The convenience of tilings

Peter van Emde Boas *

ILLC, Faculteit WINS, Universiteit van Amsterdam,
Plantage Muidergracht 24, 1018 TV Amsterdam
July 15, 1996

Abstract

Tiling problems provide for a very simple and transparent mechanism for encoding machine computations. This gives rise to rather simple master reductions showing various versions of the tiling problem complete for various complexity classes. We investigate the potential for using these tiling problems in subsequent reductions showing hardness of the combinatorial problems that really matter.

We ilustrate our approach by means of three examples: a short reduction chain to the Knapsack problem followed by a Hilbert 10 reduction using similar ingredients. Finally we reprove the Deterministic Exponential Time lowerbound for satisfiablility in Propositional Dynamic Logic.

The resulting reductions are relatively simple; they do however infringe on the principle of orthogonality of reductions since they abuse extra structure in the instances of the problems reduced from which results from the fact that these instances were generated by a master reduction previously.

1 Introduction

This paper presents some results originally obtained in 1982 [31, 29] on the use of tiling problems for combinatorial reductions, together with a more recent application of these ideas. Aside from the wish to present these hardly accessible papers to a broader audience we are looking for an improved understanding why these tiling problems are convenient for use as source for a combinatorial reduction.

As we shall see it is on the one hand the simplicity of the tiling problem combined with the very local structure of the tiling constraints which makes these problems attractive for use in reductions. On the other hand the encoding of computational structures by means of tilings is sufficiently transparent to make information from the computational realm available to the combinatorial world to be invoked when needed. This relates to the distinction between arbitrary reductions and master reductions.

We use the phrase *Master Reduction* for a reduction between a computational problem and a combinatorial problem where there is an intended directly visible correspondence between records of accepting computations for the computational problem and solutions of the combinatorial problem under consideration. The traditional proof of the Cook-Levin

*supported in part by HCM program COLORET (Complexity, logic and recursion theory, nr. CHRX-CT93-0415 (DG 12 COMA))

theorem establishing the NP-completeness of **SATISFIABILITY** is a prime example of a master reduction; the formulas in propositional logic constructed by the reduction are designed in such a way that every satisfying assignment encodes the computation record of an accepting computation.

By providing a reduction f between some problem A which is already shown to be NP-complete and a new problem B a new master reduction to B may be obtained from a master reduction to A by composition, assuming that the reduction f preserves in some reasonable way solutions of the combinatorial problem. It is not trivial to turn this latter requirement into a formal condition; however most reductions which are found in the literature indeed preserve such a correspondence. At the same time the reductions are shown to preserve solvability in a scope much wider than required in order to provide a master reduction to B or to prove NP-hardness of the problem B, since f is also defined on (solvable) instances of A which are outside the range of the original master reduction to A.

In fact, when presenting and proving the reduction f it is almost felt to be unethical to exploit information which follows from the assumption that some particular instance of A indeed is the translation of a computational problem; it is a violation of a *principle of orthogonality for reductions*: the correctness of the reduction f should be based on the combinatorial structure of the problems A and B and nothing else.

In this paper we will discuss a number of reductions which are obtained by violating this principle. We shall consider a well known master reduction towards a tiling problem, and infer from it a number of known completeness results. Next we present a combinatorial reduction between the tiling problem and a set covering problem reproving a known result using a not so well known shortcut. A short treatment of Karp's original reduction from the set-covering problem to the Knapsack problem is presented next.

Extra information is obtained by investigating the combination of these three reductions. It is the insight thus obtained which supports the Hilbert 10 reduction which I originally discovered in 1982; this reduction provides us with an elementary proof of the undecidability of the solvability of Exponential Diophantine equations: the Davis, Putnam and Robinson part of the Hilbert 10 problem.

In the final section we use a two-person game version of the tiling problem as a staring point for a new proof of a well known hardness result about Propositional Dynamic Logic. Also in this case we discover that we must infringe on our orthogonality principle because a termination assumption in the computational world can't easily be expressed as a reasonable condition for the tiling games; instead we must adapt the rules for determining the winner in the tiling games.

2 The Turing machine model and the Cook-Levin theorem

The basic model in the computational world used in this paper is the standard single tape Turing machine. A Turing machine M as a device consists of a finite control, and a reading head which moves over a two-way infinite tape consisting of tape cells. The machine is described by a finite set K of internal states, a finite alphabet Σ of tape symbols, an initial state $q_0 \in K$, a blank symbol $\Box \in \Sigma$ and a program P. The program is a relation $P \subseteq ((K \times \Sigma) \times ((K \cup \{\perp\}) \times \Sigma \times \{L, 0, R\}))$.

The meaning of a quintuple $(q, s, q', s', m) \in P$ is that the machine while in state q and reading the symbol s by its reading head can replace this symbol on the tape by symbol s', change the internal state to q', and perform move m with its reading head on the tape (where $L(R)$ denotes "move one step left (right)" and 0 means "don't move"). We use $q' = \perp$ as an indication for a termination instruction.

A configuration for machine M consists of a finite string of tape symbols, and internal state and an indication which symbol is currently being scanned by the reading head. A configuration is most conveniently encoded by a string of the format $\$\Sigma^*(K \times \Sigma)\Sigma^*\$$. This string encodes the left part of the tape, the scanned symbol together with the internal state and the right part of the tape. Under this encoding the impact of an instruction in P is easily expressed by a context-sensitive production rule. For example the instruction (q, s, q', s', R) corresponds to the collection of production rules $(q, s), t \Rightarrow s'(q', t)$ where $t \in \Sigma$. The rule $(q, s)\$ \Rightarrow s'(q', \Box)\$$ must be added in order to represent the effect of this instruction when performed while scanning the rightmost used symbol on the tape.

A full computation of M on input $x = x_0, x_1, ..., x_{n-1}$ is a sequence of configurations $c_0, \ldots, c_m$ where $c_0 = \$(q_0, x_0)x_1 \ldots x_{n-1}\$$ is the initial configuration corresponding to x, for all $j = 0, \ldots m - 1$ we have that $c_j \Rightarrow c_{j+1}$ according to one of the production rules corresponding the program P, and where c_m is a halting configuration where the last instruction executed had $q' = \perp$. In a configuration we may as well replace the pair $(s, \perp)$ by the symbol s by way of indication that the machine has halted and that the internal state no longer matters.

By convention a regularly terminating computation accepts its input.

Our machine model is nondeterministic in the sense that for a given pair (q, s) more than a single instruction in P may be applicable. It is also allowed that no applicable instruction is available in P; a configuration containing such a pair (q, s) can't have a successor in the sequence. A sequence of configurations terminating in this manner represents a finite rejecting computation. Trere exeist also infinite computations. A machine where in the program for every pair (q, s) there is at most one applicable instruction in P is called deterministic.

The configurations in a computation can be represented in a two dimensional tableau where the symbols representing a single tape square are aligned. This tableau represents the time-space diagram of the computation, with the time running from top to bottom and the space being represented in the horizontal direction. Height and width of this diagram represent the time and space for this computation.

Notwithstanding the fact that time and space defined in this way have obtained a machine based interpretation the resulting measures turn out to be reasonable. Up to a polynomially bounded overhead in time and a constant factor overhead in space the resource requirements for a given algorithm turns out to be independent of the machine model on which it is implemented provided the model is selected within a collection of well studied reasonable sequential models [32].

By way of example consider the Machine M whose program P is given by:

$$\{(q, 0, q, 0, R), (q, 1, q, 1, R), (q, \Box, p, \Box, L), (p, 0, \perp, 1, 0), (p, 1, p, 0, L), (p, \Box, \perp, 1, 0)\}$$

A typical computation of this machine is represented by the time space diagram in Figure 1:

This machine searches in state q for the last input symbol by moving right until a blank is detected; afterwards it moves left in state p, rewriting 1's by 0 until a symbol

(q,0)	1	0	1	1	□
0	(q,1)	0	1	1	□
0	1	(q,0)	1	1	□
0	1	0	(q,1)	1	□
0	1	0	1	(q,1)	□
0	1	0	1	1	(q,□)
0	1	0	1	(p,1)	□
0	1	0	(p,1)	0	□
0	1	(p,0)	0	0	□
0	1	1	0	0	□

Figure 1: Time Space diagram of a computation by the successor machine

0 or blank symbol is detected which subsequently is rewritten by a 1, after which move the machine halts. When considering the input to denote a number written in binary the interpretation of this computation is straightforward: the machine has incremented a binary counter. Turing machines can count, and where counting represents the basis for all mathematics it is no surprise that Turing machines represent one of the many universal formalisms for computability.

The time-space diagram can be used to obtain a direct proof for the Cook-Levin theorem which states the NP-completeness of the problem **SATISFIABILITY** :

SATISFIABILITY :

INPUT: A formula ϕ in the language of the propositional calculus, built from a set of propositional variables $\{x_1, \ldots, x_k\}$

QUESTION: Does there exist a truth value assignment to the variables $\{x_1, \ldots, x_k\}$ such that ϕ under this assignment evaluates to **true**?

Given a nondeterministic machine M which accepts a set of strings L in time (and space) n^d where n denotes the length of the input string w. The question whether a given string w belongs to L can now be expressed by the satisfiability of a formula $\phi(M, w, n, d)$ which is obtained as follows:

The input w belongs to L if and only if there exists a time-space diagram of an accepting computation by M on input w whose width and height is bounded by n^d. Let K and Σ denote the set of states and tape symbols for M. The set of symbols which may occur in the time-space diagram is given by $L = \Sigma \cup (K \times \Sigma)$. Let $t = |L|$. Now introduce propositional variables $q(i, j, l)$ for $i, j = 0, \ldots, n^d, l = 1, \ldots, t$ with the intended meaning: in row i, column j the l-th symbol in L is written. Next consider what it means in terms of these variables that the diagram is well defined, describes a legal computation of the machine M, and accepts.

The formula $\phi(M, w, n, d)$ is the conjunction of a large number of local clauses, each enforcing some properness condition at some position in the diagram, like:

$$q(i,j,1) \vee \ldots \vee q(i,j,t), \quad i,j = 0,\ldots,n^d$$

on every position some symbol is written

$$\neg q(i,j,l_1) \vee \neg q(i,j,l_2), \quad i,j = 0,\ldots,n^d, 1 \leq l_1 \neq l_2 \leq t$$

no two symbols on the same position

$$\neg q(i,j-1,l_1) \vee \neg q(i,j,l_2) \vee \neg q(i,j+1,l_3) \vee \neg q(i+1,j,l_4)$$

for $0 \leq i \leq n^d - 1, 1 \leq j \leq n^d - 1$, and $(l_1, l_2, l_3, l_4) \in C$.

Here C denotes a set of "incompatible" quadruples of symbols, the exclusion of which enforces a proper computation in the diagram. These clauses express that the symbols written in the diagram encode something resembling a computation locally.

To the above clauses one must still add clauses representing the condition that the initial row in the diagram is the initial configuration (a condition easily enforced by a sequence of unit clauses), and that the computation terminates and accepts (a condition enforced by the requirement that on the bottom row in the diagram only symbols in Σ are used.

It is interesting to observe that the formula obtained in this way has size proportional to n^{2d}. The traditional textbook proof of the Cook-Levin theorem [5, 22, 17, 10] produces a much larger formula (size $O(n^{3d})$). The larger size is caused by the part of the formula which enforces that in every configuration in the accepting computation the position of the reading head is uniquely determined. This condition must be stated explicitly when the head position is encoded separately from the rest of the configuration. Basing the proof on the time-space diagram, uniqueness of the head position is a direct consequence of local consistency throughout the diagram, and therefore it does not have to be enforced explicitly.

3 The master reduction

The basic combinatorial problem which we consider in the paper is the *Tiling problem*. A *tile* is a square of unit size which is divided in four triangles by the two diagonals. A *tile type* is obtained by selecting a colour from a finite set of colours for each of the four triangles. Tile types can not be rotated or reflected; the image of a tile type under a symmetry transformation is in general considered to be a different tile type. See Figure 2.

Note that in the example in Figure 2 several tile types actually occur together with their image under a rotation by 180 degrees. This expresses explicitly that these particular tiles may be rotated.

An X-*tiling* is a mapping from a subset of the square grid in the plane to a set of tile types X, or, equivalently, a covering of a region in the plane by instances of the tiles in X. Moreover this tiling has the property that two tiles which share a horizontal or vertical edge in the plane have equal colours for the two triangles adjacent to this edge. See Figure 3.

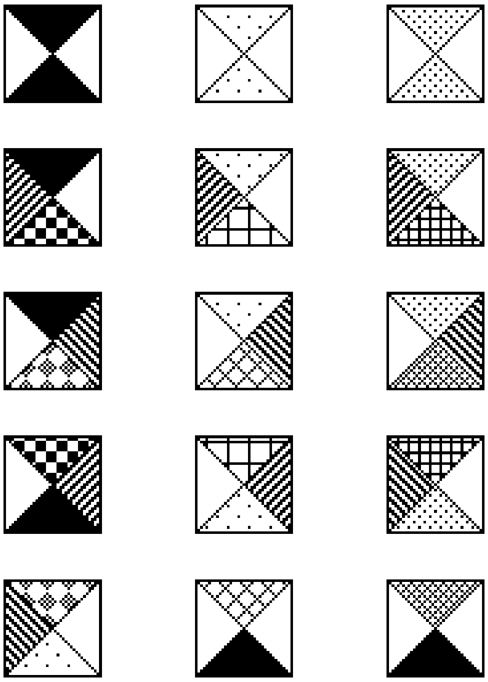

Figure 2: A sample of tile types

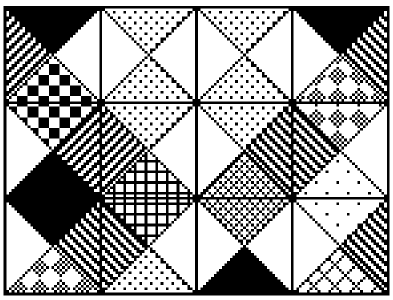

Figure 3: A region legally tiled using the tile types from Figure 2

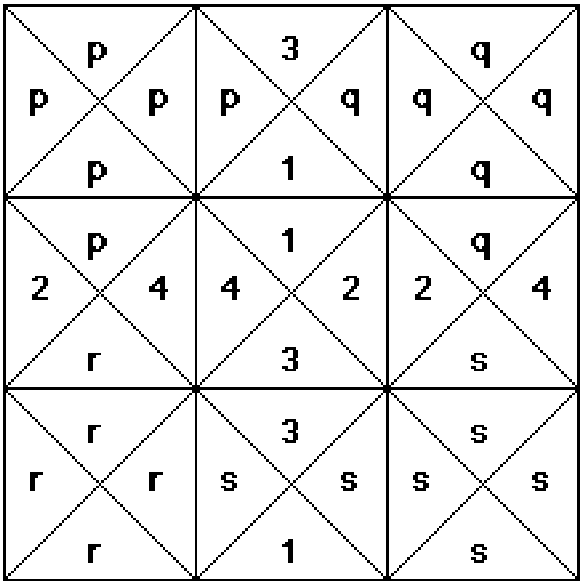

Figure 4: Reduction of arbitrary adjacency relations to matching colours

The effect of this condition is that between tile types there exists a horizontal and a vertical *compatibility relation*: tile type t_1 is horizontally compatible with t_2 if the colours for the right side triangle in t_1 equals the colour of the left triangle in t_2, etc. .

An alternative way of defining tiling problems is to start with an arbitrary horizontal and vertical compatibility relation among a finite set of tile types. Evidently the relations induced by a colouring obey a special structure condition $\mathcal{C}$:

when t_1 is compatible with t_3 and t_4 and t_2 is also compatible with t_3 then t_2 must be compatible with t_4 as well.

So the alternative definition represents a larger class of structures. On the other hand by construction of a suitable set of intermediate tile types a pair of arbitrary compatibility conditions can be encoded using standard coloured tiles at the price of doubling the size of the region to be covered. The construction is sketched in figure 4. Tiles placed at grid points with both coordinates even represent the original tiles; the tiles with coordinates of both parities connecting them represent the given horizontal and vertical compatibilities; the remaining tiles with two odd coordinates are filler tiles, all of the same type.

In the example p, q, r, s represent original tile types where the pairs (p, q) and (r, s) are in the horizontal compatibility relation and the pairs (p, r) and (q, s) in the vertical one.

For an alternative construction yielding this reduction see Grädel [11].

A third way of enforcing a compatibility relation is the use of bumps and dents like in a real life Jig-saw puzzle; in this format the compatible tiles have dual rather than equal geometrical structures along their shared edge. The resulting compatibility relations have the structural property $\mathcal{C}$. We will use this format in our short-cut reduction towards the Knapsack problem.

A tiling problem consists of a set of tile types X and a region V in the plane which has to be tiled using instances of tiles in X. Furthermore there can be specified some additional constraint like a given colouring along (part of) the edge of the region V which must be extended, or a single tile which already is placed somewhere, and which must be included in the resulting tiling.

Specific instances of tiling problems which we will consider in this paper are the following problems:

BOUNDED TILING
INPUT: A set X of tile types and a $n \times n$ square region V with a given colouring along the edge
QUESTION: Does there exist an X-tiling of V extending the colouring of the edge?

CORIDOR TILING
INPUT: A set X of tile types and a pair of segments *top* and *bot* of length n with given colourings
QUESTION: Does there exist a height m for which there exists an X-tiling of the $n \times m$-rectangle whose top and bottom edges are *top* and *bot* respectively, with white left and right sides? (white being a fixed colour in the set of colours)

ORIGIN CONSTRAINED TILING
INPUT: A set X of tile types and a single tile type $t_0 \in X$
QUESTION: Does there exist an X-tiling of the entire plane where t_0 is placed at the origin?

ORIGIN CONSTRAINED QUADRANT TILING
INPUT: A set X of tile types and a single tile type $t_0 \in X$
QUESTION: Does there exist an X-tiling of the positive quadrant in the plane where t_0 is placed at the origin?

UNCONSTRAINED TILING
INPUT: A set X of tile types
QUESTION: Does there exist an X-tiling of the entire plane?

RECURRING TILING
INPUT: A set X of tile types and a single tile type $t_0 \in X$
QUESTION: Does there exist an X-tiling of the entire plane where t_0 is used infinitely often?

The usefulness of tiling problems for obtaining reductions from computational problems results from a well-known translation between time-space diagrams of Turing machine computations of a machine M, and X-tilings for a set of tile types $X = X(M)$ which is entirely determined by M. The construction below originates from Robinson [28] but the ideas go back to Wang and Berger [33, 2]. The idea is to enforce that the colourings of adjacent horizontal lines in the tiled region encode successive configurations in a computation of M. Throughout this paper we let time run downwards. By enforcing that the region to be tiled has the initial configuration on input w as a colouring for the top boundary while at the bottom boundary an accepting configuration is encoded, one forces the entire tiling to encode an accepting computation.

In order to explain the construction of these tile types we use the following notation for a tile type: $< l, u, r, b >$ denotes the tile with left colour l, top colour u, right colour r and bottom colour b. Furthermore we denote the white colour by $\cdot$.

Let K and Σ denote the state and tape symbol alphabet of M. The horizontal colours used in our tiling are selected from the set $\Sigma \cup (K \times \Sigma)$ whereas the vertical colours are K and the white colour.

When moving from one configuration to the next one on the larger part of the Turing machine tape nothing happens; a tape symbol is preserved during the transition. This behaviour is encoded by tiles of the shape $< \cdot, s, \cdot, s >$; we have such a tile-type in $X(M)$ for every tape symbol $s \in \Sigma$ including the blank symbol $\Box$.

However, a tape cell which is presently not scanned by the head may be scanned by the head during the next configuration when the head moves to the corresponding tape square either from the left or the right neighbour. This behaviour is encoded by tiles of the shape $< \cdot, s, q, (q, s) >$ or $< q, s, \cdot, (q, s) >$. Note that having both tile-types in $X(M)$ creates a problem: the two tiles may be placed next to each other, indicating that in a position where in configuration c_1 two tape symbols are present, in the next configuration c_2 a pair of "phantom heads" has appeared. While this phenomenon may occur in our quantum-mechanical reality we can't let this happen in our encodings of Turing machine computations. The easiest way to prevent this is by normalizing our Turing machine program in such a way that every state obtains a direction: when transferring towards a next state q the head either moves right or left, but not in both directions. Stated otherwise: if the program P contains a tuple $(\_, \_, q', \_, R)$ it can't contain a tuple $(\_, \_, q', \_, L)$ at the same time (here the underscores represent arbitrary symbols of the proper type). The set $X(M)$ includes for every pair $(q, s) \in (k \times \Sigma)$ one of the two above tile-types (depending on the direction of the state q).

The third sort of tile-types in $X(M)$ represents the instructions in the program P. For example, the instruction (q, s, q', s', R) is encoded by the tile type $< \cdot, (q, s), q', s' >$, and the instruction $(q, s, q', s', 0)$ yields the tile-type $< \cdot, (q, s), \cdot, (q', s') >$. The halting instruction $(q, s, \perp, s', 0)$ is encoded by $< \cdot, (q, s), \cdot, s' >$. In this way we include for every instruction in P a corresponding tile-type in $X(M)$. Note that due to the directionality of the states it will not happen that a colour q' occurs both on the right and on the left side of an instruction-tile. Consequently we can't have the situation that a pair of heads is annihilated leaving two plain tape symbols in the next configuration.

The various sort of tiles obtained in this way are illustrated in Figure 5. We have performed this translation for the case of the successor machine described in the previous section. A sample computation of this machine encoded as a tiling is illustrated in Figure 6.[1]

Based on this construction it is easy to obtain the following theorem:

Theorem 1 BOUNDED TILING *is complete for the class* **NP***[17, 18, 20]*

[1] A demonstration model of this tiling puzzle with 5 inch tiles in full colours is presently used for teaching purposes at the University of Amsterdam.

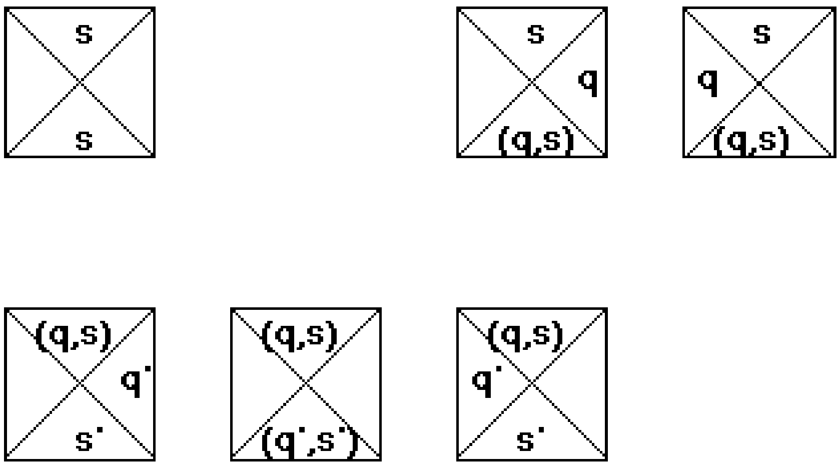

Figure 5: Tile types obtained by reducing a Turing Machine program

CORIDOR TILING *is complete for the class* **PSPACE**

ORIGIN CONSTRAINED TILING *and* **ORIGIN CONSTRAINED QUADRANT TILING** *are both* Π^0_0*-complete*

RECURRING TILING *is* Σ^1_1*-complete[12, 14]*

It is not difficult to show that each of the above problems belongs to the class for which it is claimed to be complete, so we only have to show why the problems are hard for these classes. The hardness results are obtained by a direct master reduction from Turing machine computations to tiling problems.

Evidently, a bounded tiling encodes a computation which is both time and space bounded; hence the completeness for the class **NP**. A corridor tiling encodes a space-bounded computation which explains the **PSPACE**-completeness of this problem.

By a suitable modification of the set $X(M)$ one can enforce that the unique way of extending a row containing a particular tile-type $t_0 \in X(M)$ yields the encoding of the initial configuration of machine M on a blank tape on the bottom edge (and some arbitrary neutral colour which easily can be extended over the top half of the plane on the top edge). If the computation of M on a blank tape terminates then the tiling can't be extended below the particular row encoding the halting configuration. So the solvable instances of the origin constrained problem under this translation correspond to diverging computations. This explains the Π^0_0-completeness of this problem.

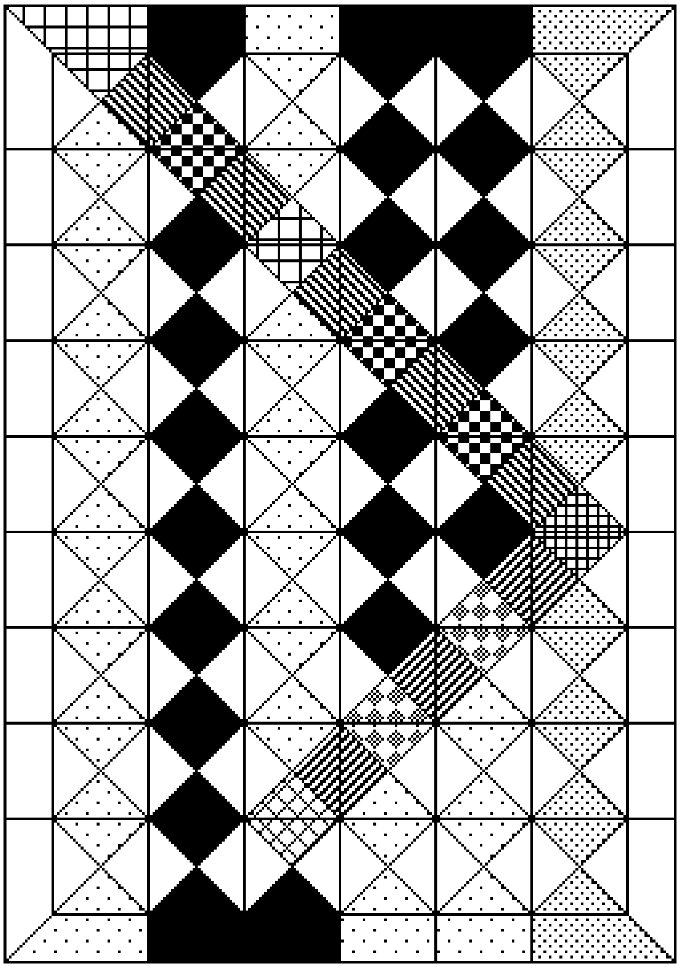

Figure 6: A tiling representing the computation $11 + 1 = 12$

Finally using a similar encoding one can show that recurrent tilings correspond to recurrent Turing machine computations (computations which return infinitely often to their initial state), explaining the Σ^1_1-completeness result.

For the bounded versions (BOUNDED TILING and CORIDOR TILING) completeness results can also be obtained without an explicit boundary condition. By a suitable extension of the set of colours one can enforce that every tile type can be used in a single column (or even a single row and column) only. In this version the question is whether a square of a given size (a rectangle of a given width) with white sides can be tiled. Use of this construction destroys however one of the nice features of the previous proofs: the fact that the tile set $X(M)$ is entirely determined by the machine M independently of the input w which on its turn determines the boundary condition.

A far more complicated proof is needed to show that the problem **UNCONSTRAINED TILING** is Π^0_0-complete. The reason is that the tile set $X(M)$ and its variations mentioned above typically include tiles which evidently by themselves allow a trivial tiling of the plane; the tiles representing the unchanged tape symbols are a good example. The result that **UNCONSTRAINED TILING** is Π^0_0-complete was originally proved by Berger and the proof was subsequently simplified by Robinson [2, 28]. The proof requires the construction of a set of tiles supporting a tiling of the plane but supporting no *periodic* tiling of the plane (a tiling which can be obtained from a tiling of a fixed rectangle with equal top and bottom sides and left and right sides respectively). Had it been the case that all tile sets supporting a tiling of the plane also would support a periodic tiling, then by Königs lemma, the set of solvable instances of **UNCONSTRAINED TILING** would have been both r.e. and co-r.e. and therefore recursive.

By constructing a sort of "tensor product" of a tile set enforcing aperiodic tilings of the plane and a set $X(M)$ one can enforce that the aperiodic tiling encodes longer and longer initial segments of a Turing machine computation. Again when the machine computation halts this encoding must get stuck somewhere. Conversely, when arbitrarily long initial segments of a computation for some machine M exist, then M must also support an infinite computation.

A final version of the master reduction takes us to the world of two person games. One way of looking at the Alternating Turing Machine model [4] is to consider a two person game where two players Alice and Elias, in turn select the next transition of the machine. It is the task of Elias to take the machine to an accepting state, whereas Alice wants to prevent this. If starting from the initial configuration Elias has a winning strategy which takes him to the accepting configuration against every strategy of Alice, the input is accepted by the alternating machine M; otherwise, when Alice can force the machine to reject, or when she can prolong the game forever, the input is rejected.

Evidently this alternating character is preserved when we consider two person tiling games where the two players tile an entire row of the region to be tiled in turns. But this is a rather artificial sort of game. It has been shown that also the version where players place during their turn only a single tile (while subject to a standard left to right and top to bottom order for their placements) results in the intended completeness results. These results were obtained by Chlebus [3].

Formally a two person tiling game consists of a tile set X and a region V to be tiled with some possible additional constraints as previously. But now also there is given a fixed order on the cells in V. The game is played by letting each player place a tile in X on the next free position in V subject to compatibility with tiles placed previously and the boundary constraint when applicable. Elias is the first player to move. Any player unable to perform a next move before the set V is tiled looses the game. When the set V is fully determined at the start of the game, as is the case with bounded tiling, Elias wins by completing the tiling. Otherwise when the region to be tiled is of some particular shape, as is the case with corridor tiling, Elias wins by completing a tiling of the required shape. Alice wins by preventing Elias from attaining his goal (possibly by letting the game go on forever).

The main ingredient of the proof is a modification of the construction of the set $X(M)$ which prevents any player from deviating from the protocol of encoding a proper Turing machine computation. For example, by introducing a state accepting tile $< \cdot, s, q, (q, s) >$ in a position where it shouldn't be any player can ruin the game. The problem is solved by turning such a move into an immediate loss of the game. On the other hand Elias should not be able to prevent Alice from moving the head one square to the left by refusing to play such a state accepting tile. The problem is solved by dividing the moves into two stages: in the first stage a player announces his intent to move, while in the second stage the move is performed with assistance of the other player; failure to provide this support yields an immediate loss. For details I refer to Chlebus' paper.

As a result we obtain two more completeness results for tiling games obtained by a master reduction:

Theorem 2 **TWO PERSON BOUNDED TILING** *is* **PSPACE**-*complete[3]*

TWO PERSON CORIDOR TILING *is* **EXPTIME**-*complete[3]*

These results follow from the well known results on alternating complexity classes by use of Chlebus' construction.

An alternative way of encoding Alternating Turing Machine computations by tiling games is given by Grädel [11]. His tilings are described in terms of adjacencies rather than by matching colours conditions. Players place during their turn a series of tiles until they are forced to give the turn to their opponent. In this construction there exists a one-one correspondence between turns and alternations of the encoded Turing Machine computation. This yields complete tiling games at bounded levels of the Polynomial Time Hierarchy. Grädel has also considered bounded versions of tiling connectability problems. One of his constructions provides for a complete game for feasible complexity classes within **P**. Furthermore he shows that the one-dimensional versions of the tiling game are **P**-complete.

Our master reduction to BOUNDED TILING has yet another interesting property. When using an encoding consisting of a part representing the tile-type set X, followed by a part encoding the four sides of the boundary, one observes that the length of the first part is fully determined by the Turing Machine M. For a specific problem in **NP** this machine is fixed so this part of the BOUNDED TILING instance is fixed as well. The size of the second part is linear in the size of the square to be tiled, which in turn is proportional to the time of the computation. Hence the BOUNDED TILING instance produced by the master reduction has a size which is proportional to the running time of the accepting nondeterministic Turing machine. For most master reductions in the literature a quadratic or cubic blow-up of the instance size is required. Note that a quadratic blow-up results as soon as the encoding of the BOUNDED TILING instance requires us to list all cells in the square individually.

4 A shortcut to the Knapsack problem

The textbook proofs for the NP-completeness of the Knapsack problem (see for example the Garey & Johnson textbook [10]) introduce a sequence of intermediate problems and reductions. Each of these problems has its nice properties and uses, but for someone just interested in proving NP-completeness of the Knapsack problem to students in some Cryptography course such detours better be eliminated, if feasible

For reference I first define the problems I will use in this section. For other problems mentioned I refer to the standard textbooks.

EXACT COVER
INPUT: A finite set V of ponts and a family F of subsets of V
QUESTION: Does there exist a covering of V consisting of mutually disjoint members of F?

KNAPSACK
INPUT: A sequence of nonnegative integers $(a_1, ..., a_k)$ and a target value G
QUESTION: Does the linear equation $X_1 a_1 + \ldots + X_k a_k = G$ have a solution with $X_i = 0, 1$? Stated alternatively, can M be written as a sum of a subsequence of elements of $(a_1, ..., a_k)$ where each a_i is used at most once?

The standard proof first regularizes **SATISFIABILITY**, in such a way that formulas are required to be in CNF with at most three literals per clause (**3-SAT**). Next **3-SAT** is reduced to the problem called **3-DIM MATCHING** where one requires to cover the union of three disjoint sets X, Y and Z of k points each using k disjoint triangles from a given family F where each member of F contains a member in each of the sets X, Y and Z. This reduction requires a substantial amount of component engineering [10].

3-DIM MATCHING is easily seen to be a special case of the more general problem **EXACT COVER** where one just requires to find a disjoint cover of a set V consisting of subsets in some given family F, without further restrictions on cardinalities of the members of F or the size of the covering. A final bit-string encoding already described in Karp's original paper [22] completes the chain of reductions to the Knapsack problem.

We have already seen that the master reduction to **SATISFIABILITY** obtained by encoding state-time transition diagrams yields a reasonably short propositional formula in CNF. This construction can be made even more transparent by reducing from **BOUNDED TILING**: all one needs to assign a new meaning to the variables $q(i, j, l)$: on row i, column j in the tiling a tile of type l is placed. Here $1 \leq i, j \leq n$ and $1 \leq j \leq t$ where t equals the number of tile-types.

The resulting propositional formula consists of four parts:
1) everywhere some tile is placed: n^2 clauses of size t each;
2) nowhere two tiles are placed: $O(n^2 \times t^2)$ clauses of size 2 each;

3) no incompatible pairs along horizontal or vertical edges: another $O(n^2 \times t^2)$ clauses of size 2 each;
4) the boundary condition; easily enforced by $O(n \times t)$ unit-clauses.

So the resulting formula is not only in CNF but, except for part 1), it is already in 3-CNF.

The real shortcut however is that we can present a direct reduction from **BOUNDED TILING** to **EXACT COVER**, which requires no unintuitive component engineering at all. All one needs is geometrical insight. The magic word is *digitizing*.

First we use the equivalence between the various forms of the tiling problem and proceed to the true Jig-saw world of bumps and dents; by aligning these bumps and dents across the horizontal and vertical edges of the tile-types we still can ensure that each tile placement in a solution corresponds to a specific grid position (i, j).

The next step is to replace the tiles, which represent after all continuous subsets in the Euclidean plane by finite discrete sets of pixels, obtained by digitizing at a sufficiently fine scale to expose the differences between the various bumps and dents. Then the area to be covered by tiles becomes a finite set V consisting of all pixels in the image and each placement of a tile of type k at a proper grid position (i, j) becomes a finite subset of pixels $F(i, j, k) \in F$. We add to F a single set B expressing the boundary condition. Evidently there is a one-one correspondence between tilings of the original square and coverings by disjoint subsets in F. So, under this translation **BOUNDED TILING** has become a special case of **EXACT COVER**.

The above idea can be simplified further by being more specific about the structure of the bumps and dents used. For every colour which can occur on a horizontal (vertical) edge we need a matching pair of a bump and a dent, such that this bump (dent) won't match the ones corresponding to different colours. This can be realized by assigning to each horizontal (vertical) edge occurring as a gridline in or on the boundary of the square to be tiled a finite set of points $H_0 = \{p_1, \ldots, p_h\}$ ($V_0 = \{q_1, \ldots, q_v\}$) corresponding to the the h (v) colours occurring along a horizontal (vertical) edge. For the dent corresponding to colour i we take the singleton set $H_i^+ = \{p_i\}$ ($V_i^+ = \{q_i\}$) and for the bump we take the complement set $H_i^- = \{p_1, \ldots, p_h\} \setminus \{p_i\}$ ($V_i^- = \{q_1, \ldots, q_v\} \setminus \{q_i\}$). Our digitized tiles will have bumps on their top and left edges and dents on their bottom and right edges.

Note that in this encoding each properly placed tile covers all pixels in the square cell corresponding to its position not located on a gridline. These pixels therefore are non-informative and may be removed from our construction. This has the effect that the size of the set V can be reduced by a factor $h \times v/(h + v)$.

The resulting translation, together with the earlier one is illustrated in Figure 7. The structure of the reduction should now be evident; however for future reference we need the precise description of the sets $F(i, j, k)$ and B.

First consider the set V. If we just draw the edges of the grid-cells of a square of size $n \times n$ we observe that there occur $n+1$ vertical edges along the top of the square and only n horizontal edges. For reasons of regularity it is therefore convenient to add "spurious" horizontal edges on the right side of the square; similarly vertical edges will be added at the bottom; see figure 8.

The result is that we effectively have added new cells along the right and bottom side of the square, including a single cell having its left-upper corner at the right lower corner of the square. Evidently the added points will not be covered by any of the sets $F(i, j, k)$,

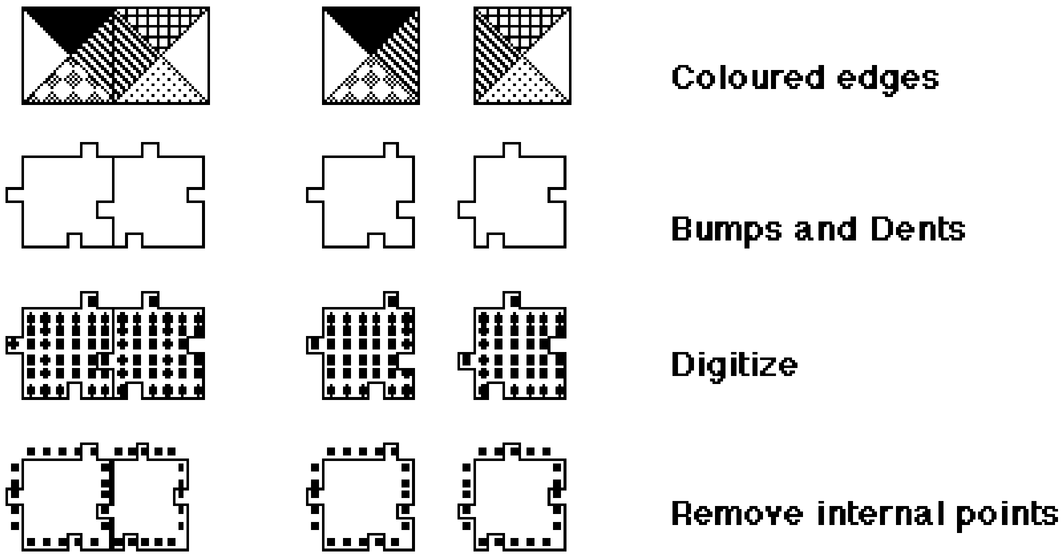

Figure 7: Reducing tilings to EXACT COVER

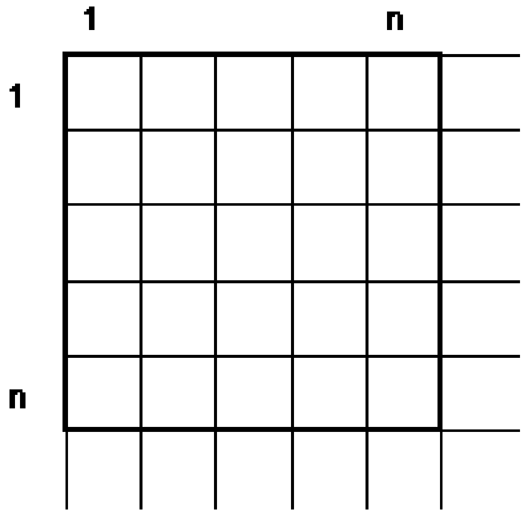

Figure 8: Extending the tiling region by spurious edges

but we will cover them by the boundary set B. After this extension the set V consists of $(n+1) \times (n+1) \times (h+v)$ points.

We introduce the following constants: $N = n+1, c = h+v, M = N \times c$ and $U = N \times M$, so U equals the size of V. We number the elements in V in a systematic way: cell after cell, row by row, from top to bottom, where each cell takes care of the points on its upper and left edge. The added cells cover the right and lower side of the original square and the spurious edges.

The key observation is that under this numbering an explicit description of the sets $F(i,j,k)$ becomes available. For a set of numbers $A = \{a_1, \dots, a_m\}$ and a number s we introduce the notation $A + s$ for the set $\{a_1 + s, \dots, a_m + s\}$. Define the for the k-th tile-type $< l, u, r, b >$, where u and b (l and r) are horizontal (vertical) colours the tile-set

$$T(k) = H_u^+ \cup V_l^+ \cup (H_b^- + M) \cup (V_r^- + c)$$

Then we have the following nice expressions for the sets $F(i,j,k)$

$$F(i,j,k) = T(k) + (i \times c + j \times M)$$

So each tile placement set $F(i,j,k)$ results from shifting the tile set $F(k)$ depending on the tile type only over a distance which is fully determined by the position! This observation will be key to the Hilbert 10 reduction in the next section.

We still need to describe the boundary set B. This set is determined by the given colouring on the edges of the original square. Let $U_i^+, L_j^+, U_i^-, L_j^-$ denote the colour at the i-th (j-th) edge of the top, left, bottom and right border of the square respectively, we obtain the rather unpleasant expression:

$$\begin{aligned}
B \;=\; & \bigcup_{i=0}^{n-1} (H^{-}_{U_i^{+}} + c \times i) \cup \bigcup_{j=0}^{n-1} (V^{-}_{L_j^{+}} + M \times j) \cup \\
& \bigcup_{i=0}^{n-1} (H^{+}_{U_i^{-}} + M \times n + c \times i) \cup \bigcup_{j=0}^{n-1} (V^{+}_{L_j^{-}} + c \times n + c \times M \times j) \cup \\
& \bigcup_{i=0}^{n-1} (V_0 + M \times n + c \times i) \cup \bigcup_{j=0}^{n-1} (H_0 + c \times n + M \times j)
\end{aligned}$$

The six terms cover the contributions of the upper, left, lower and right side of the square followed by the spurious edges at the lower and right side of the square.

Remark that in the tilings problems resulting from encoding a Turing machine computation the colours along the edges of the square will be a constant (white or the colour corresponding to the blank tape-symbol) except for the section on the top encoding the given input. Our encoding of an accepting configuration consists of a tape filled by the blank symbol only. This has the effect that the H-sets and the V-sets in the above expression become constant as well which shows that up to a disturbance proportional to the size of the input B consists of the union of six arithmetic sequences. This observation will also be needed in the next section.

The final step in the reduction chain towards the Knapsack problem is unchanged from Karp's construction. I will present it since we need its details for the sequel of this paper.

Assume that the set $V = \{0, 1, \ldots, v-1\}$ and let F denote the family of subsets of V out of which an exact cover has to be found. We associate to V the target value $G = \sum_{j=0}^{v-1} r^j$ and to each member $f_i \in F$ the knapsack item $a_i = \sum_{j \in f_i} r^j$. So both G and the a_i are represented in radix r by digit strings consisting of 0's and 1's.

Evidently an exact cover of V by members of F yields under this translation yields a solution of the resulting Knapsack equation. Conversely, each such solution originates from an exact cover provided the radix r is selected so large that the addition proceeds without generating any carry. Any choice of r larger than the number of elements in V will work. The latter bound can be sharpened to any number r such that an element of V does occur as member of at most $r-1$ members of F.

This latter bound is appropriate for the instances produced by the **EXACT COVER** instances generated by the reduction from **BOUNDED TILING**: each edge in the grid is adjacent to precisely two cells unless it is an edge along the border of the square in which case its points occur within the tile placement sets of one grid-cell and (possibly) in the boundary set. Hence $r > 2t$ suffices (where t denotes the number of tile types). So in our chain of reductions the lower bound on the radix is determined by the Turing Machine program and independent of the length of the input.

It will be convenient to take for r some power of 2 satisfying $2t < r \leq 4t$.

Observe that the nice structure of the sets $F(i, j, k)$ is preserved: the relation

$$F(i, j, k) = T(k) + (i \times c + j \times M)$$

translates into

$$f_{i,j,k} = t_k \times r^{i\times c+j\times M}$$

where $t_k = \sum_{j\in T(k)} r^j$. So the corresponding Knapsack items have values which factorize as a product consisting of a factor depending on the tile-type only and another factor completely determined by the position of the tile.

A similar remark can be made concerning the various sets which are united into the set B representing the boundary. Each contribution to this set consists of a set possibly determined by an edge colour, shifted over a distance depending on the position only. This translates into a product.

The explicit expression is given by:

$$\begin{aligned} b \ = \ & \sum_{i=0}^{n-1}(h^-_{U_i^+} \times r^{c\times i}) + \sum_{j=0}^{n-1}(v^-_{L_j^+} \times r^{M\times j}) + \\ & \sum_{i=0}^{n-1}(h^+_{U_i^-} \times r^{M\times n+c\times i}) + \sum_{j=0}^{n-1}(h^+_{L_j^-} \times r^{c\times n+M\times j}) + \\ & \sum_{i=0}^{n-1}(v_0 \times r^{M\times n+c\times i}) + \sum_{j=0}^{n-1}(h_0 \times r^{c\times n+M\times j}) \end{aligned}$$

In this expression we use the notation $h^-_{U_i^+}$ for the expression $\sum_{l\in U_i^+} r^l$ and similar for the five other sets consisting of edge-colours.

In case the edge colours are constant along the sides of the border this colour-factor may be factored out with the result that the knapsack weight corresponding to a side of the tiled region becomes a product of a constant factor multiplied by the sum of a geometric series depending on the height and width of the region only.

These observations will be crucial in the Hilbert 10 reduction in the next section.

I conclude by presenting the final Knapsack equation which results from this reduction:

$$\sum_{i=0,j=0}^{n-1} \sum_{k=1}^{t} X_{i,j,k} f_{i,j,k} + b = G$$

Note however that due to the combinatorics behind the construction it is evident that any solution of this equation will also satisfy the constraint which expresses that at every cell precisely one tile will be placed: $\sum_{k=1}^{t} X_{i,j,k} = 1$ for every pair $0 \leq i,j < n$.

5 A Hilbert 10 reduction

Matijasevič' solution of the Hilbert 10 problem completed in 1970 a line of research which had been going on for several decades. The result that there is no decision procedure for solvability Diophantine equations, basically consists of two steps. First it is shown by a reduction from the Halting problem that solvability of equations in the substantial larger language of *Exponential Diophantine equations* is undecidable. This result is known since the early sixties (results by Davis, Putnam and Robinson [8]). Subsequentially it is shown that the remaining exponential relations can be expressed in the more restricted language of Diophantine equations, as shown by Matijasevič in 1970. For a historical survey I refer to [7].

In this section I present a much simplified proof for the Davis, Putnam and Robinson part of the proof. Where the original reduction from machine computations to Exponential Diophantine Equations was based on the encoding potential of Gödels Beta function it was realized during the mid 70-ies that a far more powerful technique was obtained using the so-called *Dominance* relation, which is expressible in Exponential Diophantine terms using classic results by Kummer and Lucas.

This technique has been used by Matijasevič and Jones for obtaining simplified proofs of the Davis, Putnam and Robinson result as well. When presenting these results at a workshop in Paderborn in October 1982 they raised the question whether their methods which were invoked on Register Machine computations could be used for the Turing Machine model as well. The proof presented below answers this question affirmatively. It was obtained during the workshop and published in its proceedings. Other than this rather inaccessible private publication of the University at Paderborn [31] the proof has remained unpublished.

Rather than presenting yet another historical survey I will attempt to explain the problem and the our solution.

Hilbert asks in his tenth problem for a general method for determining whether a given Diophantine equation is solvable or not. Such an equation can be represented by a Polynomial $Q(X_1, \ldots, X_k)$ with integral coefficients in k integral variables. Alternatively one could require the solution to be defined over the non-negative integers, but it is easy to see that the two versions of the problem are interreducible: since every non-negative integer x_0 can be written as the sum of four squares $x_0 = x_1^2 + x_2^2 + x_3^2 + x_4^2$ natural variables can be replaced by integral variables at the price of multiplying the number of variables by a factor 4 and doubling the degree of the polynomial. Conversely every integer can be written as the difference of two non-negative integers, and therefore a converse reduction exists which only doubles the number of variables, leaving the degree of the polynomial invariant.

An alternative way of looking at the problem is obtained if we realize that solvability of the equation $Q(X_1, \ldots, X_k) = 0$ amounts to truth of the existentially quantified sentence: $\exists X_1, \ldots, \exists X_k[Q(X_1, \ldots, X_k) = 0]$. This illustrates the analogy between the Hilbert 10 problem and the problem of expressibility of mathematical properties in first order arithmetic.

In fact we are seldomly interested in just a single Diophantine equation but rather in solvability of equations depending on parameters. So actually we investigate the class of predicates over the integers definable as

$$p(A_1, \ldots, A_m) \Longleftrightarrow \exists X_1, \ldots, \exists X_k[Q(A_1, \ldots, A_m, X_1, \ldots, X_k) = 0]$$

The real question is how large is this class of so-called *Diophantine predicates*? For example when the Halting problem resides among these predicates (assuming that the translation is sufficiently effective) undecidability of Hilberts problem is a direct consequence.

Matijasevič' result shows that this class indeed is as large as it can be: it coincides with the full class of recursively enumerable predicates (that it can't be larger is evident from the definition - a Turing Machine by brute force can find a solution for a given equation if there exists one ...).

Also the result provides for various nice representation schemes for the RE sets. Let W_i denote the i-th RE set in a standard enumeration. Diophantine representability is expressed by the existence of polynomials Q_i satisfying

$$x \in W_i \iff \exists X_1, \ldots, X_k[Q_i(x, X_1, \ldots, X_k) = 0]$$

However the reduction turns out to be uniform: there exists a single polynomial Q satisfying:

$$x \in W_i \iff \exists X_1, \ldots, X_k[Q(i, x, X_1, \ldots, X_k) = 0]$$

Furthermore the RE sets can also be represented by the range of a polynomial function R rather than its domain of solvability as an equation:

$$x \in W_i \iff \exists X_1, \ldots, X_k[x = R(i, X_1, \ldots, X_k) \wedge x \geq 0]$$

Given the above results mathematicians have started to look for the simplest polynomials satisfying these conditions. Two evident measures are the number of variables and the degree. Jones [15] lists a number of best results showing a definite trade-off between variables and the degree. It is not hard to see that the degree can be reduced to 4 but then he needs 58 variables. The best known bound on the number of variables equals 9 but then the degree is of order 1.6×10^{45}!

Adleman and Manders [1] have investigated the role of the size of the solution for solvable equations: this has resulted into a theory of Diophantine Complexity.

I first present a sample of predicates which are easily seen to be Diophantine; first I state the property on the Y-s and next the Polynomial equation which should be solvable in the X-es.

Y is nonnegative	$Y = X_1^2 + X_2^2 + X_3^2 + X_4^2$
Y is positive	$Y = 1 + X_1^2 + X_2^2 + X_3^2 + X_4^2$
$Y_1 < Y_2$	$Y_2 = Y_1 + 1 + X_1^2 + X_2^2 + X_3^2 + X_4^2$
Y is a square	$Y = X^2$
Y is nonprime	$Y = (X_1 + 2) \times (X_2 + 2) \wedge X_1, X_2$ positive
$Y_1 \mid Y_2$	$Y_2 = Y_1 \times X$
$Y_1 \equiv Y_2 \bmod Y_3$)	$Y_1 = X \times Y_3 + Y_2$ for nonnegative $Y_2 \wedge Y_3 - (Y_2 + 1)$

In fact some of the definitions involve not a single equation but a finite set which should be simultaneously satisfied by the solution. However, it is not difficult to see that the collection of Diophantine predicates is closed under finite union and intersection. This follows from the two relations:

$$P = 0 \wedge Q = 0 \iff P^2 + Q^2 = 0$$

$$P = 0 \vee Q = 0 \iff P \times Q = 0$$

Also the set of Diophantine predicates is by definition closed under existential quantification: promote a parameter to an unknown, while preserving the equation. So life would be easy provided we could find a simple method to close the Diophantine predicates under the logical operations of negation and Universal quantification.

The key ingredient which seems to be missing from Diophantine language is a simple method of talking about combinatorial structures like sets and sequences. In the original proofs (finite) sequences of numbers are encoded in a single number as residues mod. an arithmetic sequence of divisors (Gödels Beta function). There exists an alternative method based on the following *Dominance* relation:

Definition 1 $X_1 \sqsubseteq X_2 \iff$ *in the binary representations of* X_1 *and* X_2 *a digit 1 occurs for* X_1 *only at locations where a digit 1 occurs for* X_2

So for example, given the binary representations $3 = 011$, $5 = 101$ and $7 = 111$ one sees that $3 \sqsubseteq 7$ but not $3 \sqsubseteq 5$.

Kummer [24] observed in 1852 that $X_1 \sqsubseteq X_2$ iff $\left(\begin{array}{c} X_1 \\ X_2 \end{array}\right)$ is odd; this is a special case of a more general theorem due to E. Lucas (1878) [21]:

Theorem 3 (Lucas) *Let* p *be a prime number and let the numbers* X *and* Y *in radix* p *be represented by the digit sequences:* $X = X_n, \ldots, X_0$ *and* $Y = Y_n, \ldots, Y_0$. *Then*

$$\left(\begin{array}{c} X \\ Y \end{array}\right) \equiv \prod_{j=0}^{n} \left(\begin{array}{c} X_j \\ Y_j \end{array}\right) \bmod p$$

.

This result is proven by considering the polynomial $(T+1)^X \bmod p$. By binomial expansion we have:

$$(T+1)^X = \sum_{y=0}^{X} \left(\begin{array}{c} X \\ y \end{array}\right) T^y$$

Using the radix p expansion of $X = X_n p^n + \ldots + X_0$ and $Y = Y_n p^n + \ldots + Y_0$ and using the fact that $(A+B)^p \equiv A^p + B^p \bmod p$ we obtain:

$$\begin{aligned}
(T+1)^X &= (T+1)^{X_n p^n + \ldots + X_0} \\
&= \prod_{j=0}^{n} (T+1)^{p^j \times X_j} \equiv (mod.p) \\
&= \prod_{j=0}^{n} (T^{p^j} + 1)^{X_j} \\
&= \prod_{j=0}^{n} \sum_{y_j=0}^{X_j} \left(\begin{array}{c} X_j \\ y_j \end{array}\right) T^{y_j p^j} \\
&= \sum_{y_n=0,\ldots,y_0=0}^{X_n,\ldots,X_0} \prod_{j=0}^{n} \left(\begin{array}{c} X_j \\ y_j \end{array}\right) T^{\sum_{j=0}^{n} y_j p^j}
\end{aligned}$$

The properties of radix-p expansion entail that each choice of values y_j leads to a different exponent in the last expression. Subsequently comparing coefficients in the two expressions for $(T+1)^X$ yields the result.

The relation below shows how binomial coefficients in their turn are defined in terms of exponentials:

$$Y_3 = \left(\begin{array}{c} Y_1 \\ Y_2 \end{array} \right) \Longleftrightarrow 0 \leq Y_3 < X_1 \wedge 0 \leq X_2 < X_1^{Y_2} \wedge (X_1+1)^{Y_1} = (X_3 \times X_1 + Y_3) \times X_1^{Y_2} + X_2$$

In the above expression one may choose for X_1 a suitably large chosen power of 2. The correctness of this description can be seen from inspection of the decimal representations of small powers of nice decimal numbers like 100...01 .

We now have collected all the ingredients we need for describing our reduction from Halting problem to the problem of deciding solvability of Exponential Diophantine Equations.

Without loss of generality we consider instances of the Halting problem of the following simple form: a Turing machine M starts on a blank semi-infinite tape; does the computation halt or not?

Using the techniques from Section 3 we can construct a collection T of tile types with the following properties: if M halts on blank input there exists a number S such that it is possible to tile the square of size $S \times S$ given the following boundary colouring: except for the leftmost cell along the upper boundary which has a colour c_0 corresponding to $< q_0, \Box >$ all edge colours are white. In case the machine diverges, regardless the size of S, no such tiling of an $S \times S$ square of this form is possible.

As previously, the structure of the set of tile types T depends on the program of M only. Also in this scenario the boundary constraint is fixed (since there is no input). There is however an important difference compared to the reduction towards **BOUNDED TILING**: the size of the square to be tiled S now is part of the solution rather than of the specification of the problem instance!

This dependence becomes evident when we consider the effects of the reduction in Section 4. We select for the radix r some power of 2 satisfying $2t < r = 2^p \leq 4t$ where t denotes the number of tile types. For tile $T(k)$ we have the corresponding basic set:

$$T(k) = H_u^+ \cup V_l^+ \cup (H_b^- + M) \cup (V_r^- + c)$$

Now we have $M = (S+1) \times c$ so this set becomes dependent on the size of the output. Remember that c is completely determined by the program of M only.

The set corresponding to placing $T(k)$ at position (i,j) in the square has the following form:

$$F(i,j,k) = T(k) + (i \times c + j \times M)$$

So this set exhibits one more dependence on S.

Finally we reconsider the set corresponding with the boundary constraint. Let wh (wv) denote the code for the white colour along the horizontal (vertical) edge. The set B' represents the boundary constraint for a square which is entirely white.

$$
\begin{aligned}
B' \quad = \quad & \bigcup_{i=0}^{n-1} (H^-_{wh} + c \times i) \cup \bigcup_{j=0}^{n-1} (V^-_{wv} + M \times j) \cup \\
& \bigcup_{i=0}^{n-1} (H^+_{wh} + M \times n + c \times i) \cup \bigcup_{j=0}^{n-1} (V^+_{wv} + c \times n + c \times M \times j) \cup \\
& \bigcup_{i=0}^{n-1} (V_0 + M \times n + c \times i) \cup \bigcup_{j=0}^{n-1} (H_0 + c \times n + M \times j)
\end{aligned}
$$

In order to accommodate for the single non-white colour c_0 along the edge one has to update this set by:

$$B = B' \setminus H^-_{wh} \cup H^-_{c_0}$$

In this expression for B two quantities have become dependent on the size of the square: as before $M = (S+1) \times c$ and moreover $n = S$.

Next we inspect the subsequent reduction to a Knapsack equation. The target value for the Knapsack instance G is given by:

$$G = \sum_{l=0}^{v-1} r^j$$

where r is the radix and $v = (n+1)^2 \times c$. So also G now depends on the size of the square since $n = S$. Since G is the sum of a geometric series we obtain: $G = (r^{(S+1)^2 \times c} - 1)/(r-1)$. This shows that we have also a simple exponential expression for G.

The Knapsack items were given by:

$$f_{i,j,k} = t_k \times r^{i \times c + j \times M}$$

The size S affects the values t_k by means of the contribution of the bottom edge of the cell to be tiled: this effect leads to an expression of the form $t_k = t_{k,0} + r^{(S+1)c} t_{k,1}$ with constants $t_{k,0}$ and $t_{k,1}$ which depend on the program only. Furthermore the exponent of r above depends on S. Note that both dependencies are expressed by means of simple exponentials.

For the boundary item b we obtain the expressions:

$$
\begin{aligned}
b' \quad = \quad & \sum_{i=0}^{S-1} (h^-_{wh} \times r^{c \times i}) + \sum_{j=0}^{S-1} (v^-_{wv} \times r^{M \times j}) + \\
& \sum_{i=0}^{S-1} (h^+_{wh} \times r^{M \times S + c \times i}) + \sum_{j=0}^{S-1} (h^+_{wv} \times r^{c \times n + M \times j}) + \\
& \sum_{i=0}^{S-1} (v_0 \times r^{M \times S + c \times i}) + \sum_{j=0}^{S-1} (h_0 \times r^{c \times S + M \times j})
\end{aligned}
$$

Since b' expresses the boundary constraint for the entirely white square we must adjust to obtain the correct value as $b = b' - h^-_{wh} + h^-_{c_0}$. Since each of the six summands in b' represents a simple geometric series the expression can be rewritten to:

$$\begin{array}{rcl} b & = & (h^-_{wh} + (h^+_{wh} + v_0) \times r^{M\times S}) \times (r^{c\times S} - 1)/(r^c - 1) \\ & + & (v^-_{wv} + (v^+ - wv + h_0) \times r^{c\times S}) \times (r^{M\times(S+1)} - 1)/(r^M - 1) \\ & + & (h^-_{c_0} - h^-_{wh}) \end{array}$$

So also the quantity b can be expressed as simple exponentials.

Finally there is the Knapsack equation itself. At first it seems that our task is hopeless since the number of unknowns $X_{i,j,k}$ has become dependent on S. However, rearranging the order of summation we obtain the expression:

$$\sum_{k=1}^{t} \sum_{i=0,j=0}^{S-1} X_{i,j,k} f_{i,j,k} + b = G$$

Subsequently we substitute for the $f_{i,j,k}$ the expressions obtained above yielding:

$$\sum_{k=1}^{t} \sum_{i=0,j=0}^{S-1} X_{i,j,k} \times t_k \times r^{i\times c+j\times M} + b = G$$

which can be rearranged into:

$$\sum_{k=1}^{t} (\sum_{i=0,j=0}^{S-1} X_{i,j,k} r^{i\times c+j\times M}) \times t_k + b = G$$

Let $Z_k = \sum_{i=0,j=0}^{S-1} X_{i,j,k} r^{i\times c+j\times M}$. Then this equation reduces to the linear equation

$$\sum_{k=1}^{t} Z_k \times t_k + b = G$$

This no longer represents a Knapsack equation since the unknowns Z_k are not $0-1$ valued. They represent however numbers of a rather special form: keeping in mind that $r = 2^p$ represents a suitable power of 2 and the fact that the $X_{i,j,k}$ are $0-1$ valued we see that the Z_k are numbers which have, if written in binary, digits 1 only at multiples of $p \times c$.

In the previous section we observed that the solutions to the Knapsack equation obtained by the reduction must also satisfy the condition: $\sum_{k=1}^{t} X_{i,j,k} = 1$ for every pair $0 \leq i, j < S$; this condition translates into yet another equation on the Z_k:

$$Z = \sum_{k=1}^{t} Z_k = \sum_{k=1}^{t} \sum_{i=0,j=0}^{S-1} X_{i,j,k} r^{i\times c+j\times M} =$$

$$\sum_{i=0,j=0}^{S-1} (\sum_{k=1}^{t} X_{i,j,k}) r^{i\times c+j\times M} = \sum_{i=0,j=0}^{S-1} r^{i\times c+j\times M} =$$

$$(r^{S\times c} - 1)/(r^c - 1) \times (r^{S\times M} - 1)/(r^M - 1)$$

So this quantity Z as well is a well structured number, if expressed in binary; it has digits 1 exactly at all multiples of $p \times c$ except for the positions corresponding to the added cells at the right boundary of the square which are of the form $p \times c \times (S + j(S+1))$.

It is at this point that we invoke the dominance predicate of Kummer: the Z_k obey the following conditions:

$$Z_1 \sqsubseteq Z, \ldots, Z_t \sqsubseteq Z, \quad \sum_{k=1}^{t} Z_k = Z$$

and these conditions combined with the linear equation $\sum_{k=1}^{t} Z_k \times t_k + b = G$, express everything we need.

Gathering what we have seen so far, we can present the final reduction.

Theorem 4 *Given a Turing Machine M, one can effectively obtain a collection of constants (dependent on M only) given by:*

$$t, r = 2^p, c, t_{k,0}, t_{k,1} \quad (k = 1, \ldots, t), h_{wh}^-, h_{wh}^+, v_{wv}^-, v_{wv}+, v_0, h_0, h_{c_0}^-$$

with the following properties: Machine M halts if started on empty input if and only if the system of exponential Diophantine equations given below has a solution in

$$S, M, G, t_1, \ldots, t_k, b, Z, Z_1, \ldots, Z_t$$

$$\begin{array}{rcl}
t_k & = & t_{k,0} + r^{(S+1)c} t_{k,1} \quad (k = 1, \ldots, t) \\
M & = & c \times (S+1) \\
G & = & (r^{(S+1)^2 \times c} - 1)/(r-1) \\
b & = & (h_{wh}^- + (h_{wh}^+ + v_0) \times r^{M \times S}) \times (r^{c \times S} - 1)/(r^c - 1) \\
 & + & (v_{wv}^- + (v^+ - wv + h_0) \times r^{c \times S}) \times (r^{M \times (S+1)} - 1)/(r^M - 1) \\
 & + & (h_{c_0}^- - h_{wh}^-) \\
Z & = & (r^{S \times c} - 1)/(r^c - 1) \times (r^{S \times M} - 1)/(r^M - 1) \\
Z_k & \sqsubseteq & Z \quad (k = 1, \ldots, t) \\
Z & = & \displaystyle\sum_{k=1}^{t} Z_k \\
G & = & \displaystyle\sum_{k=1}^{t} Z_k \times t_k + b
\end{array}$$

Note that the real unknowns in the above equations are S and the $Z_k, k = 1, \ldots, t$; the remaining unknowns are (exponential) expressions in the above and the constants determined by M which, in principle could have been eliminated from the system.

6 A lower bound for PDL

As a final example of the use of tiling problems for reductions I will present an alternative proof for the deterministic exponential time lowerbound for satisfiability for Propositional Dynamic Logic [13]. This lowerbound originally was proven by Fischer and Ladner [9] by means of a master reduction from Polynomial Space bounded Alternating Turing Machines [4].

We have seen that the computational power of space bounded Turing Machines is captured by the Corridor Tiling problem. Also, by Chlebus' construction [3] the alternating space bounded Turing machines are conveniently simulated by the two person version of this tiling game. So our target will be reached by designing a reduction from the two person Corridor tiling game to Satisfiability in PDL, hoping that the resulting reduction will turn out to be more elegant than the original one.

PDL [13] is one of the foremost logics of programs. Invented by Pratt [27] in the more extended version known as First Order Dynamic Logic and further developed among others by Harel, it became the standard example of an application of intensional logic in computer science. The system has a sound and complete axiomatization by means of the so-called Segerberg axioms.

For details I refer to Harel's chapter on Dynamic Logic in the Handbook of Philosophical Logic cited above. For the purpose of this paper it suffices to give a short description of the language and its semantics.

The language consists of expressions of two sorts: Formulas and Programs. Propositional variables $Prop$ represent the basic building blocks for formulas and atomic program variables Act the building blocks for programs.

Formulas are closed under the traditional connectives from propositional logic: $\neg, \wedge, \vee, \Rightarrow$. Programs are closed under the regular operators: ; (sequential composition), $\cup$ (union) and $*$ (Kleene star). Finally there are two rules in the grammar connection the two sorts:

If α denotes a program and ϕ denotes a formula then $< \alpha > \phi$ (after α it is possible to attain ϕ) and $[\alpha]\phi$ (after α it is certain to attain ϕ) are formulas.

If ϕ denotes a formula then $\phi?$ (test for ϕ) denotes a program.

The standard semantics is based on Kripke structures, which amounts to interpreting PDL as a multi-modal logic with a highly structured set of modal operators.

A model consists of a set of worlds W (called states), and an interpretation for the propositional variables and program variables. $p \in Prop$ is interpreted by a subset $v(p) \subseteq W$, whereas $a \in Act$ is interpreted by a binary relation $I(a) \subseteq W \times W$.

The interpretation $v(\phi)$ of general propositions as subsets of W is standard: $\neg, \wedge$ and $\vee$ obtain their traditional meaning as complement, intersection and union on the subsets of W, and also $\Rightarrow$ is defined in the standard way as abbreviation: $\phi \Rightarrow \psi = \neg\phi \vee \psi$.

The interpretation $I(\alpha)$ as binary relations on W, capturing the conception of α as a nondeterministic multivalued partial state transition is obtained as follows: the connectives ; and $\cup$ are interpreted by the relation product and union respectively. α^* is interpreted by the transitive reflexive closure of the interpretation of α.

The semantics of the two operators connecting the two sorts are given by:

$(x, y) \in \phi?$ iff. $x = y \wedge x \in v(\phi)$.

$x \in < \alpha > \phi$ iff. $\exists y[(x, y) \in I(\alpha) \wedge y \in v(\phi)$

$x \in [\alpha]\phi$ iff. $\forall y[(x, y) \in I(\alpha) \wedge y \in v(\phi)$

The last two clauses show that the program modalities $< \alpha >$ and $[\alpha]$ have obtained the standard Kripke semantics for modal logic.

A model of a PDL formula ϕ is an interpretation together with a world x in its set of states W such that $x \in v(\phi)$. The formula is called satisfiable if it has a model.

It was shown that Satisfiability of PDL formulas is decidable by means of a Finite Model construction: if a formula has a model at all it has also a small model where the number of states is bound by an exponential function of the length of formula; moreover such a model can be constructed in deterministic exponential time, leading to a DEXP upper bound for this problem [23, 30]. Our purpose is to show that this upper bound is actually tight.

Our proof method consists of expressing the two person version of the Corridor Tiling game by means of a PDL formula in such a way that this formula has a model iff the first player in the tiling game has a winning strategy.

So we consider an instance of the two person Corridor Tiling Problem. Let X denote the set of tile types. Let $u \in X$ denote a particular tile type which will enforce the condition that the corridor sides are white. We ensure that the only tile which can be placed below u is u itself. The width of the corridor is denoted n. There is a given initial configuration along the upper end of the corridor, which starts and ends with a tile of type u. Elias and Alice in turn place a tile in the first free location (row by row from left to right) observing the tiling adjacency conditions. They both can use the same collection of tile types. The tiles of type u which represent the sides of the coridor however are not placed by Alice and Elias but by the referee of the game.

Whoever is unable to make a next move loses the game. The game is also won by Elias when he locates a tile of the special type $w \in X$ in the first column. Without loss of generality we assume that w only can be placed in the first column, and that, as soon as w is placed the game terminates (since there exists no tile which can be placed next to w). The remaining tile types in X are denoted $\{t_1, \ldots, t_s\}$ with $u = t_0, w = t_{s+1}$.

We must also consider the possibility of a game that doesn't terminate. This will require that the corridor will include repeated rows. Evidently it is never to the advantage of Elias to construct repeated rows, so we will consider such an infinite game a win for Alice.

We introduce propositional variables $p_{i,k}$ with the intended meaning: *the last tile placed in column i of the corridor is of type k*. Furthermore we introduce propositional variables q_j with the meaning *the last tile placement occurred in column j*. The propositional variable b encodes who is next to move: b is true in those positions where Elias places the next tile.

We introduce a single program variable a with the intended meaning *one move in the tiling game*.

We now describe how the structure of the tiling game in its progression can be expressed using PDL formula's. The formula constructed in the reduction is a conjunction of a collection of subformula's each expressing one of the required conditions.

Note that in the intended model of our PDL formula a state means a position in the game. The conditions which should hold before a move of Elias (Alice) are expressed using the Existential (Universal) modal operator $< a >$ $([a])$, followed by expressions which should hold after such a move. A model of the formula will represent a complete game tree where for every position in the tree where Alice (Elias) is next to move all (a single) successor positions will be required to exist.

Our first subformula expresses the initial position of the game:

$$b \wedge q_1 \wedge p_{0,u} \wedge p_{1,t_1} \wedge \ldots \wedge p_{n,t_n} \wedge p_{n+1,u}$$

That both players alternate their moves is expressed by:

$$[a^*]((b \Rightarrow [a]\neg b) \wedge (\neg b \Rightarrow [a]b)$$

The next two subformulas express that every move occurs in exactly one column in the corridor:

$$[a^*](q_1 \vee q_2 \vee \ldots \vee q_n)$$

$$[a^*](\neg q_i \vee \neg q_{i'}) \qquad (i \neq i')$$

That the tiles are placed in the correct left-to-right order is expressed by:

$$[a^*](q_1 \Rightarrow [a]q_2 \wedge \ldots \wedge q_n \Rightarrow [a]q_1)$$

That every column has exactly one tile which is last played into it is expressed by the following collection of subformulas:

$$[a^*](p_{i,u} \vee p_{i,1} \vee \ldots \vee p_{i,s} \vee p_{i,w}) \qquad (i = 0 \ldots n+1)$$

$$[a^*](\neg p_{i,j} \vee \neg p_{i,j'}) \qquad (i = 0 \ldots n+1, \;\; 0 \leq j \neq j' \leq s+1)$$

The next subformula expresses that in columns where no move is made nothing changes:

$$[a^*](\neg q_i) \Rightarrow ((p_{i,t} \Rightarrow [a]p_{i,t}) \wedge (\neg p_{i,t} \Rightarrow [a]\neg p_{i,t})) \qquad (i = 1, \ldots, n, \;\; t = 0 \ldots s+1)$$

Next we need to express that a tile placement obeys the adjacency conditions. Whether tile t_j can be placed in column i depends in general only on the tile recently placed to the left in column $i-1$ (possibly column 0) and the tile placed in column i. Only when $i = n$ the constraint that in column $n+1$ a tile of type u must be placed has to be enforced as well.

So let INC denote the collection of triplets of tile types $(t, t', t")$ such that it is illegal to place $t"$ below t and right from t' and let INC' denote the larger collection of such triplets describing illegal placements in column n. The the subformulas below express that all tile placements are legal:

$$[a^*]((q_i \wedge p_{i,t} \wedge p_{i-1,t'}) \Rightarrow [a]\neg p_{i,t''}) \qquad (i = 1 \ldots n-1, (t, t', t'') \in INC)$$

$$[a^*]((q_n \wedge p_{n,t} \wedge p_{n-1,t'}) \Rightarrow [a]\neg p_{n,t''}) \qquad ((t, t', t'') \in INC')$$

We also must express that all possible moves for Alice are being taken into consideration. This requirement for Alice is expressed by the subformulas:

$$[a^*]((\neg b \wedge q_i \wedge p_{i,t} \wedge p_{i-1,t'}) \Rightarrow \bigwedge_{(t,t',t'')\notin INC} <a> p_{i,t''}) \qquad (i = 1 \ldots n-1,\ \ 0 \leq t, t' \leq s+1)$$

$$[a^*]((\neg b \wedge q_n \wedge p_{n,t} \wedge p_{n-1,t'}) \Rightarrow \bigwedge_{(t,t',t'')\notin INC'} <a> p_{n,t''}) \qquad (0 \leq t, t' \leq s+1)$$

For Elias it suffices to express that he can move when it is his turn:

$$[a^*](b \Rightarrow <a> (b \vee \neg b))$$

It is easy to see that models of the conjunction of the above PDL formulas represent legal game trees, which include for every position where Elias (Alice) is to move one (all) possible successor configuration(s). So there only remains to express that the initial state (the starting position in the game) is a win for Elias.

Now evidently in order that Elias wins the game, all terminal positions (states without successor states in the game) should be positions where tile w is located in column 1, since this expresses the winning condition for Elias. This can be expressed by the subformula:

$$[a^*]([a](b \wedge \neg b) \Rightarrow p_{1,w})$$

However, this formula is insufficient to guarantee that the game in fact terminates. Since existence of an infinite branch in the game tree was interpreted as a win for Alice our PDL formula therefore is still incomplete.

If we compare our proof with the original one given by Fischer and Ladner we see that we obtained some simplifications due to the fact that the combinatorics of the Tiling game (even in its two person variant) are simple compared to Turing Machine configurations. However we are now facing a problem Fischer and Ladner didn't have to consider. It is evident that a Polynomial Space bounded Alternating Turing Machine may be assumed to terminate, but there is no reason to assume that the same holds for corridor tilings.

So we are facing the following choice. Either we leave our proof in its present shape , observing that it still will suffice eventually for establishing the required lowerbound for PDL satisfiability; if we consider the reduction chain from alternating Turing Machines via Tiling games to PDL as a whole we can invoke the same hidden termination assumption which is implicit in the Fischer Ladner proof. The alternative is extending our formula by another component which excludes the presence of infinite branches in the game tree.

The latter strategy is easily implemented by introducing a binary counter which counts the number of moves. Positions where the counter overflows being declared a win for Alice and so we must enforce that the counter in fact doesn't overflow. It is easy to see that the required number of bits for the counter is proportional to the width of the corridor: $N = cn$ bits suffice for a suitable value of c. The bits of the counter are represented by the propositional variables $q_0, \ldots, q_N$ where q_j is true if the bit equals 0.

The formula below expresses that the counter is properly initialized in the initial state:

$$q_0 \wedge q_1 \wedge \ldots \wedge q_N$$

The next set of subformulas express that the counter properly is incremented at every move.

$$\begin{array}{ll} [a^*] & ((\neg q_0 \wedge \ldots \wedge \neg q_{j-1} \wedge q_j) \\ \Rightarrow & ([a]((q_0 \wedge \ldots \wedge q_{j-1} \wedge \neg q_j) \\ \wedge & (\bigwedge_{i=j+1}^{N} (q_i \Rightarrow [a]q_i \wedge \neg q_i \wedge [a]\neg q_i) \;)) \qquad (j = 0 \ldots N) \end{array}$$

Incrementing a binary counter amounts to replacing the leftmost block of 1's by 0's, followed by substituting the 0 before this block by a 1; the remaining digits remain unaffected.

Counter overflow is detected when bit q_N becomes 1. So the termination of the game is enforced by the final subformula:

$$[a^*]q_N$$

7 Conclusions

By reproving a number of known results we have illustrated the convenience of the use of tiling games in stead of (Turing) Machine configurations as combinatorial structure for proving lower bounds in computation and complexity theory. The higher regularity and decreased interaction between successive constituents in the combinatorial structure leads to formula which are easier to write down and understand. The example of the Hilbert 10 reduction shows that even for this notorious scenario a relatively simple and well structured system of Exponential Diophantine equations can be obtained starting with the Turing Machine model of computation.

The reverse side of the medal is that in order to reap these gains we frequently had to resort to infringements against the principle of orthogonality of reductions: we abuse structure present in the instances of a problem generated by a previous reduction when presenting the next reduction in line.

It remains to be seen whether in the future further examples of (master) reductions will be encountered where our strategy can be shown to offer advantages over the standard proofs in the literature.

References

[1] Adleman L. & Manders, K., *Diophantine complexity*, proc. IEEE FOCS 17 (1976) 81–88.

[2] Berger, R., *The undecidability of the Domino problem*, Mem. Amer. Math. Soc. 66 (1966).

[3] Chlebus, B.G., *Domino-tiling games*, J. Comput. Syst. Sci. 32 (1986) 374–392.

[4] Chandra, A.K, Kozen, D.C. & Stockmeyer, L., *Alternation*, J. Assoc. Comput. Mach. 28 (1981) 114–133.

[5] Cook, S.A., *The complexity of theorem-proving procedures*, Proc. ACM STOC 3 (1971), pp. 151-158.

[6] Davis, M., *Hilbert's tenth problem is unsolvable*, Amer. Math. Monthly 80 (1973) 233–269.

[7] Davis, M., Matijasevič, Y.V. & Robinson, J., *Hilbert's tenth problem. Diophantine equations: positive aspects of a negative solution*, in. Mathematical developments arising from Hilberts problems, Proc. Symposia in Pure Math. vol 28 part 2, AMS. (1976) pp. 323–378.

[8] Davis, M. Putnam, H. & Robinson, J., *The decision problem for exponential Diophantine equations*, Annals of Math. ser. 2 vol. 74 (1961) 425–436.

[9] Fischer, M.J. & Ladner, R.E., *Propositional dynamic logic of regular programs*, J. Comput. Syst. Sci. 18 (1979) 194–211.

[10] Garey, M.R. & Johnson, D.S., *Computability and Intractability; a guide to the theory of NP-completeness*, Freeman, 1979.

[11] Grädel, E., *Domino games and complexity*, SIAM J. of Comput. 19 (1990), 787-804.

[12] Harel, D., *Recurring dominoes: making the highly undecidable highly understandable*, in M. Karpinski (ed.), proc. FCT 1983, Springer LNCS 158 (1983), pp. 177–194.

[13] Harel, D., *Dynamic logic*, in D. Gabbay & F. Guenthner (eds.), Handbook of philosophical logic, vol. 2, Reidel (1984) pp. 498–604.

[14] Harel, D., *Recurring dominoes: making the highly undecidable highly understandable*, Ann. Discrete Math. 24 (1985) 51–72.

[15] Jones, J.P., *Universal Diophantine equation*, J. Symb. Logic 47 (1982) 549–571.

[16] Jones, J.P. & Matijasevič, Y.V., *Register machine proof of the theorem on exponential Diophantine representation of enumerable sets*, J. Symb. Logic 49 (1984) 818–829.

[17] Levin, L.A., *Universal'nie zadachi perebora*, Problemi Peredachi Informatsie IX (1973), pp. 115–116 (in Russian).

[18] Lewis, H.R., *Complexity of solvable cases of the decision problem for the predicate calculus*, Proc. IEEE FOCS 19 (1978), pp. 35-47.

[19] Lewis, H.R., *Complexity results for classes of quantificational formulas*, J. Comput. Syst. Sci. 21 (1980) 317–353.

[20] Lewis, H.R. and Papadimitriou, C.H., *Elements of the Theory of Computation*, Prentice-Hall (1981).

[21] Lucas, E., *Sur les congruences des nombres Eulériens et des coefficients différentiels des fonctions trigonométriques suivant un module premier*, Bull. Soc. Math. de France 6 (1878) 49–54.

[22] Karp, R.M., *Reducibilities among combinatorial problems*, in. R.E. Miller & J.W. Thatcher (eds.) Complexity in computer computations, Plenum (1972) pp. 85–103.

[23] Kozen, D. & Parikh, R., *An elementary proof of the completeness of PDL*, Theor. Comp. Sci., 14 (1982) 348–359.

[24] Kummer, E.E., *Über die Ergänzungssätse zu den algemeinen Reciprocitätsgesetzen*, Crelle J. 44 (1852) 93–146.

[25] Matijasevič, Y.V., *Diophantine representation of enumerable predicates*, Izvestija Akademii Nauk SSSR, Ser. Math. 35 (1971) 3–30. (transl. Math. USSR Izvestija 4 (1971) 1–28).

[26] Matijasevič, Y.V., *A new proof of the theorem on exponential Diophantine representation of enumerable sets*, Zapiski Naučnyh Seminarov Leninsgradskogo Otdelenija Matiatičeskogo Instituta im. V.A. Steklova (LOMI) Akademia Nauk SSSR 60 (1976) pp. 77–93. (transl. J. Soviet Math. 14 (1980) 1475–1486).

[27] Pratt, V.R., *Semantical considerations on Floyd-Hoare logic*, proc. IEEE FOCS 17 (1976) 109-121.

[28] Robinson, R.M., *Undecidability and non periodicity for tilings in the plane*, Inv. Math. 12 (1971) 177-209.

[29] Savelsberg, M.P.W. & van Emde Boas, P. *BOUNDED TILING, an alternative to SATISFIABILITY?*, in G. Wechsung (ed.) proc. 2nd Frege Memorial Conference, Schwerin, Sep 1984, Akademie Verlag, Mathematische Forschung vol. 20, 1984, pp. 401–407.

[30] Sherman, R. & Harel, D., *A combined proof of one-exponential decidability and completeness for PDL* in L. Priese (ed.), proc. 1st GTI Workshop, Paderborn Oct. 1982, Rheie Theoretische Informatik UGH Paderborn. pp. 221–233.

[31] van Emde Boas, P. *Dominoes are forever*, in L. Priese (ed.), proc. 1st GTI Workshop, Paderborn Oct. 1982, Rheie Theoretische Informatik UGH Paderborn. pp. 75–95

[32] van Emde Boas, P. *Machine models and simulations*, to appear in J. van Leeuwen (ed.), Handbook of theoretical computer science, North Holland Publ. Comp. 1989. Preprint: rep. ITLI-CS-89-02.

[33] Wang, H. *Proving theorems by pattern recognition*, Bell System Tech. J. 40 (1961) 1–41.