# Tiling problems



Tiling problems are basic combinatorial problems that are well suited as target problems for reductions to show both complexity and undecidability results.

Tilings have an intuitive correspondence with TM computations, which we want to explore.

A tile is a square of unit size divided in four triangles by the two diagonals. Tiles cannot be rotated or reflected.

A tile type is obtained by assigning to each of the four triangles a colour from a finite set of colours.

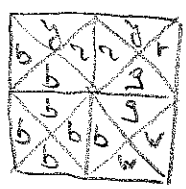
E.g. 

For a tile type $t$, we use $l(t)$, $r(t)$, $u(t)$ $b(t)$ to denote the left, right, upper, and bottom color of $t$

Consider a fixed finite set $C$ of tile types.

A $C$-tiling is a mapping from a subset of the square grid in the plane to $C$. Equivalently, a $C$-tiling is a covering of a region in the plane by instance of tiles in $C$.

The basic property of a tiling is that adjacent tiles must have the same color on the adjacent triangles.

E.g. 

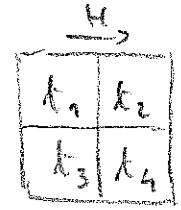
The coloring $C$ induces a horizontal compatibility relation $H_C$ and a vertical $V_C$

E10.2

$$H_C = \{ (t_1, t_2) \mid t_1, t_2 \in C \text{ and } r(t_1) = l(t_2) \}$$

$$V_C = \{ (t_1, t_2) \mid t_1, t_2 \in C \text{ and } b(t_1) = u(t_2) \}$$

Note: the relation V "points" downwards




$(t_1, t_2) \in H$

$(t_1, t_3) \in V$

An alternative way of defining tiling problems is to start directly from a set X of tile types (which are just symbols) and define on X two relations H and V.

Note: - to each set C of tile types with colors correspond the two binary relations $H_C$ and $V_C$, but

- there is a set X and relations H, V on X for which there is no coloring that directly corresponds to them.

The reason is that a coloring C imposes a specific condition on $H_C$ and $V_C$:

if $(t_1, t_2), (t_1, t_2'), (t_3, t_2) \in H_C$,

then also $(t_3, t_2') \in H_C$

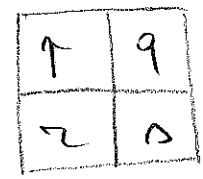
(similarly for $V_C$)

However, given an arbitrary $X$ and relations $H, V$, we can define an extended set $C$ of colored tile types that correspond to it:

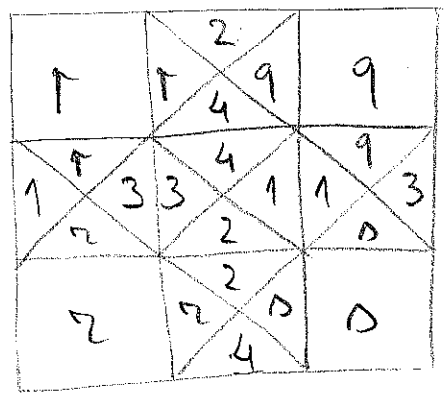
Consider $\uparrow, 9, \rightarrow, \circ \in X$, with

$(\uparrow, 9), (\rightarrow, \circ) \in H$

$(\uparrow, \rightarrow), (9, \circ) \in V$



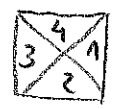

Then we can define a set $C$ of colored types that "corresponds" to $H$ and $V$. We need to introduce intermediate tiles:




We use  for 

Hence $C = X \cup \{$  $\mid (\uparrow, 9) \in H\}$

$\cup \{$  $\mid (\uparrow, \rightarrow) \in V\}$

$\cup \{$  $\}$

We have that $H = (H_C \circ H_C)|_X$

$V = (V_C \circ V_C)|_X$

($R|_X$ denotes $R$ restricted on $X$)

A tiling problem consists of a set $C$ of tile types and a region $R$ of the plane to be tiled using tiles of $C$. Additional conditions on the tiling may be given, such as
- a given colouring along (part of) the edge of $R$, or
- a given tile already placed on $R$.

The following tiling problems have been considered:

- Bounded Tiling:
  input: $C$
  $R$ is an $n \times n$ square, with a given colouring along the edge
  question: is there a $C$-tiling of $R$ extending the edge colouring?

- Corridor Tiling
  input: $C$
  pair of segments top and bot of length $n$ with given colouring
  question: is there a $C$-tiling of the $n \times m$ region with colouring top and bot on the top and bottom edges, and white colour on the left and right edges?

- Origin constrained Tiling
  input: $C$
  a single tile type $t_0 \in C$
  question: is there a $C$-tiling of the entire plane where $t_0$ is placed at the origin

We can exploit a correspondence between tilings and TM computations to show the following results:

1) Bounded Tiling is NP-complete
2) Corridor Tiling is PSPACE-complete
3) Origin Constrained Tiling is recursively enumerable but non-recursive.

We give an idea of how tilings can be used to simulate the computation of a TM M:

We give a construction in which the set C of tile types is entirely determined by M.

Idea: colouring of successive horizontal lines encode successive IDs of the computation of M

We let time run downwards.

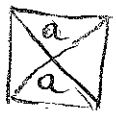
To encode an accepting computation of M on w:
- the top row corresponds to the initial ID on input w
- the bottom row encodes an accepting ID

Consider $M = (Q, \Sigma, \Gamma, \delta, q_0, \#, F)$

We use as horizontal colours: $Q \cup \{w\}$ w... white
vertical colours: $\Gamma \cup (Q \times \Gamma)$

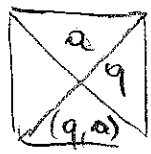
We use the following tile types:

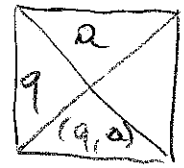
1) to encode that a tape symbol is not changed from one ID to the next:



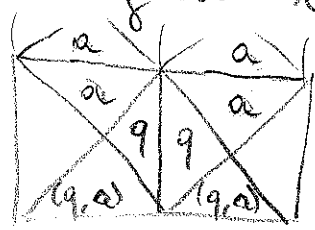

for each $a \in \Gamma$

2) To encode that the head may move to a tape symbol either from the left or from the right




Note that having both these tile-types creates a problem:




$ID_1$

$ID_2$

When in $ID_1$ we have two a next to each other, in $ID_2$ two "heads have appeared"

We can avoid this by "normalizing" the TM transitions, in such a way that each state gets a direction: when moving to a state $q$, the head either moves left or right, but not both.

We can normalize the TM by duplicating each state $q$, creating a "left copy" $q_L$ and a "right copy" $q_R$

$\delta(q,a) = (q, b, R)$ becomes $\delta(q,a) = (q_R, b, R)$

$\delta(q,a) = (q, b, L)$ becomes $\delta(q,a) = (q_L, b, L)$

for all transitions of $M$

Moreover, we let $q_R$ and $q_L$ behave as $q$:

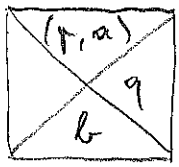
$\delta(q,a) = (q', b, d)$ becomes $\delta(q_R, a) = (q', b, d)$

$\delta(q_L, a) = (q', b, d)$

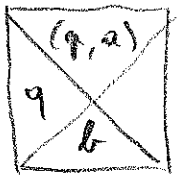
Once $M$ is normalized, for each state $q_R$ or $q_L$ we introduce only the appropriate tile type above.

3) To represent the transitions of M we use, for every $q, q' \in Q$, $a, b \in \Gamma$:

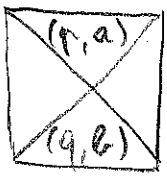
- if $\delta(q, a) = (q', b, R)$



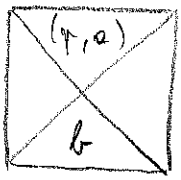

- if $\delta(q, a) = (q', b, L)$




- if $\delta(q, a) = (q', b, S)$




When we have a transition $\delta(q, a) = (q_f, b, -)$ with $q_f \in F$, we let the state "disappear" from the next row




Note that, since M is normalized and the states have a direction, no state $q$ will appear both on the left and on the right of a tile type. Hence, we cannot have that the head "disappears"