ON THE COMPLEXITY OF MOTION PLANNING FOR MULTIPLE INDEPENDENT OBJECTS; PSPACE HARDNESS OF THE "WAREHOUSEMAN'S PROBLEM".(*)

J.E. Hopcroft

Computer Science Department,

Cornell University

J.T. Schwartz

Computer Science Department,

Courant Institute of Mathematical Sciences

and

M. Sharir

School of Mathematical Sciences

Tel Aviv University

ABSTRACT

Coordinated motion planning for a large number of three dimensional objects in the presence of obstacles is a computational problem whose complexity it seems important to calibrate. In this paper we show that even the restricted 2-dimensional problem for arbitrarily many rectangles in a rectangular region is *PSPACE*-hard. This result should be viewed as a guide to the difficulty of the general problem and lead researchers to consider more tractable restricted classes of motion problems of practical interest.

(*) Work on this paper by the first author has been supported in part by National Science Foundation Grant MCF 81-01220. Work by the second and third authors has been supported in part by Office of Naval Research Grant N00014-82-K-0381, by a Grant from the U.S.-Israeli Binational Science Foundation, and by Grants from the Digital Equipment Corporation and the Sloan Foundation.

C.1

1. Introduction

In this paper we prove that the *coordinated motion planning* problem for a collection of disjoint 2-dimensional rectangular objects constrained to move within a 2-dimensional rectangular box is *PSPACE*-hard. The problem considered can be defined as follows: Given an initial and a final configuration of the objects in question, determine whether there exists a continuous coordinated motion of the objects between the initial and final configurations during which they do not penetrate either the 'walls' of the enclosing box, or each other.

Previous work on the complexity of motion planning problems of this sort was begun in [Re], where the motion planning problem for a certain kind of many-jointed 3-D system constrained to move within a complex system of narrow tunnels was shown to be *PSPACE*-hard. It was subsequently shown in [SS1] that the general motion-planning problem can be solved in polynomial time, provided that the number of degrees of freedom of the moving body or bodies is held fixed (and provided that the system is 'algebraic'; see [SS1] for detail). This motivates our attempt to find moving systems (necessarily involving arbitrarily many degrees of freedom) whose geometry is as simple as possible, but for which the motion planning problem is still *PSPACE*-hard.

A simple class of systems having these properties is described in [HJW]; it consists of a family of systems consisting of many links, all lying in the plane, connected to each other at specified joints, some of which are fixed to the plane in which the links move. [HJW] shows that the motions of such a system can simulate an arbitrary linear bounded automaton, so that planning the motion of such a system is *PSPACE*-hard. (An interesting feature of the systems described in [HJW] is that their motion is not constrained at all by obstacles.)

Here we consider the case of the coordinated motion of several independent objects. (Earlier work on planning such motions for two and three circular 2-D objects amidst polygonal obstacles is found in [SS2].) The systems on which this paper concentrates have a very simple geometric structure, involving only 2-D rectangular objects and just one bounding rectangular obstacle. Nevertheless we prove that the problem of planning the motion of such a system is *PSPACE*-hard

by showing that such systems, like the systems described in [HJW], can simulate arbitrary linear bounded automata. This is achieved via a sequence of reductions, which start with a formal string manipulation problem, proceed through a motion problem for systems of moving independent bodies whose shape is somewhat irregular, and finally reduce to the simple 'rectangular' motion problem described above.

2. Formal Considerations.

Even though the ultimate aim of this paper is to show that a certain type of coordinated motion planning problem is *PSPACE*-hard, we begin with a combinatorial problem. Specifically we observe that a deterministic linear bounded automaton can be simulated by a rewriting system all of whose productions are of one of the following two forms: $AB \rightarrow AC$ or $AB \rightarrow CB$. Furthermore, the simulation can begin with a string S of length n and remain subject to the condition that at each moment only two productions are applicable, and both those productions overlap in at least one symbol of the string being manipulated. If these productions overlap in precisely one symbol X, then both attempt to modify X. More precisely, suppose in this latter case that near X the string S has the form $\cdots AXB \cdots$. Then the two applicable productions must be of the form $AX \rightarrow AY$ and $XB \rightarrow ZB$. With these restrictions, and even if the set of productions is required to be *reversible*, i.e. $AC \rightarrow AB$ (resp. $CB \rightarrow AB$) appears whenever $AB \rightarrow AC$ (resp. $AB \rightarrow CB$) does, the question as to whether a given symbol ever appears in the string is *PSPACE*-complete. (See [HJW], p. 24 bottom, *ff*.) Similarly, the question as to whether the string S can be transformed into another string S' in this manner is also *PSPACE*-complete.

Our first simple step in moving from this conceptual starting point towards a motion-planning problem is to transform the *production system rewrite problem* described in the preceding paragraph to another symbol manipulation problem that is easier to simulate by means of a set of object motions. This is the *symbol transposition problem* defined as follows:

(i) A finite string S of symbols is given. Among these there occur two special 'bracket' symbols, which we will write as '[' and ']'.

(ii) A sequence of elementary 'moves' can be applied to the string S. Each such 'move' acts by taking a single (non-bracket) symbol and changing its position within S. However, these moves are related by a certain set of *adjacency rules*, which constrain the symbols which can stand next to each other. (The specific adjacency rules we require are described in detail below.)

(iii) The transposition problem is: to determine whether there exists a sequence of moves which takes S into a specified string S'.

We use this symbol transposition problem as an intermediate step in showing certain motion-planning problems involving physical objects to be *PSPACE*-hard. In rewriting systems symbols appear or disappear. In the case of physical objects we need a reservoir for storing objects not currently used to encode the string of the transposition problem. It is for this reason that the brackets are introduced. Simulation of a rewriting system, subject to the conditions stated above, by a transposition system goes as follows. The string of symbols up to the leftmost bracket corresponds to the string of symbols in an ordinary rewrite system. The symbols after the left-most bracket represent a 'pool' of 'spare' symbols held for possible future use. Note that in a symbol transposition problem the collection of symbols being manipulated remains invariant and so corresponds more closely to the motion of physical objects.

We set up the following transposition problem:

(a) The (non-bracket) symbols of the transposition problem are as follows:

(a.1) For each symbol A of the rewrite problem, we introduce three symbols A_0, A_1 and A_2.

(a.2) If the i-th production of the production system is $AB \rightarrow AC$, we introduce three special symbols M^i_{01}, M^i_{12}, and M^i_{20}. Similarly, if the i-th production of the production system is $BA \rightarrow CA$, we introduce three special symbols N^i_{01}, N^i_{12}, and N^i_{20}.

(a.3) We introduce two distinguished nonspecial symbols Λ and Γ.

(b) The adjacency rules of the transposition system are as follows:

(b.1) If the nonspecial symbols D_i and E_j are adjacent (with D_i lying to the left of E_j) in the portion of the transposition string left of Γ, then $j = i+1$ (*mod* 3).

(b.2) Using the notations introduced in (a.2) above: the symbol M^i_{jk} can only have the symbol A_j to its left and only the symbols B_k, C_k, or any symbol E_l with $l = k+1$ (*mod* 3), to its right. (Note here that the i-th production is $AB \to AC$). Similar rules apply to N^i_{jk}, interchanging left and right in the above.

(c) Given an initial string $ABCDE \cdots$ for the rewrite problem we introduce a corresponding initial string for the transposition problem. This has the form:

(1) $\Lambda A_0 B_1 C_2 D_0 E_1 \cdots \Gamma\ [M^1_{01}]\ [M^1_{12}]\ [M^1_{20}] \cdots [M^k_{01}]\ [M^k_{12}]\ [M^k_{20}] \cdots$
$[\,[S^1_0]\ [S^1_0] \cdots [S^1_0]\ [\,]\ [\,] \cdots [\,]\,] \cdots$
$[\,[S^n_0]\ [S^n_0] \cdots [S^n_0]\ [\,]\ [\,] \cdots [\,]\,] \cdots$
$[\,[S^n_2]\ [S^n_2] \cdots [S^n_2]\ [\,]\ [\,] \cdots [\,]\,]$

Here the sequence $A_0 B_1 C_2 D_0 E_1 \cdots$ of symbols preceding the first bracket duplicates the given string $ABCDE \cdots$, but with addition of subscripts 0,1,2 which are attached in the repeating sequence 0, 1, 2, 0, 1, 2, , . . . , . Next follow a bracketed sequence of special symbols, one each of the symbols M^i_{01}, M^i_{12}, M^i_{20}, $i=1, \ldots, N$ occuring. Finally, if $S^1, \ldots, S^m$ is the full sequence of symbols of the production system, there follows a series of bracketed sequences $[\,[S^i_j] \cdots [S^i_j]\ [\,] \cdots [\,]\,]$, $i=0,1,2$, and $j=1, \ldots, m$; each of these sequences has the same length as the initially given string $ABCDE \cdots$ and the number of empty brackets in the sequence is equal to the number of times the symbol S^i_j appears in the initial given string left to Γ.

Only that part of the string (1) which stands to the left of the symbol Γ is considered to be 'significant': the remainder is regarded as a mere 'pool' of symbols which can be moved into this significant part, and also as a 'storage area' into which symbols moved from the significant part of (1) can be placed. The transposition problem we seek to solve is: can the string (1) be transformed, by a valid sequence of moves, into

(2) $\Lambda F_0 G_1 H_2 I_0 J_1 \cdots \Gamma\ [M^1_{01}]\ [M^1_{12}]\ [M^1_{20}] \cdots [M^k_{01}]\ [M^k_{12}]\ [M^k_{20}] \cdots$
$[\,[S^1_0]\ [S^1_0] \cdots [S^1_0]\ [\,]\ [\,] \cdots [\,]\,] \cdots$
$[\,[S^n_0]\ [S^n_0] \cdots [S^n_0]\ [\,]\ [\,] \cdots [\,]\,] \cdots$
$[\,[S^n_2]\ [S^n_2] \cdots [S^n_2]\ [\,]\ [\,] \cdots [\,]\,]$

where $FGHIJ \cdots$ is the target string for the corresponding rewrite problem, and where the

length of the bracketed sequences [$[S_i]$ $[S_i]$ $\cdots$ $[S_i]$ [ ] $\cdots$ [ ]] equals the length of the string $FGHIJ \cdots$, with the number of empty brackets being equal to the number of times S_i appears in the target string $FGHIJ \cdots$.

To demonstrate the reduction claimed, we must show that (2) can be derived from (1) by a series of legal moves if and only if $FGHIJ \cdots$ can be derived from $ABCDE \cdots$ by a series of productions. In one direction this is easy, since the effect of applying any production can be simulated by a series of moves. Suppose for definiteness that the production applied is $XY \rightarrow XZ$, that this is the i-th production of the production system, and that the symbol X of the symbol pair XY to which it is applied occurs at a position equal to zero *mod* 3 within some string $\cdots XYW \cdots$. Then we can transform the symbols $\cdots X_0Y_1Y_2 \cdots$ in the significant part of the corresponding transposition problem string successively as follows:

The string	$\cdots X_0Y_1W_2 \cdots$
goes to	$\cdots X_0M^i_{01}Y_1W_2 \cdots$
goes to	$\cdots X_0M^i_{01}W_2 \cdots$
goes to	$\cdots X_0M^i_{01}Z_1W_2 \cdots$
goes to	$\cdots X_0Z_1W_2 \cdots$

(Note that the characters which appear and disappear in the sequence of steps are actually moved from/to the non-significant 'storage' portion of the transposition problem string, which is not shown.) It is clear that all other possible cases can be handled similarly.

To show the converse, we argue as follows. Consider the manner in which the ('significant') portion of a transposition string, i.e. that part

$$\Lambda A_0B_1C_2D_0E_1 \cdots \Gamma [\cdots$$

which lies to the left of the symbol Γ, changes during a sequence of moves. Since the subscripts of nonspecial symbols must cycle in the pattern 0,1,2,0,1,2, . . . , no single move can either remove a symbol from this significant substring or introduce any nonspecial character.

Hence the only possible moves consist of introducing a special symbol between a pair of symbols XY that appear on the left hand side of a production. There are only two such possibilities corresponding to the two productions applicable at a given moment. Note that either the left hand sides of these two productions coincide, in which case the two corresponding special symbols (which we will denote respectively as M and M') have to be inserted at the same place, or these left hand sides overlap at just one symbol X, in which case the locations where M and M' can be inserted are separated by X. Recall that in this latter case both productions would change the same symbol X.

First consider the situation when the positions are the same. If M is inserted to give $\cdots WMXZ \cdots$ (where the subscripts 0,1,2 are suppressed for the sake of convenience), and $WX \to WY$ is the corresponding production, then the only possible moves are to remove M (a possibility which we ignore, since this merely 'cancels' the insertion of M), or to delete X. After X is deleted the only possible move other than re-inserting it is to insert Y and then to delete M. This sequence of moves converts the substring WXZ to WYZ, which is precisely the effect of applying the production $WX \to WY$. An analogous situation occurs if M' is inserted instead of M. Note that, strictly speaking, the final deletion of M may not be the only possible move after inserting Y. M can be left where it is. The possibility that this leads to is analyzed carefully in the following paragraph. In preparation for this discussion we remark that instead of leaving M in the string we can think of M as being deleted and then reinserted; since WY is the left hand side of the production $WY \to WX$, reinsertion of M can simply be viewed as the first step of preparation to apply this production, and hence raises exactly that possibility discussed in the following paragraph.

In the case where the positions into which M and M' may be inserted are separated by a single symbol we must also consider the case where M is inserted and then M' is inserted one symbol further right or left. Suppose for definiteness that M' is inserted one symbol to the right of M. This gives a substring of the form $WMXM'Z$. However, in this case no further move (other than deletion of either M or M') is then possible since M (resp. M') cannot exist in a context in which W (resp. Z) is removed, whereas removing X would leave M and M' adjacent to one another

which is forbidden. Thus one of M or M' must be deleted before progress can be made, and this consideration brings us back to the more limited set of possibilities considered above.

Hence the only way of progressing is to insert a special symbol between a pair of symbols appearing in a production rule. This allows a symbol to be removed and then replaced by the other symbol in the production rule corresponding to the special symbol just inserted. After this the only way of continuing to progress is to remove the special symbol. (Note that the nonspecial characters that are removed from the significant prefix string can always be stored between appropriate brackets in the second part of the string.) Clearly this sequence of actions will always simulate application of successive productions drawn from the production system.

This completes the reduction from the production system rewrite problem to the symbol transposition problem, showing that the latter problem is also *PSPACE*-hard.

3. Representing the transposition problem as a motion problem.

Next we show that the *PSPACE*-hard transposition problem considered above can be reduced to the problem of planning the coordinated motion of a disjoint collection of nearly rectangular objects.

For this purpose, consider a set of rectangular blocks constrained to move within a 2-dimensional rectangular box as shown in Fig. 1. The box is very nearly full: the cross-hatched rectangular region appearing near the right-hand side of the box is the only empty space within the box.

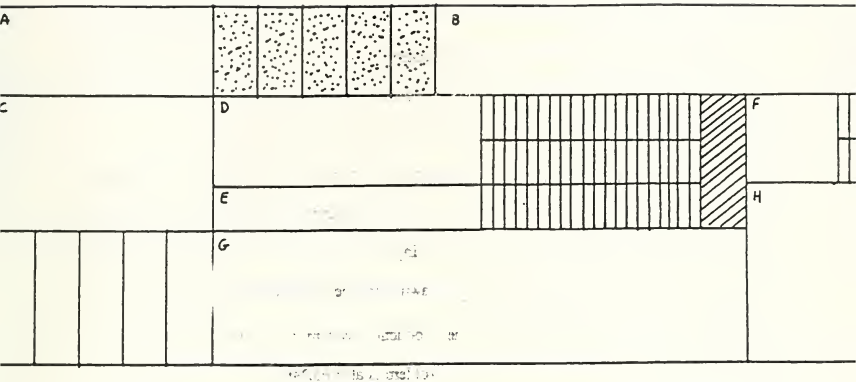

Fig. 1. The general layout of objects in the enclosing box.
All dominoes in upper track; slider D moved as far left as possible.

The bounding rectangular box has dimensions 8 by $8kL$ where L is a horizontal/vertical length factor which we take to be 1 in the following discussion (but which is shown in Figures 1, 2, and 4 below as 1/2), and where k is an integer that will be specified later. The blocks $A, B, \ldots, H$, called *sliders*, have the following sizes:

A	2 *by* $2k$
B	2 *by* $4k-\frac{1}{2}$
C	3 *by* $2k$
D	2 *by* $2k+2$
E	1 *by* $2k+2$
F	2 *by* $2k-6$
G	3 *by* $4k+4$
H	4 *by* $2k-4$

The rectangular box in which all motion takes place contains various other smaller blocks besides the sliders A through H. Between A and B are k 2×2 blocks. These blocks will play the role of *dominoes* whose motions will simulate the transposition problem described in the preceding section; the region running from A through the dominoes and the slider B and to the end of the

box will sometimes be called the *upper track*. The 3×2 blocks to the left of G are called *bricks*, and the (1/2)×1 blocks to the right of D and E are called *fillers*. The only empty space is the 3×2 cross-hatched area shown left of F and H. The items are packed as closely as has been described in order to constrain their motions to just those corresponding to the steps of a transposition problem. As will now be explained, only two basic motions are possible.

Observe first that the boundary between C and D can be moved left and right underneath the k dominoes between A and B. The motion is a kind of simple rotation in which C, D and E all move in one direction, while G moves in the other direction and the bricks and fillers at the ends of D, E and G move to compensate. More precisely, this is accomplished by moving C, D, E and the fillers on the right of D and E; all these move to the right. A brick from the left of G is moved up to the new vacant space on the left of C and G is moved left. Twelve small fillers to the right of D and E are then dropped down into the space between G and H. This process can be repeated or reversed to position the boundary between C and D below any domino between A and B. Since each brick to the left of G has the same size as the available crosshatched free space, accomplishing this motion requires use of all empty space, and hence for this motion to be possible the blocks must be in a configuration that results from the initial configuration shown in Figure 1 by an appropriate clockwise "rotation" of the sliders C, D, E and G and the corresponding bricks and fillers.

In addition to this rotating motion, a second significant motion is possible. Specifically, a gap can be opened between C and D by sliding D right, permitting a 2×2 domino to drop down from the top row. The space in the upper row can then be closed by moving B left, filling the space to the right of B by the small fillers to the right of F and moving F right. Eight fillers must then be moved into the space to the left of F to create the 2×3 vacant space needed for the rotating motion described above. Once this has been done, the domino currently between C and D can be transported left or right (see Fig. 2). Finally this domino can be reinserted back into its original row at a different position. This motion allows us to permute the dominoes among positions between A and B as required to simulate a transposition problem.

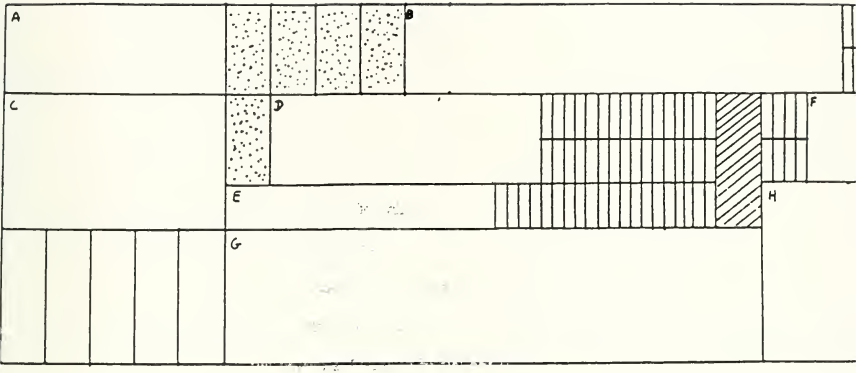

Fig. 2: One domino moved below upper track, and compensating adjustments to prepare for transport of this domino.

To enforce the adjacency rules required by the transposition problem that we want our domino motions to simulate, we can simply add short extrusions and indentations to the dominoes, as shown schematically in Figure 3.

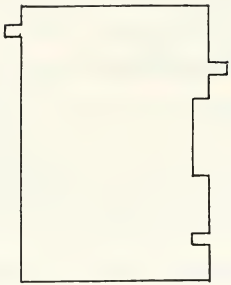

Fig. 3. Extrusions and indentations in the vertical sides of a domino.

Even after this has been done, the dominoes can still be permuted provided that during each permutation, the extrusions and indentations in adjacent dominoes match up, and provided that *C* and *D* contain notches which allow any domino at all to be inserted between them. This is because even when *C* and *D* are moved apart, *A* and *B* can be pulled apart by one half unit to unlock the short extrusions and indentations in the dominoes, and moreover *C* and *D* can be moved an extra half unit to allow the domino to drop down without interference from the small extrusions on it. This is accomplished (see Figure 4) by moving the two half-sized fillers at the right of *B* to a position on the right of *F* (causing *F* to move left by 1/2 unit). *D* can then be moved right two and a half units, creating a two and a half unit opening between *C* and *D*. (This requires moving four of the small fillers from the area right of *D* to the area right of *E*.)

However, whenever the sliders *C* and *D* move to transport a domino or to prepare for such transport, the space between *A* and *B* must be tightly closed; thus the adjacency rules defined by the extrusions and indentations on the dominoes must be satisfied during every such move.

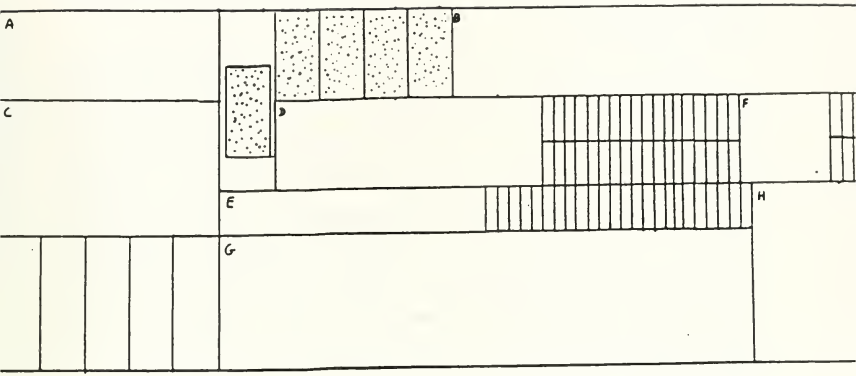

Fig. 4: 'Opening' the upper track and the row immediately below it to permit a domino to be extracted from the upper track and moved down, even in the presence of slight 'tabs' and 'dents' in adjacent dominoes.

Next, we wish to show that by choosing appropriate shapes for the vertical boundaries of the dominoes and brackets we can enforce the adjacency rules described in the preceding section. For this, we simply choose appropriate extrusions and indentations for their sides (and also on the right edge of the slider *C* and the left edge of the slider *D*). An 'extrusion' or 'tab' is simply a small rectangular protrusion, and an 'indentation' or 'dent' is a rectangular indentation of the same size (as in Figure 3).

Non-special dominoes are given their 0, 1, or 2 parity by partitioning their vertical dimension into three regions labeled 0, 1, and 2. We then personalize the dominoes (as symbols of a transposition system) as follows: for the type i domino A, a tab is added on the left side at an otherwise unused vertical interval in region i. Let $j = i+1$ (*mod* 3). Since the domino A must permit adjacencies of type j on its right, for each domino B of type j an indentation is created in A at the vertical location of B's tab. Thus each domino A of type i has one identifying tab in region i on its left and several indentations (corresponding to all possible symbols B of type j) in region j on its right. Clearly this forces the parity of adjacent nonspecial symbols to be 0,1 or 1,2 or 2,0, but

allows any nonspecial character of a given type can be adjacent to any other nonspecial character of the appropriate type.

Next we define the shape of special dominoes. Let $XY \rightarrow XW$ be the i-th production. We will explain the geometry of a special domino $M = M'_{01}$ for the case where X is of type 0 and Y and W are of type 1. M has a tab on its left side in region 1 at some location as yet unused. X has an indentation on its right at this level. Furthermore, X is the only symbol other than a bracket that has an indentation at this position. M has two indentations in region 1, one to accept a tab from a W of type 1, the other to accept a Y of type 1. In addition, M has indentations in region 2 on its right to accept any type 2 symbol. Thus the only symbols that can be adjacent to M other than brackets are an X of the type 0 on the left and Y and W of the type 1 on the right or any symbol of type 2 on the right.

The brackets and also the two special characters Λ and Γ will be rectangles whose width is that of three dominoes. This prevents the brackets from ever entering the 'bucket' which can be opened up between D and E. For every symbol there will be available sufficiently large set of brackets with indentations placed to hold that symbol between a corresponding pair of brackets. This completes the description of the layout of the various objects within the enclosing box. The parameter k appearing in the specification of the layout has to be chosen to be equal to the total length of all dominoes and brackets appearing in the upper track of the box between A and B.

It should now be apparent that the transposition problem is solvable if and only if the problem of moving the objects described above so as to produce a specific sequence of dominoes to the left of the leftmost bracket is solvable. Hence the domino motion planning problem that we have described is *PSPACE*-hard, i.e. we have

Theorem: The problem of deciding whether there exists a continuous motion that takes a collection of disjoint rigid objects, enclosed within a rectangular region, from one specified configuration to another, is *PSPACE*-hard.

4. Reduction of the transposition problem to the motion planning problem for rectangles.

The final step reduces the domino-motion planning problem described in the preceding section to that of planning the motion of rectangular objects enclosed within a rectangular box. In the preceding section dominoes were realized as non-convex objects, roughly rectangular, but having extrusions and indentations that encode the various symbols that these dominoes represent and that serve to enforce the various adjacency constraints essential to our construction. The final step uses the same general layout of rectangular pieces enclosed within a rectangular box. However, each domino is partitioned into a bottom rectangle, a thick vertical spine and a left and right set of rectangles as shown schematically in Figure 5. In this partitioning, longer rectangles are used to represent tabs and shorter rectangles are used to represent dents. The width of the spine is chosen to be larger than four fifths the nominal width of a domino. By properly designing the lengths and heights of the domino pieces, we shall ensure that all new motions that become possible are irrelevant, in the sense that motions of domino pieces which do not correspond to motions of full dominoes always lead to dead-end positions from which no further useful motion is possible unless one reverses such a sequence of 'domino-fragment' motions completely.

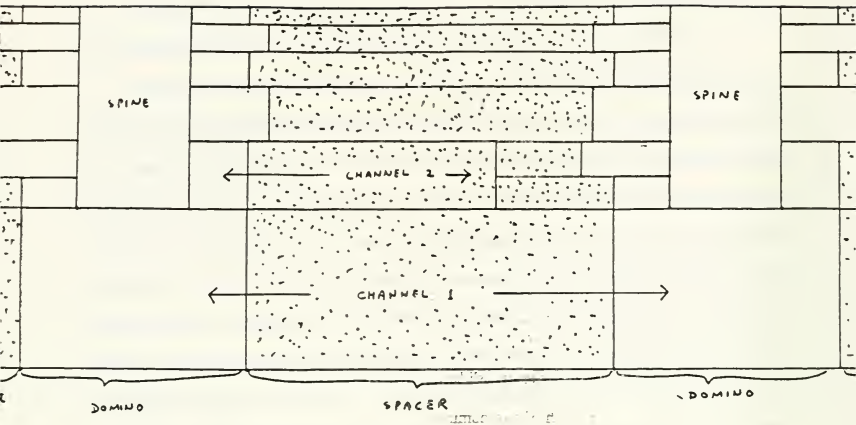

Fig. 5. Schematic representation of the decomposition of dominoes and spacers into rectangles

In defining the dimensions of the domino pieces, it will be convenient to make use of several scales of length that differ by a large constant from one another. We will speak of these scales as being 'infinitesimally small' or 'infinitesimally large' with respect to each other. Since only a fixed number of such scales are used this does not effect the polynomical time aspect of the reduction.

We begin by supposing that all vertical scales are infinitesimal relative to horizontal scales. This implies that blocks cannot be rotated but can only be translated vertically and horizontally.

The new types of motion that might create configurations that no longer encode legal configurations of the transposition problem can be enumerated as follows.

Rearrangement of rectangles within a domino. A rearrangement is possible by opening a space in the upper track (between A and B) next to a domino d, moving pieces of d into this space and then inserting them back into different positions within d, thereby creating a configuration of pieces that no longer represent d.

Exchange of pieces between two dominoes. If there is an open space next to a domino d_1 in the upper track and there is a domino d_2 either adjacent to d_1 or in the bucket directly below the open

space, then it may be possible to interchange pieces between d_1 and d_2.

To prevent these anomalies we introduce a new type of structure, called a *spacer*. Spacers consist of rectangular blocks that are much longer than those constituting a domino (say, 10 times as long). Initially, two of these spacers are placed between each pair of successive dominoes in the upper track. The structure of each such spacer is as shown in Figure 5. Since all pieces constituting a spacer are longer than the width of the bucket, none of them can ever be moved out of the upper track. Furthermore, it is impossible to rearrange the pieces of a spacer or to exchange pieces between spacers. Hence the gross structure of each spacer remains unchanged as dominoes are moved.

It is easily seen that whenever the sliders C and D are able to move left or right by an amount equal to at least the width of two dominoes, there must occur a moment at which both the topmost track (which we will call the *upper track*) and the second track from the top of Fig. 1(a) are filled almost completely. (We will describe this condition by saying that the *upper tracks are closed.)* This observation applies both when nothing is present in the space which can be opened between C and D (which we will call the *bucket*) and when some portion of a domino (and hence, as will be seen below, all parts of a domino) are present in the bucket.

As shown in Figure 5, we regard the area in the upper track that contains the dominoes and the spacers as being divided into a fixed number of horizontal *channels*, and we number these channels starting from the bottom. The j-th channel has a height $h_1/3^{j-1}$ where h_1 is the height of the first channel; thus the first channel occupies more than half the total height of a domino. Each rectangle, other than a domino spine or one of the rectangles in channel 2, has a height differing only infinitesimally from the nominal height of the channel; certain of the rectangles in channel 2 are given a height of only one half the channel height, but even so each of these rectangles occupies something more than the total height of channels 3 through n.

To prevent any 'mixing' between the parts of different dominoes, we simply define the heights of the slabs constituting a given domino to differ by infinitesimal amounts from the nominal height of the channel to which they belong, in the following manner: Call the part of the

domino lying to the left (resp. right) of its spine a left (resp. right) *half domino*. This divides each domino into two domino halves, and accordingly divides the t different types of dominoes into $2t$ different types of domino halves. (We regard the lowermost slab of the domino, which runs horizontally across its whole width, as belonging to both its left and its right half.) Let the precise height of the j-th slab (exclusive of the domino spine) in the pile representing the k-th type of domino half be h_{jk}. Then for any $k \neq m$, and for any two proper nonempty subsets S, S' of the set of channels, we insist that

$$(*) \qquad \sum_{j \in S} h_{jk} \neq \sum_{j \in S'} h_{jm}$$

(with the obvious exception that equality will occur when k and m denote the two halves of the same domino, and either both the sets S and S' consist of just the first lowermost channel, or both consist of all the other channels.) Finally, we insist that for each k the sum of h_{jk}, extended over all channels j, be equal to the unit height of the tracks appearing in Fig. 1. These properties can be obtained by choosing the heights h_{jk} to be appropriate infinitesimal perturbations of the nominal heights of the corresponding channels.

This condition, together with the fact that the slabs on the left and the right of a domino spine have significantly different lengths, implies that whenever the bucket is nonempty and the upper tracks are closed, the bucket contains exactly one spine piece, one full-width slab positioned either above or below the spine, and on one side of the spine all the pieces belonging to one domino half, and on the other side of the spine all the pieces constituting the other half of exactly the same type of domino; moreover, the single full-width (channel 1) slab present in the bucket must belong to the same type of domino as these two piles of half slabs. This is because the height of the full-width slab will add up to the full height of the track only when matched with the heights of the slabs in the left pile and with the heights of the slabs in the right pile of the same domino. Note also that slabs belonging to spacer piles should be given heights slightly smaller than the nominal height of the corresponding channels, to allow their vertical positions to be adjusted to slight (indeed, 'infinitesimal') changes in the vertical positions of adjacent domino slabs.

Once we begin to move pieces of a domino into or out of the bucket, further motion of C and D is impossible until either the bucket becomes entirely empty, or until all the pieces of a complete domino have been moved in some consistent order into the bucket. Hence if motion is not to cease the pieces in the bucket must all be moved to some one end of some spacer; this end region must clearly lie either between two spacers or between a spacer and a domino adjacent to it. This second case can only occur if we place two dominoes d_1, d_2 next to each other, but it is easy to see that further motion of the bucket must cease the first time we do this. For this, note first of all that domino spines are too wide ever to move past the full-width (channel 1) slab contained in a domino; hence in the configuration envisaged both domino spines must lie above a corresponding channel 1 slab. Moreover condition (*) implies that all the pieces of each domino d_1, d_2 must lie above the channel 1 slab belonging to that domino, and that the domino pieces drawn from either side (left or right) of each of these two dominoes must be grouped together. (However, this argument leaves open the possibility that the newly inserted domino should be reversed right-to-left.) But reference to Figure 5 shows at once that we cannot either reverse the parts of any single domino or place two dominoes adjacent to each other in a closed upper track, since doing so would cause a 'tab' in the upper half of channel 2 to collide either with a channel 2 block of an adjacent domino or with the left-hand channel 2 block of a spacer.

Note also that this same argument shows that it is not possible to replace a complete domino in the upper track by the domino in the bucket. Indeed, to do so would require moving all the parts of both dominoes into adjacent positions in the upper track at least momentarily and then moving the bucket, and we have just proved that this is impossible.

Having shown that interchanging pieces of dominoes is impossible, it remains to be shown that permuting rectangles within a domino is also impossible. Permuting the rectangles is prevented by using alternate channels as *separator* channels and confining all the tabs and dents to the channels between separator channels for the tabs and dents. Separator channels are used exlusively to force the domino rectangles to remain unpermuted. The lengths of left and right domino rectangles in separator channels are increased by an amount Δ that is greater than the dif-

ferential used for tabs and dents, and to compensate the lengths of rectangles in separator channels of spacers are reduced by 2Δ apiece. It follows that when the upper track is closed, each nonseparator channel must contain nonseparator rectangles exclusively and all domino rectangles must maintain their original order. Indeed, any permutation of this order would force some separator slab to (partially) overlap a nonseparator channel, which however is too short to accomodate it. Hence dominoes must retain their original structure if the upper track is to close and the bucket to move.

Note also that the dimensions of separator and nonseparator channels permit a domino to be inserted between two adjacent spacers, and that in our initial configuration two spacers are placed between every pair of dominoes in the upper track. This is done to allow a special domino to be placed between two other dominoes. If only one spacer was present, we would be forced to place a domino immediately adjacent to another, but as just seen this would terminate motion.

We remark in conclusion that since the spacer rectangles in nonseparator channels have exactly the same length as the total length of the spacer, it follows that when the upper track is closed, a domino d_A can only follow a domino d_B (with any number > 0 of spacers between them) if for each nonseparator channel, the sum of the length D_r of the right-hand slab of d_A in this channel, plus the length D_l of the left-hand slab of d_B in the same channel, is less than or equal to $3d$. But this is exactly the condition of compatibility between tabs and dents of adjacent dominoes, which completes verification of the assertion that every significant configuration available to our partitioned dominoes is also available to the corresponding intact dominoes.

We have therefore proved our desired result:

Theorem: The coordinated motion planning problem for a collection of 2-D rectangular objects moving inside a 2-D rectangular box is *PSPACE*-hard.

References

[HJW]J. Hopcroft, D. Joseph and S. Whitesides, Movement problems for 2-dimensional linkages, Tech. Rept. 82-575, Computer Science Dept., Cornell University, 1982 (to appear in *SIAM*

J. Computing)

[Re] J. Reif, Complexity of the movers' problem and generalizations, Proc. 20th IEEE Conf. on Foundations of Computer Science, 1979, pp. 421-427.

[SS1] J.T. Schwartz and M. Sharir, On the piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds, *Advances in Appl. Math.* 4(1983) pp. 298-351.

[SS2] J.T. Schwartz and M. Sharir, On the piano movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers, *Robotics Research* 2(1983) (3) pp. 46-75.

NYU CS TR-103 c.1
Hopcroft, J E
On the complexity of motion planning for multiple n