JOURNAL OF COMPUTER AND SYSTEM SCIENCES 32, 374–392 (1986)

Domino-Tiling Games*

BOGDAN S. CHLEBUS

Institute of Informatics, Warsaw University, PKiN, VIIIp., 00-901, Warsaw, Poland

Received April 4, 1984; revised July 23, 1985

Games in which players build domino tilings are considered. The computational complexity of problems of existence of winning strategies is investigated. These problems are shown to be complete in the respective complexity classes, e.g., SQUARE TILING GAME is complete in PSPACE, HIGH TILING GAME is complete in 2EXPTIME and has a doubly exponential time lower bound. As an application, new simple hardness proofs for certain propositional logics are obtained. © 1986 Academic Press, Inc.

1. INTRODUCTION

A square of unit area with coloured edges is called a domino. Given a set of dominoes, the standard "existential" problem is to verify whether it is possible to tile a prescribed portion of the plane with the dominoes, i.e., to cover it tightly with them in such a way that adjacent tiles have matching colours along their common borders.

Domino tilings have proved to be a convenient vehicle to encode computations, especially in order to employ them further in hardness or undecidability proofs in logic. The idea of encoding computations as tilings in the plane can be traced back to Wang [28] who introduced and investigated the *unbounded* domino-tiling problems. The final solution of the problem of classification, with respect to the decidability question, of the classes of first-order formulas determined by prefixes of quantifiers was a spectacular application of dominoes (cf. Kahr, Moore, and Wang [14]). Many variants of the domino-tiling problem have been shown to be undecidable, the most general case was considered by Berger [1]. A modern exposition of the interplay between decision problems of first-order logic and domino-tiling problems can be found in a monograph [18] by Lewis, which also contains an extensive bibliography.

The *bounded* variants of domino-tiling problems, concerning the existence of tilings of bounded portions of the plane, were considered by Levin [16], Lewis [17], and Garey, Johnson, and Papadimitriou (cf. [8]). An example of this kind is SQUARE TILING in which the input is a finite set of domino types and a natural

* The results presented herein are taken from the author's doctoral dissertation [4] written under the supervision of Antoni Kreczmar.

0022-0000/86 $3.00

number n written in the unary notation, and the question is whether an $n \times n$ square can be tiled with the given dominoes. SQUARE TILING is complete in NPTIME. It was shown by Savelsberg and van Emde Boas [22] to be a viable alternative to SATISFIABILITY as a foundation of the NP-completeness theory.

Another family of *recurring* domino problems, characterized by the requirement that a designated domino or colour occur infinitely many times in the tiling, was introduced by Harel [9, 10]. He showed that these problems fall beyond the scope of the arithmetical hierarchy, and used them to exhibit a high degree of undecidability of certain logical calculi.

In this paper we consider strategy problems in domino-tiling games. The games are played by two players who build a bounded domino tiling in the course of a game. One of the players, CONSTRUCTOR, wins after construction of a domino tiling of a prescribed kind has been successfully completed. We show that the problems whether CONSTRUCTOR has a forced win in such games are complete in the respective complexity classes. The classification of the computational complexity of these problems is proved using the theory of alternating Turing machines developed by Chandra, Kozen, and Stockmeyer [2] (for a similar application to combinatorial games see Fraenkel and Lichtenstein [7] and Stockmeyer and Chandra [26]). Three particular game-strategy problems are considered herein: SQUARE TILING GAME, RECTANGLE TILING GAME, and HIGH TILING GAME; they are complete, respectively, in PSPACE, EXPTIME, and 2EXPTIME. The latter ones have exponential and doubly exponential lower bounds on their time complexity, respectively. These completeness results are used in the sequel as a basis of new simple hardness-in-a-class proofs for the propositional logics having means to express the "alternation of quantifiers."

In connection with the existential domino-tiling results mentioned above, many arguments have been presented to the effect that domino problems are a convenient tool to establish intractability or undecidability of logics (cf. [4, 5, 9, 10, 14, 17, 18, 19, 22, 28]). The reason for the usefullness of tiling problems, when dealing with the decision problems of the logical calculi, is that simple configurations of figures in the plane can be directly rendered into equally simple logical formulas. The present paper reinforces the arguments for the usefullness of tiling problems by showing that it is possible to extend this approach by way of domino-tiling games to cover other logics to which the previous methods did not apply directly.

2. Semantics of Alternating Computations

We recall here the basic ideas needed to understand and follow the use of the notion of the alternating Turing machine (TM). Our description of alternation does not follow directly the exposition of Chandra, Kozen, and Stockmeyer [2], but is equivalent to it. We employ the notion of a winning strategy in the respective games explicitly in the definition to render alternation applicable in a straightforward

manner to the classification of complexity of the investigated game-strategy problems (cf. Peterson and Reif [21]).

An alternating TM is a system $M=(k, Q, E, q_0, a, \Sigma, \Gamma, b, \$, \#, \delta)$, where

$k \geqslant 1$ is the number of storage tapes;

Q is the set of states;

$E \subseteq Q$ is the set of *existential* states;

($Q-E$ is the set of *universal* states);

q_0, a are the initial and accepting states, respectively;

Σ is the tape alphabet;

$\Gamma \subseteq \Sigma$ is the input alphabet;

b, \$, $\# \in \Sigma - \Gamma$ are the blank symbol and two markers;

$\delta \subseteq Q \times (\Gamma \times \{\$, \#\}) \times \Sigma^k \times Q \times \Sigma^k \times \{L, R\}^{k+1}$ is the next-move relation.

The "physical representation" of M has a read-only input tape, k read/write storage tapes, and a finite control. An *instantaneous description* (ID) is a sequence of the form $(q, \$x\#, y_1,..., y_k, i_o, i_1,..., i_k)$, where q is a state, $\$x\#$ is the input x from Γ^* written on the input tape between the markers, y_1 to y_k are the contents of the storage tapes, i_0 is the number of the scanned cell on the input tape, and i_1 to i_k are the respective numbers concerning storage tapes. If q is universal, then this ID is called universal, otherwise it is existential.

Let $x \in \Gamma^*$ be an input for M. Consider the following two-person game G_x, with the players denoted as A (from "accept") and R (from "reject"). The players perform moves in such a way that there is obtained a sequence of IDs forming a *computation*, i.e., a string of consecutive IDs in the sense of the next-move relation. The game starts from the initial ID. Each single move of A or R determines the next ID. If the processed ID is existential, it is A's turn to perform a move, otherwise it is its adversary's. Suppose that $\tau=(q, \$x\#, y_1,..., y_k, i_0, i_1,..., i_k)$ is such an ID. Let u_0 be the i_0th element of the string $\$x\#$, and, for $1 \leqslant n \leqslant k$, let u_n be the i_nth element of y_n. Consider the set $S_\tau = \{(q', u_1',..., u_k', m_0, m_1,..., m_k)$: $(q, u_0, u_1,..., u_k, q', u_1',..., u_k', m_0,..., m_k) \in \delta\}$. The player whose turn is to move selects an element, say, $(q', u_1',..., u_k', m_0,..., m_k)$ from S_τ. Then the next ID τ' is obtained from τ by changing: q to q', u_n to u_n' in y_n, and either i_n to i_n+1 if $m_n=R$ or to i_n-1 if $m_n=L$, for $1 \leqslant n \leqslant k$. The game proceeds until either an accepting ID or an ID τ with the empty set S_τ appears. In the former case it is player A who wins, in the latter one R wins. The game does not need to terminate, its course may be interminable with none of the players winning. A strategy of a player is a function which gives as a value $f(\tau)$ an element from the set S_τ, for those τ's for which it is this very player's turn to perform a move. A *winning strategy* is such a strategy which forces win under all the strategies of the adversary. An alternating TM M is said to *accept* $x \in \Gamma^*$ iff player A has a winning strategy in the game G_x. Let s_x be such a winning strategy for A in G_x. The *time* of s_x is defined to be the maximum of the numbers of IDs that can appear in the course of G_x when strategy s_x is employed by A. Similarly, the space

of s_x is the maximum of the lengths of words that can appear on a storage tape during playing G_x when A applies s_x. Let T and S be functions from N (the set of natural numbers) into N. Let L be the set of words accepted by an alternating TM M. M is said to *accept L in time $T(n)$* (*in space $S(n)$*, respectively) if, for each $x \in L$, there is such a winning strategy s_x for A in the game G_x that the time of s_x (the space of s_x, respectively) is not greater than $T(|x|)$ ($S(|x|)$, respectively), where $|x|$ is the length of x. It follows from the above definitions that alternating TMs are a generalization of nondeterministic TMs: the nondeterministic TMs are the alternating TMs with only existential states.

3. Computational Complexity

We shall recall here certain facts concerning complexity classes (cf. Hopcroft and Ullman [12]). Let $T: N \to N$ be a function. Sets DTIME($T(n)$), NTIME($T(n)$), and ATIME($T(n)$) denote the classes of languages accepted in time $T(n)$ by deterministic, nondeterministic, and alternating TMs, respectively. The classes DSPACE($S(n)$), NSPACE($S(n)$), and ASPACE($S(n)$), for a function $S: N \to N$, are defined analogously, the storage space being measured now. Let symbol X stand for one of the letters D, N, or A. The following classes of languages will appear in further considerations:

$$\text{XPTIME} = \bigcup_{k \in N} \text{XTIME}(n^k),$$

$$\text{XEXPTIME} = \bigcup_{k \in N} \text{XTIME}(2^{n^k}),$$

$$\text{X2EXPTIME} = \bigcup_{k \in N} \text{XTIME}(2^{2^{n^k}}).$$

The classes XPSPACE, XEXPSPACE, and X2EXPSPACE are defined analogously. Letter D is usually omitted: for example, DPTIME is denoted as PTIME.

3.1. Fact (Savitch [23]). *If $S: N \to N$ is a fully constructible function and $S(n) \geqslant \log n$, then* $\text{NSPACE}(S(n)) \subseteq \text{DSPACE}(S(n)^2)$.

As a corollary, the following identities are obtained: NPSPACE = PSPACE, NEXPSPACE = EXPSPACE.

3.2. Fact (Chandra, Kozen, and Stockmeyer [2]):

1. *If $S(n) \geqslant n$ then* $\text{NSPACE}(S(n)) \subseteq \bigcup_{c>0} \text{ATIME}(cS(n)^2)$;
2. *If $T(n) \geqslant n$ then* $\text{ATIME}(T(n)) \subseteq \text{DSPACE}(T(n))$;
3. *If $S(n) \geqslant \log n$ then* $\text{ASPACE}(S(n)) \subseteq \bigcup_{c>0} \text{DTIME}(c^{S(n)})$;
4. *If $T(n) \geqslant n$ then* $\text{DTIME}(T(n)) \subseteq \text{ASPACE}(\log(T(n)))$.

The above results yield the following characterization of the alternating space: if $S(n) \geqslant \log n$ then $\mathrm{ASPACE}(S(n)) = \bigcup_{c>0} \mathrm{DTIME}(c^{S(n)})$. As a corollary, the following equalities between the alternating and deterministic complexity classes are obtained: ALOGSPACE = PTIME, APTIME = PSPACE, APSPACE = EXPTIME, AEXPTIME = EXPSPACE, AEXPSPACE = 2EXPTIME.

Let us consider the deterministic TMs with the added write-only output tape. Assume further that the head on the output tape cannot move to the left. Let M be such a machine, and let $f\colon \Gamma^* \to \Sigma^*$ be a function. Function f is said to be *computed* by M in time $T(n)$ (space $S(n)$), respectively) if, for each input x, M halts eventually with $f(x)$ written on the output tape after at most $T(|x|)$ moves (with at most $S(|x|)$ cells used on each storage tape, respectively) in the course of the computation.

Let $L_0 \subseteq \Gamma^*$, $L_1 \subseteq \Sigma^*$ be languages. L_0 is *log-space reducible* to L_1, $L_0 \leqslant_{\log} L_1$, if there is a function $f\colon \Gamma^* \to \Sigma^*$ such that $x \in L_0$ iff $f(x) \in L_1$, and f is computed in logarithmic space (cf. Jones [13]). We also say that L_0 is reducible to L_1 via f. Let C be a class of languages. A set L of words is *hard* for C iff $K \leqslant_{\log} L$, for each $K \in C$. L is *complete* in C iff $L \in C$ and L is hard for C. Let $f\colon \Gamma^* \to \Sigma^*$ and $g\colon N \to N$ be functions. Function f is said to be $g(n)$-*length bounded* iff, for each $x \in \Gamma^*$, the inequality $|f(x)| \leqslant g(|x|)$ holds. A language L_0 is *reducible to* L_1 *via length order* $g(n)$ iff L_0 is log-space reducible to L_1 via some f such that, for some constant $c > 0$, f is $cg(n)$-length bounded. This definition extends accordingly to classes of languages: if C is a class of languages then C is said to be reducible to a language L via length order $g(n)$ iff each $K \in C$ is reducible to language L via length order $g(n)$.

3.3. FACT (Jones [13], Meyer and Stockmeyer [20]). *Let T and S be monotone non-decreasing functions from N into N. Let $A \leqslant_{\log} B$ via f which is $g(n)$-length bounded. Let X stand for one of the letters D, N, or A. Then the following implications hold:*

1. $B \in \mathrm{XSPACE}(S(n))$ *implies* $A \in \mathrm{XSPACE}(S(g(n)) + \log n)$;
2. $B \in \mathrm{XTIME}(T(n))$ *implies* $A \in \mathrm{XTIME}(T(g(n)) + p(n))$, *for a polynomial* $p(n)$.

In the proposition below we use the following notation: let X be one of the letters D, N, or A; the expressions of the form $\mathrm{XTIME}(2^{\mathrm{lin}\, n})$ or $\mathrm{XSPACE}(2^{\mathrm{lin}\, n})$ denote $\bigcup_{c>0} \mathrm{XTIME}(2^{cn})$ or $\bigcup_{c>0} \mathrm{XSPACE}(2^{cn})$, respectively.

3.4. PROPOSITION. 1. *If* $\mathrm{DTIME}(2^{\mathrm{lin}\, n})$ *is reducible to a language L via length order $n \log n$ then there is a constant $c > 0$ such that* $L \notin \mathrm{DTIME}(2^{cn/\log n})$.

2. *If* $\mathrm{NSPACE}(2^{\mathrm{lin}\, n})$ *is reducible to a language L via length order $n \log n$ then there is a constant $c > 0$ such that* $L \notin \mathrm{NSPACE}(2^{cn/\log n})$.

3. *If* $\mathrm{ASPACE}(2^{\mathrm{lin}\, n})$ *is reducible to a language L via length order $n \log n$ then there is a constant $c > 0$ such that* $L \notin \mathrm{DTIME}(2^{2^{cn/\log n}})$.

Proof. 1. Suppose that $\mathrm{DTIME}(2^{\mathrm{lin}\, n})$ is reducible to L via some f such that $|f(x)| \leqslant d|x| \log(|x|)$, for a certain $d > 1$. From the deterministic hierarchy theorem of Hennie and Stearns [11] it follows that there is a language $B \in \mathrm{DTIME}(3^n) - \mathrm{DTIME}(2^n/n)$. Assume, for the time being, that $L \in \mathrm{DTIME}(2^{en/\log n})$, for some $e > 0$. By Fact 3.3 we can infer that $B \in \mathrm{DTIME}(2^{den \log n/\log(dn \log n)} + p(n))$, for a certain polynomial $p(n)$. Since $\log n/\log(dn \log n) \leqslant 1$, the following inequality $2^{den \log n/\log(dn \log n)} + p(n) \leqslant 2^{n - \log n}$ is true provided $de < 1$ and n is sufficiently large. Therefore, to avoid contradiction, we have $L \notin \mathrm{DTIME}(2^{cn/\log n})$ for each $c < d^{-1}$, by the choice of B.

2. The proof is analogous; it employs the respective hierarchy theorem for nondeterministic space—see Seiferas [24].

3. A hierarchy theorem for the alternating space, analogous to those used above, can be readily proved—the argumentation follows closely the proof of the respective theorem concerning deterministic space. From this we obtain similarly $L \notin \mathrm{ASPACE}(2^{dn/\log n})$, for a certain $d > 0$. Therefore, by Fact 3.2, L does not belong to $\mathrm{DTIME}(2^{2^{c/\log n}})$, for a certain $c > 0$. ∎

4. Bounded Tilings

A domino is a unit area "tile" with coloured edges. An ordered quadruple of colours is said to be a *domino type*. A unit square whose top, right, bottom, and left sides are coloured as specified by the type is called a *domino of* (the given) *type*. We assume that dominoes are placed on the grid in the plane, and that they are not turned in any way. Let T be a finite set of domino types, and let us distinguish one of the colours, say, white. Consider a number of dominoes of types from T placed in the plane in such a way that they cover a rectangle with k rows and n columns, the adjacent edges have matching colours, and the external edges are coloured white. Such a set of dominoes is said to be a *T-tiled $k \times n$ rectangle*.

There are two "existential" decision problems concerning bounded tilings which are relevant to the further considerations. The first, called SQUARE TILING, has its instances determined by a finite set T of domino types and a natural number n written in the unary notation, and the question is whether there is a T-tiled $n \times n$ square. The second one, termed RECTANGLE TILING, is concerned again with set T and number n, but the question is whether there is a T-tiled $k \times n$ rectangle for some k. These problems are complete in NPTIME and PSPACE, respectively. The proofs of these facts are based on the coding abilities of tilings. Since we shall need in the sequel a refined argumentation of this kind, we describe a possible method of proof (cf., for example, [22]).

Given a computation of a nondeterministic single-tape TM, i.e., a sequence of IDs, we convert it into a sequence of rows of a tiled rectangle, where two consecutive rows encode a single ID. The tiles may be viewed as "records" containing

information which they can transmit to adjacent ones via coloured edges. They fall into the following three categories:

1.The "external" dominoes: they have one of the horizontal edges coloured white; in the bottom row they encode the input, and in the topmost one are compatible only with the "accepting" row below.

2. The "tape symbol" dominoes: they encode single tape symbols which are not being scanned by the head; they are of the form (a) depicted in Fig. 1, where x is a tape symbol of M and the empty quadrant denotes the white colour.

3. The "state" dominoes: they encode the actions of the finite control and the scanned symbol, and are of the six forms from (b) to (g) in Fig. 1. They are defined as follows: For each $(q, x, q', x', L) \in \delta$ and $(q, x, q', x', R) \in \delta$ there are the domino types (b) and (c), respectively. We may assume without loss of generality that the set of states is divided into two separate parts of the *left-moving* states, which can be entered only while the head is moving to the left, and the *right-moving* states, which can be entered while moving to the right. For each left-moving state q and tape symbol x there are the domino types of the form (d) and (e), and for each right-moving state q and tape symbol x there are the domino types (f) and (g). It follows from our assumptions that dominoes of types (e) and (g) cannot be adjacent. This prevents "phantom heads" to appear.

It should be clear from the form of dominoes and the above description how a tiled rectangle encodes an accepting computation.

Now the respective proofs of completeness in a class can be carried out as follows. The fact that SQUARE TILING and RECTANGLE TILING belong to NPTIME and PSPACE, respectively, is clear. Consider a language $L \in$ NPTIME. There is a nondeterministic single-tape TM M recognizing L in time n^k for a certain $k \in N$. Given a word x in the alphabet of M, construct a set of domino types T_x in a way described above (the external dominoes in the bottom row should encode x), and such that $x \in L$ iff there is a T_x-tilled square with $2|x|^k$ rows and columns. This provides the required reduction. A similar argumentation prevails in the second case.

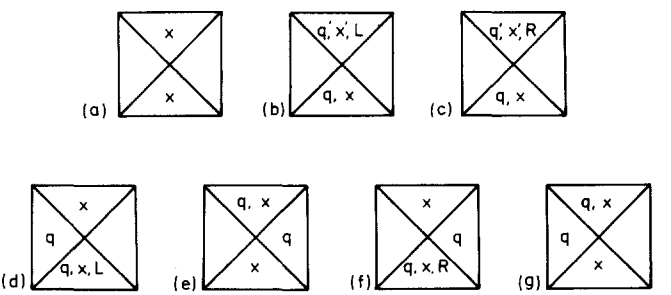

FIG. 1. The domino types for the existential tiling problem.

5. Games

Consider the following tiling games. There are two players: CONSTRUCTOR and SABOTEUR, and there is given a finite set T of domino types and a natural number n written in unary. The players are building a T-tiled rectangle with n columns in the course of a game. The ultimate aim of CONSTRUCTOR is to build a T-tiled rectangle of the precribed kind, while SABOTEUR wants to hinder him. The players alternately select and put a domino fitting in the consecutive vacant place. More precisely, the first move is performed by CONSTRUCTOR who selects a domino for the first (i.e., leftmost) column and first (i.e., bottom) row. Then SABOTEUR chooses a tile for the second column and first row, and so on, until the first row consists of n tiles. Then the second row is being filled, from left to right, upon its completion the next one, and so on. A domino *fits* in the vacant place if it is compatible with the adjacent squares to the left and to below (or only to the one adjacent square in the case of the first row or column). If there is no domino type fitting in the consecutive vacant place, or if a domino with a non-white outer edge is placed by a player, and thus immediately violating the definition of a T-tiling, then SABOTEUR automatically wins. It depends on the kind of game to say when CONSTRUCTOR wins. In this section we consider two cases. In the SQUARE TILING GAME problem the game terminates after a square with n rows has been built (if not before) and CONSTRUCTOR wins provided a T-tiled square has been constructed; in the other case his adversary wins. In RECTANGLE TILING GAME problem the game may proceed interminably. It terminates only if SABOTEUR wins or if a T-tiled rectangle has been built; in the latter case CONSTRUCTOR wins. In both problems the question is whether CONSTRUCTOR has a winning strategy in the respective game determined by the given T and n.

5.1. Theorem. 1. *SQUARE TILING GAME is complete in PSPACE.*

2. *RECTANGLE TILING GAME is complete in EXPTIME.*

Proof. To show that these problems belong to the respective classes, take the following straightforward algorithm: simulate the game in such a way that player A acts as CONSTRUCTOR and player R as SABOTEUR. These algorithms require polynomial time and space, respectively.

In the hardness part we again use the properties of alternation expressed in Fact 3.2. The reasoning is along the lines of the proofs of the existential counterparts of these problems considered in the previous section. Take a language L from PSPACE (or EXPTIME) recognized by an alternating TM M in polynomial time (space, respectively). Consider the sets T_x of domino types defined as before. We would like to find a method of updating them in such a way that CONSTRUCTOR would have a forced win in the respective game determined by T_x iff x belonged to L. The T_x-tillings, for the original T_xs, correspond to computations accepting x. Therefore CONSTRUCTOR loses unless he keeps to the general con-

straints on the consistency of the tiling, while SABOTEUR has the advantage of being able to win by "cheating." Therefore CONSTRUCTOR should have means to control his adversary to some extent to make the game square.

First let us make the game more regular. Observe that the following restrictions may be imposed without losing generality:

1. The initial state of M is existential.

2. The existential and universal states of M alternate during computations of M.

3. The number n is even.

With these stipulations CONSTRUCTOR and SABOTEUR would select dominoes for the odd and even numbered columns, respectively. Assuming for the time being a proper behaviour of SABOTEUR, the players would choose the tiles of types (b) and (c) alternatively in such a way that CONSTRUCTOR would select the existential and SABOTEUR the universal dominoes. Types (b) and (c) determine the moves of M, and the need of them was the reason to adopt the method of encoding computations of M with two rows per one ID.

Let us determine the particular ways of violating the contraints on the consistency of the tiling which SABOTEUR might utilize. They are the following:

1. He might put a domino of type (e) in a place where a plain tape tile of type (a) is needed.

2. He might put a tape domino of type (a) in a place where a state domino of type (e) is needed. Such possibility occurs after CONSTRUCTOR has placed in the row below a square of type (d) enforcing a move to the left.

Therefore it would be sufficient if CONSTRUCTOR could enforce the exact form of the tile adjacent to the right. To attain this goal we modify the domino types in the following way. The external dominoes remain the same. Each of the other domino types is substituted by its two variants: one with letter S added on its right quadrant, and the second with letter S added on the left quadrant. The former will be used by CONSTRUCTOR and the latter by SABOTEUR; in their final form they are depicted on Figs. 2 and 3, respectively. Let $\bar{q}$ denote a fresh copy of state q,

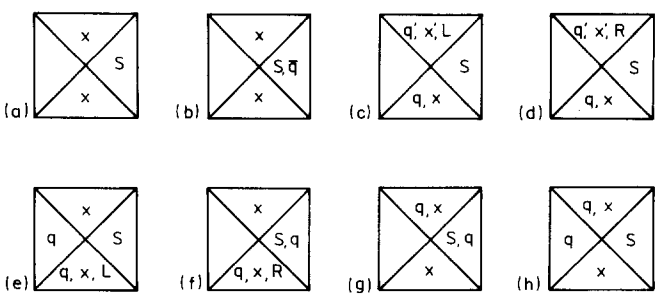

FIG. 2. The domino types used by CONSTRUCTOR.

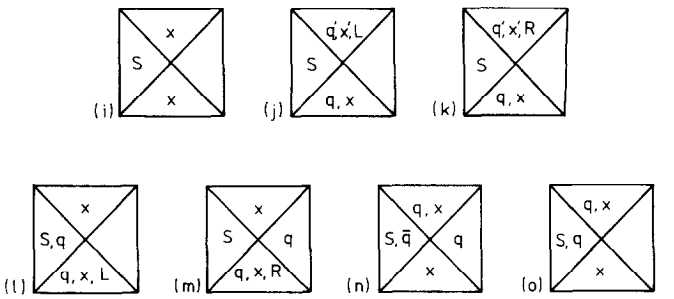

FIG. 3. The domino types used by SABOTEUR.

for each original state q of the TM. CONSTRUCTOR has additionally at his disposal domino tiles of type (b), for each tape symbol x and state q. Finally, the domino types (n) of SABOTEUR, corresponding to (e) in Fig. 1, have symbols $\bar{q}$ added on the left side, where q is the state from the right side of the tile. Observe that there is no confusion between types (n) and (o).

Let T'_x denote the set of the new domino types corresponding to a word x. The mechanizm of encoding, via types from T'_x, has remained essentially the same as in T_x, and CONSTRUCTOR again does not have a tricky way of winning except for trying to build a tiling corresponding to an accepting computation. An inspection of the new types shows that the only places where SABOTEUR has a possibility of choice between distinct dominoes are those where the types (j) and (k) fit in the vacant place, i.e., when SABOTEUR decides the next transition from a universal state. The previous possibilities of cheating have disappeared because now CONSTRUCTOR can use either type (a) enforcing SABOTEUR to put type (i), or can select type (b) enforcing the adversary to place type (n), which will be followed by (e). From this it follows that there is a correspondence betwen the acceptance of M and the existence of a winning strategy for CONSTRUCTOR. The remaining details are the same as in the proofs of the existential counterparts of the considered problems. ∎

From Proposition 3.4 we can infer the following corollary:

5.2. COROLLARY. *There is a constant $c>0$ such that RECTANGLE TILING GAME $\notin DTIME(2^{cn/\log n})$.*

Proof. It follows from the above proof that DTIME($2^{\text{lin}\,n}$) is reducible to RECTANGLE TILING GAME via length order $n \log n$. ∎

6. HIGH TILINGS

In this section we consider more involved tilings. Their decision problems turn out to be complete in higher complexity classes.

Let T be a finite set of domino types, and let $A \subseteq T \times T$ be a binary relation. If $A(x, y)$ holds, for two domino types x and y, they are said to be *A-adjacent*. Let R be a T-tiled rectangle. Two rows in R are *equivalent* if they have dominoes of the same type in each even column. Two dominoes from the first column of R are *distantly adjacent* if their rows are equivalent, and there is no third equivalent row between them. R is said to be a *high* (T, A)*-tiling* if each pair of distantly adjacent dominoes in R is A-adjacent.

Problem HIGH TILING is the following: given a finite set T of domino types, a relation $A \subseteq T \times T$, and a natural number n written in unary, to decide whether there exists a high (T, A)-tiling with n columns. The games considered before have also their high-tiling counterpart. Consider sets T and A and number n as above, and the tiling game determined by T and n. The high-tilling game is defined by the stipulation that CONSTRUCTOR wins if a high (T, A)-tiling has been built in the course of a game. The problem HIGH TILING GAME is the following: given a finite set T of domino types, $A \subseteq T \times T$, and n written in unary, decide whether CONSTRUCTOR has a forced win in the high-tiling game determined by T, A, and n.

6.1. THEOREM. 1. *HIGH TILING is complete in EXPSPACE.*

2. *HIGH TILING GAME is complete in 2EXPTIME.*

Proof. First we describe a nondeterministic algorithm recognizing HIGH TILING. There are given T, A, and n. The algorithm guesses dominoes in consecutive rows storing temporalily the two top rows. After a new row has been determined, the algorithm verifies if the distant adjacency constraints are met. To this end a set of non-equivalent rows is stored additionally, each of them being the last guessed row among the ones equivalent to it and guessed before. This collection is updated after appearance of a new row. The input is accepted after a high (T, A)-tilling has been constructed. The algorithm in this form may loop but it can be converted into an algorithm with the halting property by a standard procedure tantamount to adding a counter (cf. [12]). To evaluate the space requirements, suppose that the input has length k. There are at most $k^k = 2^{k \log k}$ non-equivalent rows, and each row requires $O(k \log k)$ storage space. Therefore $c2^{k \log^2 k \log \log k}$ is an upper space bound, for some c. From the linear speed-up theorem (cf. [12]) we obtain HIGH TILING $\in$ NSPACE$(2^{k \log^2 k \log \log k})$. HIGH TILING GAME can be handled similarly: the above algorithm is transformed into an alternating one in such a way that player A simulates CONSTRUCTOR and player R simulates SABOTEUR; we obtain HIGH TILING GAME $\in$ ASPACE$(2^{k \log^2 k \log \log k}) = \bigcup_{c>0}$ DTIME$(c^{2^{k \log^2 k \log \log k}})$.

Suppose that L is a language accepted in spce 2^{n^k}, for a certain constant $k \in N$, by a single-tape nondeterministic TM M. We shall show how to obtain in log-space a set H_x of domino types and a relation $A_x \subseteq H_x \times H_x$, for a word x in the input alphabet of M, with the property that $x \in L$ iff there is a high (H_x, A_x)-tilling with $2|x|^k$ columns. The idea is the following: the first column in the tilling is the *con-*

tents part (it contains an ordinary tiling with a modified pattern of tiles), the rest is the *address* part (each tile in the first column has its address determined by the remaining part of the row). The address part is controlled by a tile collection which cycles around according to the pattern of a cyclic binary counter. Suppose there is given an ordinary tiling. It can be converted into a high one in such a way that in the contents column the horizontal adjacencies of the original tiling are enforced by vertical relationships, whereas the vertical adjacencies are enforced by the distant adjacencies: whenever the counter assumes the same value mod 2^{n^k} then the two squares of the original tiling are located above each other.

Given a word x, let T_x be the set of domino types defined as in Section 4. Transform T_x into T'_x in such a way that in each tile the top edges are coloured white and the bottom ones encode via one colour the pair of the original top and bottom colours. The sets H_x and $A_x \subseteq H_x \times H_x$ are defined as follows:

1. The contents column: the tiles are these of T'_x rotated by the angle $\pi/2$; more precisely, the vertical adjacency relation in H_x is the horizontal one of T'_x, relation A_x is the vertical adjacency relation of the original T_x, the left edges are white, and the right edges are coloured as the bottom ones of the tiles in T'_x.

2. The address column: a counter in $2|x|^k - 1$ columns is programmed in such a way that in the first column there appear two tiles in alternating rows, in the third one there appear two tiles in alternating blocks of length two, in the fifth one, two tiles in alternating blocks of length four, and so on. The squares in the remaining columns play the role of transmitters between the neighbouring columns. The left external edges match the colours of the contents column. The technical details of this construction are omitted (cf. [4]).

Additionally H_x contains dominoes which appear in single rows separating full cycles of the counter. With this construction, an ordinary tiling of 2^{n^k} columns is converted into a high one with $2n^k$ columns. This completes the first part of the proof.

We sketch the proof of the remaining part since the method of updating H_xs and A_xs is essentially the same as that applied in Section 5.

M is now an alternating TM. Add a new "dummy" column to the address part so that the total number of columns will be $2|x|^k + 1$, and hence an odd one, and add another row to separate the cycles of the counter. Now the players will place tiles in the first column in alternate rows, simulating the way they do it in the ordinary tilings. But still the moves of SABOTEUR in the first column are enforced by the vertical adjacencies only, and he may violate the distant adjacencies. To rule out this possibility do the following: convert the domino types for the contents column in such a way that the same general mechanism of encoding computations is preserved but the types can be separated into two disjoint classes to be used either by CONSTRUCTOR or SABOTEUR; then modify the top colours on the CONSTRUCTOR's tiles, and the bottom ones on the SABOTEUR's accordingly, in such a way that they enforce both vertical and distant adjacencies simultaneously via vertical relationships. Interpreting the original dominoes as records containing

information, each of them is changed into a set of variants, the distinction being the allowance of different sets of tiles immediately above (in the CONSTRUCTOR's case) or below (in the other case). Relation A_x is modified then in such a way that $A_x(y', z')$ holds for variants, say y' of y and z' of z iff it holds for the original domino types y and z. In the address part the moves of both of the players are uniquely determined. With this updating, CONSTRUCTOR will abide by the rule to build a consistent high tiling of this own will, and will be able to enforce SABOTEUR to do the same. Therefore there is a correspondence between the acceptance of M and the existence of a winning strategy for CONSTRUCTOR. This completes the proof. ▮

From this theorem and Proposition 3.4 we can infer the following fact:

6.2. COROLLARY. 1. *There is a constant* $c>0$ *such that HIGH TILING* $\notin$ *NSPACE*$(2^{cn/\log n})$.

2. *There is a constant* $c>0$ *such that HIGH TILING GAME* $\notin$ *DTIME*$(2^{2cn/\log n})$.

Proof. It was actually shown above that both NSPACE$(2^{\text{lin } n})$ is reducible to HIGH TILING and ASPACE$(2^{\text{lin } n})$ is reducible to HIGH TILING GAME via length order $n \log n$. ▮

7. PROPOSITIONAL LOGICS

In this section we show how the results described above can be employed to produce conceptually simple and concise proofs of the hardness in a class of languages of a certain satisfiability problems of propositional logics. All of these complexity results are known, but in this way we contribute to a program initiated in [17, 18, 10, 22]. The logics we consider are those having means to express the "alternation of quantifiers," such as the quantifiers themselves or the modality operators.

There is given a finite set T of domino types and a number n. The following notations will be useful later on. Denote the dominoes in T with top, right, bottom, or left edges coloured white by *TO*, *RI*, *BO*, or *LE*, respectively. For $d \in T$, let $t(d)$ and $r(d)$ denote the subsets of T of elements with the bottom or left edges coloured the same colour as the top or right one, respectively, of d. The following propositional variables are used in the sequel:

1. The variables of the form $v(d, c)$, for each $d \in T$ and $1 \leqslant c \leqslant n$, meaning that there is a domino of type d in the cth column.

2. The variables of the form $v(d, c, r)$, for each $d \in T$ and $1 \leqslant c, r \leqslant n$, meaning that there is a domino of type d in the cth column and the rth row.

The expressions of the form $\bigwedge_{\varphi \in X} \varphi$ and $\bigvee_{\varphi \in X} \varphi$ denote the conjunction and disjunction of elements of X, and are defined to be true or false, respectively, if X is

empty. Given a logic L, notation SAT(L) denotes the set of the satisfiable formulas of L. In the theorems below, proofs are restricted to the hardness parts only.

Quantified Boolean Formulas (QBF)

In this logic the formulas are built up from the propositional variables using the classical propositional connectives and the universal and existential quantifiers ($\forall p$) and ($\exists p$).

7.1. PROPOSITION (Stockmeyer and Meyer [25, 27]). *SAT(QBF) is complete in PSPACE.*

Proof. We show tat SQUARE TILING GAME is reducible to SAT(QBF). Let T be a finite set of domino types and let n be an even natural number. We need the following subformulas:

(1) TIL(c, r): to guarantee that the selected tiles match the adjacent ones; it is defined depending on the values of c and r:

(a) if $c = r = 1$ then it denotes

$$\bigvee_{d \in T} v(d, 1, 1);$$

(b) if $c = 1$ and $r > 1$ then it denotes

$$\bigwedge_{d_1 \in T} (v(d_1, 1, r-1) \to \bigvee_{d \in t(d_1)} v(d, 1, r);$$

(c) in the case $c > 1$ and $r = 1$ it is defined similarly;

(d) if both c and r are greater than 1 then it denotes

$$\bigwedge_{d_1, d_2 \in T} (v(d_1, c-1, r) \wedge v(d_2, c, r-1) \to \bigvee_{d \in r(d_1) \cap t(d_2)} v(d, c, r).$$

(2) UNI(c, r): to guarantee that exactly one tile is selected for each place:

$$\bigwedge_{\substack{d_1, d_2 \in T \\ d_1 \neq d_2}} \neg v(d_1, c, r) \vee \neg v(d_2, c, r).$$

(3) SEL(c, r): it denotes TIL(c, r) $\wedge$ UNI(c, r).

(4) WHI: to guarantee that the external edges are coloured white:

$$\bigwedge_{1 \leqslant i \leqslant n} \left(\bigvee_{d \in TO} v(d, i, n) \wedge \bigvee_{d \in RI} v(d, n, i) \wedge \bigvee_{d \in LE} v(d, 1, i) \wedge \bigvee_{d \in BO} v(d, i, 1) \right).$$

(5) $\exists_{c,r}$: it is an abbreviation for the sequence of quantifiers

$$\exists v(d_1, c, r)\ \exists v(d_2, c, r) \cdots \exists v(d_k, c, r), \text{ where } T = \{d_1, \ldots, d_k\}.$$

(6) $\forall_{c,r}$: it is a similar sequence of $\forall s$.

Consider the following final formula:

$$\exists_{1,1}\mathrm{SEL}(1,1) \wedge \exists_{2,1}\mathrm{SEL}(2,1) \wedge \forall_{2,1}(\mathrm{SEL}(2,1)$$
$$\rightarrow \exists_{3,1}\mathrm{SEL}(3,1) \wedge \exists_{4,1}\mathrm{SEL}(4,1) \wedge \forall_{4,1}(\mathrm{SEL}(4,1) \rightarrow \cdots$$
$$\rightarrow \exists_{1,2}\mathrm{SEL}(1,2) \wedge \exists_{2,2}\mathrm{SEL}(2,2) \wedge \forall_{2,2}(\mathrm{SEL}(2,2) \rightarrow \cdots$$
$$\rightarrow \exists_{1,n}\mathrm{SEL}(1,n) \wedge \exists_{2,n}\mathrm{SEL}(2,n) \wedge \forall_{2,n}(\mathrm{SEL}(2,n)$$
$$\rightarrow \exists_{3,n}\mathrm{SEL}(3,n) \wedge \cdots \wedge \exists_{n,n}\mathrm{SEL}(n,n) \wedge \forall_{n,n}(\mathrm{SEL}(n,n) \rightarrow \mathrm{WHI})\cdots))))).$$

It is clear from its form that this formula is satisfiable iff CONSTRUCTOR has a forced win in the square-tiling game determined by T and n. ∎

Propositional dynamic logic (PDL)

This logic was introduced and investigated by Fischer and Ladner [6]. The characteristic feature of PDL is the ability, for an expression P being a program, to express a nondeterministic and a (dual) universal execution of P, denoted as $\langle P\rangle$ and $[P]$, respectively; and to express iterations of P of arbitrary length, denoted as $\langle P*\rangle$ and $[P*]$, respectively. Placing a formula after such an expression means that it is true after the execution of the program in the indicated manner.

7.2. PROPOSITION (Fischer and Ladner [6]). *SAT(PDL) is complete in EXPTIME.*

Proof. We show that RECTANGLE TILING GAME is reducible to SAT(PDL). Let T and n determine an instance of RECTANGLE TILING GAME; assume that n is even, the rectangle-tiling game determined by T and n always terminates, and that the bottom row is uniquely determined. Let P be a program variable, and let the propositional variables be determined by T and n as was described before. We need the following subformulas:

(1) CON: to guarantee that it is possible to select properly the next tile if the game has not terminated yet:

$$[P*] \bigwedge_{1 \leqslant i \leqslant n} \mathrm{CON}(i),$$

where the formulas CON(i) are defined as follows:

(a) CON(1) denotes

$$\bigwedge_{d_1 \in T}\left((\langle P\rangle \mathrm{TRUE}) \wedge v(d_1,1) \rightarrow \bigvee_{d \in t(d_1)} \langle P\rangle v(d,1)\right)$$

where TRUE is a formula which is always true;

(b) if $i > 1$ then CON(i) denotes

$$\bigwedge_{d_1, d_2 \in T} (\langle P \rangle v(d_1, i-1)) \wedge v(d_2, i) \rightarrow \bigvee_{d \in r(d_1) \cap t(d_2)} \langle P \rangle (v(d_1, i-1) \wedge v(d, i)).$$

(2) SAB: to guarantee that all the possible moves of SABOTEUR are taken into account:

$$[P*] \bigwedge_{\substack{1 < i \leqslant n \\ 2 | i}} \mathrm{SAB}(i)$$

where the formulas SAB(i) are defined as follows:

$$\bigwedge_{d_1, d_2 \in T} (\langle P \rangle v(d_1, i-1)) \wedge v(d_2, i) \rightarrow \bigwedge_{d \in r(d_1) \cap t(d_2)} \langle P \rangle (v(d_1, i-1) \wedge v(d, i)).$$

(3) BOT: to guarantee that the tiles in the first row have matching colours along their common edges:

$$\bigvee_{d \in T} v(d, 1) \wedge \bigwedge_{1 < i \leqslant n} \bigwedge_{d_1 \in T} \left(v(d_1, i-1) \rightarrow \bigvee_{d \in r(d_1)} v(d, i) \right).$$

(4) WHI: to guarantee that the external edges are coloured white:

$$\bigwedge_{1 \leqslant i \leqslant n} \bigvee_{d \in BO} v(d, i) \wedge [P^*] \left(\bigvee_{d \in LE} v(d, 1) \wedge \bigvee_{d \in RI} v(d, n) \right)$$
$$\wedge [P*] \langle P* \rangle \bigwedge_{1 \leqslant i \leqslant n} \bigvee_{d \in TO} v(d, i).$$

(5) UNI: to guarantee that exactly one tile occupies one place:

$$[P*] \bigwedge_{1 \leqslant i \leqslant n} \bigwedge_{\substack{d_1, d_2 \in T \\ d_1 \neq d_2}} \neg v(d_1, i) \wedge \neg v(d_2, i).$$

The final formula is defined as follows:

$$\mathrm{CON} \wedge \mathrm{SAB} \wedge \mathrm{BOT} \wedge \mathrm{WHI} \wedge \mathrm{UNI}.$$

It follows immediately from its definition that this formula is satisfiable iff CONSTRUCTOR has a forced win in the rectangle-tiling game determined by T and n. ∎

Propositional Modal Logic (K)

This logic can be defined as PDL without the formulas of the form $([P*]\, \varphi)$ and $(\langle P* \rangle\, \varphi)$.

7.3. PROPOSITION (Ladner [15]). SAT(K) *is complete in PSPACE.*

Proof. We show that SQUARE TILING GAME is reducible to SAT(K). Let T and n determine an instance of SQUARE TILING GAME, and assume that n is even and the bottom row is uniquely determined. Define $[P]^i$ to be the sequence $[P]\cdots[P]$ of length i, for $i \geqslant 0$. Transform the subformulas from the previous proof to CON′, SAB′, BOT′, WHI′ and UNI′ in the following way:

(1) substitute $[P]^{n-1}$ for $[P*]\langle P*\rangle$ in WHI;

(2) substitute $\bigwedge_{0 \leqslant i < n} [P]^i$ for the remaining occurrences of $[P*]$.

Define the final formula as

$$\mathrm{CON}' \wedge \mathrm{SAB}' \wedge \mathrm{BOT}' \wedge \mathrm{WHI}' \wedge \mathrm{UNI}' \wedge \bigwedge_{0 \leqslant i < n-1} [P]^i \langle P\rangle \mathrm{TRUE}.$$

It is satisfiable iff CONSTRUCTOR has a forced win in the square-tiling game determined by T and n. ∎

Propositional Dynamic Logic with the Double-Star Programs (PDL^+)

This logic was introduced by Chlebus [3]. It is an extention of PDL, the additional programs being of the form $\langle \xi_1, \ldots, \xi_n P** \rangle$ or of its "universal" counterpart with square brackets. Programs of the form $\langle \xi_1, \ldots, \xi_n P** \rangle$ have arbitrary strings of formulas as parameters, and describe the following action: repeat executing nondeterministically program P until all the formulas $\xi_1, \ldots, \xi_n$ have their logical values restored.

7.4. PROPOSITION (Chlebus [3]). *SAT(PDL^+) is complete in 2EXPTIME.*

Proof. We show that HIGH TILING GAME is reducible to SAT(PDL$^+$). Let a set T, an even number n and a relation A determine an instance of this problem, and assume that the game determined by it always terminates. Define SEQ as the sequence of all the variables of the form $v(d, i)$ for $d \in T$ and even numbers i such that $1 < i \leqslant n$; define $a(d) = \{x \in T\colon (d, x) \in A\}$, for $d \in T$. Then the conjunction of the final formula from the proof of Proposition 7.2 and the following:

$$[P*]\left(\bigwedge_{d \in T} v(d, 1) \rightarrow [\mathrm{SEQ}\ P**] \bigvee_{x \in a(d)} v(x, 1)\right)$$

is satisfiable iff CONSTRUCTOR has a forced win in the high-tiling game determined by T, n, and A. ∎

Similarly, the satisfiability problem of the deterministic counterpart of PDL$^+$ can be shown to be complete in EXPSPACE by a reduction from HIGH TILING (cf. [3]).

ACKNOWLEDGMENTS

The author would like to express his gratitude to Antoni Kreczmar for his guidance and constant inspiration in this research, and to the anonymous referee for his helpful suggestions.

REFERENCES

1. R. BERGER, "The Undecidability of the Domino Problem," Mem. Amer. Math. Soc. Vol. 66, Amer. Math. Soc., Providence, R.I., 1966.
2. A. K. CHANDRA, D. C. KOZEN, AND L. J. STOCKMEYER, Alternation, *J. Assoc. Comput. Mach.* **28** (1981), 114–133.
3. B. S. CHLEBUS, On the computational complexity of satisfiability in propositional logics of programs, *Theoret. Comput. Sci.* **21** (1982), 179–212.
4. B. S. CHLEBUS, Ph.D. dissertation, Warsaw University, Warsaw, 1982.
5. P. VAN EMDE BOAS, Dominoes are forever, *in* "1st GTI Workshop," Paderborn, 1983, pp. 75–95.
6. M. J. FISCHER, AND R. E. LADNER, Propositional dynamic logic of regular programs, *J. Comput. System Sci.* **18** (1979), 194–211.
7. A. S. FRAENKEL AND D. LICHTENSTEIN, Computing a perfect strategy for $n \times n$ chess requires time exponential in n, *J. Combin. Theory Ser. A* **31** (1981), 199–214.
8. M. R. GAREY, AND D. S. JOHNSON, "Computers and Intractability: A Guide to the Theory of *NP*-Completeness," Freeman, San Francisco, 1979.
9. D. HAREL, A simple highly undecidable domino problem (or, a lemma on infinite trees with applications) *in*, "Proc. Logic and Computation Conference," Clayton, Victoria, Australia, 1984.
10. D. HAREL, "Recurring Dominoes: Making the Highly Undecidable Highly Understandable," Lecture Notes in Computer Sci. Vol. 158, pp. 177–194, Springer–Verlag, Berlin 1983; *Annals of Discrete Math.*, in press.
11. F. C. HENNIE, AND R. E. STEARNS, Two tape simulation of multitape machines, *J. Assoc. Comput. Mach.* **13** (1966), 533–546.
12. J. E. HOPCROFT, AND J. D. ULLMAN, "Introduction to Automata Theory, Languages and Computation," Addison–Wesley, Reading, Mass., 1979.
13. N. D. JONES, Space bounded reducibility among combinatorial prolems, *J. Comput. System Sci.* **11** (1975), 68–85.
14. A. S. KAHR, E. F. MOORE, AND H. WANG, Entscheidungsproblem reduced to the $\forall\exists\forall$ case, *Proc. Nat. Acad. Sci. U.S.A.* **48** (1962), 365–377.
15. R. E. LADNER, The computational complexity of provability in systems of modal propositional logic, *SIAM J. Comput.* **6** (1977), 467–480.
16. L. A. LEVIN, Universal sequential search problems, *Problems Inform. Transmiss.* **9** (1973), 265–266.
17. H. R. LEWIS, Complexity of solvable cases of the decision problem for the predicate calculus, *in* "19th Found. of Comput. Sci.," 1978, pp. 35–47.
18. H. R. LEWIS, "Unsolvable Classes of Quantificational Formulas," Addison–Wesley, Reading, Mass., 1979.
19. H. R. LEWIS, AND C. H. PAPADIMITRIOU, "Elements of the Theory of Computation," Prentice–Hall, Englewood Cliffs, N.J., 1981.
20. A. R. MEYER, AND L. J. STOCKMEYER, The equivalence problem for regular expressions with squaring requires exponential space, *in* "Proc. 13th IEEE Sympos. on Switching and Automata Theory," 1972, pp. 125–129.
21. G. L. PETERSON, AND J. H. REIF, Multiple-person alternation, *in* "Proc. 20th Annu. Sympos. on Foundations of Computer Science," 1979, pp. 348–363.
22. M. P. W. SAVELSBERG, AND P. VAN EMDE BOAS, BOUNDED TILING, an alternative to SATISFIABILITY?, *in* "Proc. 2nd Frege Conference, Schwerin 1984," (G. Wechsung, Ed.), Mathematische Forschung Vol. 20, pp. 354–363, Akademie Verlag, Berlin, 1984.

23. W. J. SAVITCH, Relationship between nondeterministic and deterministic tape complexities, *J. Comput. System Sci.* **4** (1970), 177–192.
24. J. I. SEIFERAS, Relating refined space complexity classes, *J. Comput. System Sci.* **14** (1977), 100–129.
25. L. J. STOCKMEYER, The polynomial-time hierarchy, *Theoret. Comput. Sci.* **3** (1976), 1–22.
26. L. J. STOCKMEYER, AND A. K. CHANDRA, Provably difficult combinatorial games, *SIAM J. Comput.* **8** (1978), 151–174.
27. L. J. STOCKMEYER, AND A. R. MEYER, Word problems requiring exponential time: Preliminary report, *in* "Proc. 5th ACM Sympos. on Theory of Computing," 1973, pp. 1–9.
28. H. WANG, Proving theorems by pattern recogniion II, *Bell System Tech. J.* **40** (1961), 1–4.