is any string $w \in \Sigma^*$ such that, for some $n > 0$ and some (not necessarily distinct) pairs $(u_1, v_1), (u_2, v_2), \ldots, (u_n.v_n)$, $w = u_1 u_2 \ldots u_n = v_1 v_2 \ldots v_n$.

(a) Show that it is undecidable, given a Post correspondence system, to determine whether it has a match. (Start with a restricted version, in which matches must start with a distinguished pair in P.)

(b) Use (a) above to find an alternative proof of Theorem 5.5.2.

5.5.3. A (nondeterministic) **2-head finite automaton** (THFA) consists of a finite control, an input tape, and two heads that can read but not write on the tape and move only left to right. The machine is started in its initial state with both heads on the leftmost square of the tape. Each transition is of the form (q, a, b, p), where q and p are states, and a and b are symbols or e. This transition means that M may, when in state q, read a with its first head and b with its second and enter state p. M accepts an input string by moving both heads together off the right end of the tape while entering a designated final state.

(a) Show that it is solvable, given a THFA M and an input string w, whether M accepts w.

(b) Show that it is unsolvable, given a THFA M, whether there is any string that M accepts. (Use the previous problem.)

5.6 AN UNSOLVABLE TILING PROBLEM

We are given a finite set of *tiles*, each one unit square. We are asked to tile the first quadrant of the plane with copies of these tiles, as shown in Figure 5-1. We have an infinite supply of copies of each tile.

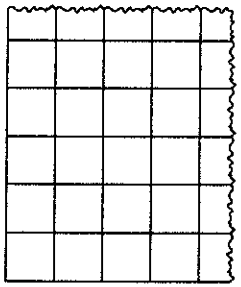

Figure 5-1

The only restrictions are that a special tile, the *origin tile*, must be placed in the lower left-hand corner; that only certain pairs of tiles may abut each other

horizontally; and that only certain pairs of tiles may abut each other vertically. (Tiles may not be rotated or turned over.) Is there an algorithm for determining whether the first quadrant can be tiled, given a finite set of tiles, the origin tile, and the adjacency rules?

This problem can be formalized as follows. A **tiling system** is a quadruple $\mathcal{D} = (D, d_0, H, V)$, where D is a finite set of **tiles**, $d_0 \in D$, and $H, V \subseteq D \times D$. A **tiling** by $\mathcal{D}$ is a function $f : \mathbf{N} \times \mathbf{N} \mapsto D$ such that the following hold:

$$\begin{aligned} f(0,0) &= d_0, \\ (f(m,n), f(m+1,n)) &\in H \quad \text{for all } m,n \in \mathbf{N}, \\ (f(m,n), f(m,n+1)) &\in V \quad \text{for all } m,n \in \mathbf{N}. \end{aligned}$$

Theorem 5.6.1: *The problem of determining, given a tiling system, whether there is a tiling by that system is undecidable.*

Proof: We reduce to this tiling problem the problem of determining, given a Turing machine M, whether M *fails to halt* on input e. This is simply the *complement* of the halting problem, and thus an undecidable problem. If this problem can be reduced to the tiling problem, the tiling problem is surely undecidable.

The basic idea is to construct from any Turing machine M a tiling system $\mathcal{D}$ such that a tiling by $\mathcal{D}$, if one exists, represents an infinite computation by M starting from the blank tape. Configurations of M are represented horizontally in a tiling; successive configurations appear one above the next. That is, the horizontal dimension represents the tape of M, while the vertical dimension stands for time. If M never halts on the empty input, successive rows can be tiled *ad infinitum;* but if M halts after k steps, it is impossible to tile more than k rows.

In constructing the relations H and V, it is helpful to think of the edges of the tiles as being marked with certain information; we allow tiles to abut each other horizontally or vertically only if the markings on the adjacent edges are identical. On the horizontal edges, these markings are either a symbol from the alphabet of M or a state-symbol combination. The tiling system is arranged so that if a tiling is possible, then by looking at the markings on the horizontal edges between the nth and $(n+1)$st rows of tiles, we can read off the configuration of M after $n-1$ steps of its computation. Thus only one edge along such a border is marked with a state-symbol pair; the other edges are marked with single symbols.

The marking on a vertical edge of a tile is either absent (it only matches vertical edges also with no marking) or consists of a state of M, together with a "directional" indicator, which we indicate by an arrowhead. (Two exceptions

are given under (e) below.) These markings on the vertical edges are used to communicate a left- or right-hand movement of the head from one tile to the next.

To be specific, let $M = (K, \Sigma, \delta, s, H)$. Then $\mathcal{D} = (D, d_0, H, V)$, where D contains the following tiles:

(a) For each $a \in \Sigma$ and $q \in K$, the tiles illustrated in Figure 5-2, which simply communicate any unchanged symbols upwards from configuration to configuration.

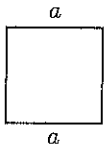

Figure 5-2

(b) For each $a \in \Sigma$ and $q \in K$ such that $\delta(q, a) = (p, b)$, where $p \in K$ and $b \in \Sigma$, the tile shown in Figure 5-3. This tile communicates the head position upwards and also changes the state and scanned symbol appropriately.

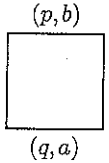

Figure 5-3

(c) For each $a \in \Sigma$ and $q \in K$ such that $\delta(q, a) = (p, \rightarrow)$ for some $p \in K$, and for each $b \in \Sigma - \{\triangleright\}$,the tiles shown in Figure 5-4. These tiles communicate head movement one square from left to right, while changing the state appropriately. Notice that, naturally enough, we allow no state to enter the left-end symbol $\triangleright$ from the left.

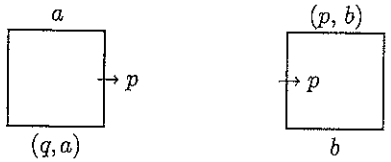

Figure 5-4

(d) Tiles similar to those of (c) for the case in which $\delta(q, a) = (p, \leftarrow)$ are illustrated in Figure 5-5. The symbol $\triangleright$ is not excepted here.

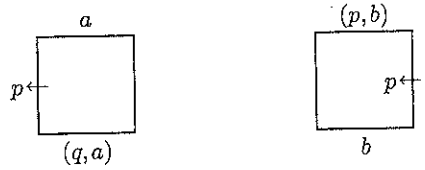

Figure 5-5

These tiles do the bulk of the simulation of M by $\mathcal{D}$. It remains only to specify some tiles to initiate the computation and ensure that the bottom row is tiled correctly.

(e) The origin tile d_0 is illustrated in Figure 5-6(a). It specifies on its top edge the initial state s of M and the symbol $\triangleright$. That is, instead of having M start at configuration $(s, \triangleright\sqcup)$, we instead think that it starts at $(s, \geq)$; by our convention concerning $\triangleright$, its next configuration will definitely be $(s, \triangleright\sqcup)$. The right edge of the origin tile is marked with the blank symbol; this edge can be matched only by the left edge of our last tile, shown in Figure 5-6(b), which in turn propagates to the right the information that the top edge of every tile in the bottom row is marked with the blank.

Figure 5-6

This completes the construction of $\mathcal{D}$. The set of tiles under (e) ensures that the border between the first two rows is marked $(s, \triangleright)\sqcup\sqcup\sqcup\ldots$; the other tiles force each subsequent border to be marked correctly. Note that no tile mentions any halt state, so that if M halts after n steps, only n rows can be tiled.

Example 5.6.1: Consider the Turing machine $(K, \Sigma, \delta, s, \{h\})$, where $\Sigma = \{\triangleright, \sqcup\}$, $K = \{s, h\}$, and δ is given by

$$\delta(s, \triangleright) = (q, \rightarrow),$$
$$\delta(s, \sqcup) = (s, \leftarrow).$$

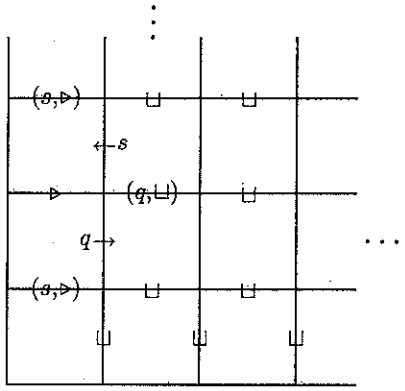

Figure 5-7

This machine simply oscillates its head from left to right and back again, never moving beyond the first tape square. The tiling of the plane associated with the infinite computation of M is shown in Figure 5-7.◇

Problems for Section 5.6

5.6.1. Let $M = (\{s\}, \{a, \sqcup, \triangleright\}; \delta, s)$, where $\delta(s, \sqcup) = (s, a)$, and $\delta(s, a) = (s, \rightarrow)$. Find the set of tiles associated with M via the construction in this section, and illustrate the first four rows of a tiling of the plane by means of these tiles.

5.6.2. Show that there is some fixed set of tiles D and adjacency rules H and V such that the following problem is undecidable: Given a partial tiling, that is, a mapping $f : S \mapsto D$ for some finite subset $S \subseteq \mathbf{N} \times \mathbf{N}$ such that f obeys the adjacency rules, can f be extended to a tiling of the whole plane?

5.6.3. Suppose the rules of the tiling game are changed so that instead of fixing a particular tile to be placed at the origin, we fix instead a particular set of tiles and stipulate that only these tiles may be used in tiling the first row. Show that the tiling problem remains undecidable

5.6.4. Suppose the rules of the tiling game are changed as follows: The tiles are not perfectly square, but may have various bumps and notches along their edges. Two tiles may be laid down next to each other only if their edges fit together perfectly, like pieces of a jigsaw puzzle; and only tiles with perfectly straight sides can be used at the edges. Show that the tiling problem remains undecidable, even if we are now allowed to rotate tiles or turn them over. (There is no specified "origin tile" in this version.)

5.6.5. Suppose that we think of square tiles as being determined by the colors of their four edges, and that two edges may abut provided that they are similarly colored. Show that if we are allowed to rotate tiles and turn them over, then any nonempty set of tiles can be used to tile the entire first quadrant (even when we continue to require that one special tile be placed at the origin).

5.7 PROPERTIES OF RECURSIVE LANGUAGES

We have already seen that every recursive language is recursively enumerable, but the two classes are not the same: The language H is one that bears witness to the difference. Which recursively enumerable languages are recursive? There are many ways to answer this question; we present one next.

Theorem 5.7.1: *A language is recursive if and only if both it and its complement are recursively enumerable.*

Proof: If L is recursive, then L is recursively enumerable by Theorem 4.2.1; also, $\overline{L}$ is recursive, and hence recursively enumerable, by Theorem 4.2.2.

For the other direction, suppose that L is semidecided by M_1 and $\overline{L}$ is semidecided by M_2. Then we can construct a Turing machine M that decides L. For convenience, we describe M as a 2-tape machine; by Theorem 4.3.1, M can be simulated by a 1-tape machine. The machine M begins by putting its input string w on both tapes and placing its heads at the right ends of both copies of the input. Then M simulates both M_1 and M_2 in parallel: at each step of the operation of M, one step of M_1's computation is carried out on the first tape, and one step of M_2's computation is carried out on the second tape. Since either M_1 or M_2 must halt on w, but not both, M eventually reaches a situation in which the simulated version of either M_1 or M_2 is about to halt. When this happens, M determines which of M_1 and M_2 was about to halt, and halts on y or n accordingly. ■

There is an interesting alternative characterization of recursively enumerable languages: They are precisely those that can be *enumerated* by some Turing machine.

Definition 5.7.1: We say that a Turing machine M **enumerates** the language L if and only if, for some fixed state q of M,

$$L = \{w : (s, \triangleright\underline{\sqcup}) \vdash_M (q, \triangleright\underline{\sqcup}w)\}.$$