13–15.

[51] S.-Y. Kuroda, Classes of languages and linear-bounded automata, *Inform. and Control* **7** (1964) 207–223.

[52] R.E. Ladner, R.J. Lipton and L.J. Stockmeyer, Alternating pushdown and stack automata, *SIAM J. Comput.* **13** (1984) 135–155.

[53] R.E. Ladner and N.A. Lynch, Relativization of questions about log space computability, *Math. Systems Theory* **10** (1976) 19–32.

[54] B.E. Litow, On efficient deterministic simulation of Turing machine computations below logspace, *Math. Systems Theory* **18** (1985) 11–18.

[55] H. Machida and T. Kasai, Space complexity in on-line computation, *J. Comput. System Sci.* **24** (1982) 362–372.

[56] C. Meinel, p-projection reducibility and the complexity classes L(nonuniformity) and NL (nonuniformity), in: *MFCS '86*, Lecture Notes in Computer Science, Vol. 233 (Springer, Berlin, 1986) 527–535.

[57] B. Monien and I.H. Sudborough, On eliminating nondeterminism from Turing machines which use less than logarithmic worktape space, in: *ICALP '79*, Lecture Notes in Computer Science, Vol. 71 (Springer, Berlin, 1979) 431–445.

[58] B. Monien and I.H. Sudborough, On eliminating nondeterminism from Turing machines which use less than logarithmic worktape space, *Theoret. Comput. Sci.* **21** (1982) 237–253.

[59] N. Pippenger, On simultaneous resource bounds, in: *Proc. 20th IEEE Symp. on Foundations of Computer Science* (1979) 307–311.

[60] D.A. Plaisted, New NP-hard and NP-complete polynomial and integer divisibility problems, in: *Proc. 18th Symp. on Foundations of Computer Science* (IEEE, place, 1977) 241–253.

[61] C.W. Rackoff and J.I. Seiferas, Limitations on separating nondeterministic complexity classes, *SIAM J. Comput.* **10** (1981) 742–745.

[62] J.B. Rosser and L. Schoenfeld, Approximate formulas for some functions of prime numbers, *Illinois J. Math.* **6** (1962) 64–94.

[63] W.L. Ruzzo, On uniform circuit complexity, *J. Comput. System Sci.* **22** (1981) 365–383.

[64] W.L. Ruzzo, J. Simon and M. Tompa, Space-bounded hierarchies and probabilistic computations, in: *Proc. 14th Ann. ACM Symp. on Theory of Computing* (1982) 330–335.

[65] W.L. Ruzzo, J. Simon and M. Tompa, Space-bounded hierarchies and probabilistic computations, *J. Comput. System Sci.* **28** (1984) 216–230.

[66] W.J. Savitch, Relationships between nondeterministic and deterministic tape complexities, *J. Comput. System Sci.* **4** (1970) 177–192.

[67] W.J. Savitch, A note on relativized log space, *Math. Systems Theory* **16** (1983) 229–235.

[68] C.P. Schnorr, The network complexity and the Turing machine complexity of finite functions, *Acta Inform.* **7** (1976) 95–107.

[69] J.I. Seiferas, Techniques for separating space complexity classes, *J. Comput. System Sci.* **14** (1977) 73–99.

[70] J.I. Seiferas, Relating refined space complexity classes, *J. Comput. System Sci.* **14** (1977) 100–129.

[71] J. Simon, On the difference between one and many, in: *ICALP '77*, Lecture Notes in Computer Science, Vol. 52 (Springer, Berlin, 1977) 480–491.

[72] J. Simon, Space-bounded probabilistic Turing machine complexity classes, in: *Proc. 13th Ann. ACM Symp. on Theory of Computing* (1981) 158–167.

[73] J. Simon, On tape-bounded probabilistic Turing machine acceptors, *Theoret. Comput. Sci.* **16** (1981) 75–91.

[74] J. Simon, J. Gill and J. Hunt, On tape-bounded probabilistic Turing machine transducers, in: *Proc. 19th IEEE Symp. on Foundations of Computer Science* (1978) 107–112.

[75] M. Sipser, Halting space-bounded computations, in: *Proc. 19th IEEE Symp. on Foundations of Computer Science* (1978) 73–74.

[76] M. Sipser, Halting space-bounded computations, *Theoret. Comput. Sci.* **10** (1980) 335–338.

[77] C. Slot and P. van Emde Boas, On tape versus core: an application of space efficient perfect hash functions to the invariance of space, in: *Proc. 16th Ann. ACM Symp. on Theory of Computing* (1984) 391–400.

[78] C. Slot and P. van Emde Boas, The problem of space invariance for sequential machines, *Inform. and Comput.* **77** (1988) 93–122.

[79] R.E. Stearns, J. Hartmanis and P.M. Lewis, Hierarchies of memory limited computations, in: *Proc. 6th IEEE Conf. of Switching Circuit Theory and Logical Design* (1965) 179–190.

[80] I.H. Sudborough, Efficient algorithms for path system problems and applications to alternating and time-space complexity classes, in: *Proc. 21st IEEE Symp. on Foundations of Computer Science* (1980) 62–73.

[81] R. Szelepcsényi, The method of forced enumeration for nondeterministic automata, *Acta Inform.* **26** (1988) 279–284.

[82] A. Szepietowski, There are no fully space constructible functions between log log n and log n, *Inform. Process. Lett.* **24** (1987) 361–362.

[83] A. Szepietowski, Remarks on languages acceptable in log log n space, *Inform. Process. Lett.* **27** (1988) 201–203.

[84] A. Szepietowski, If deterministic and nondeterministic space complexities are equal for log log n then they are also equal for log n, *STACS '89*, Lecture Notes in Computer Science, Vol. 349 (Springer, Berlin, 1989) 251–255.

[85] A. Szepietowski, Some remarks on the alternating hierarchy and the closure under complement for sublogarithmic space, *Inform. Process. Lett.* **33** (1989) 73–78.

[86] A. Szepietowski, Some notes on strong and weak log log n space complexity, *Inform. Process. Lett.* **33** (1989) 109–112.

[87] A. Szepietowski, If deterministic and nondeterministic space complexities are equal for log log n, then they are also equal for log n, *Theoret. Comput. Sci.* **74** (1990) 115–119.

[88] M. Tompa, An extension of Savitch's theorem to small space bounds, *Inform. Process. Lett.* **12** (1981) 106–108.

[89] P. van Emde Boas, Space measure for storage modification machines, *Inform. Process. Lett.* **30** (1989) 103–110.

[90] K. Wagner and G. Wechsung, *Computational Complexity* (Reidel, Dordrecht, 1986).

[91] D.J.A. Welsh, Randomized algorithms, *Discrete Appl. Math.* **5** (1983) 133–145.

[92] J. Wiedermann, Normalizing and accelerating RAM computations and the problem of reasonable space measures, in: *ICALP '90*, Lecture Notes in Computer Science, Vol. 443 (Springer, Berlin, 1990) 125–138.

[93] C.B. Wilson, A measure of relativized space which is faithful with respect to depth, *J. Comput. System Sci.* **36** (1988) 303–312.