A Software to Solve Strip Tiling Problems

Luca Martinelli, Donatella Merlini, Romina Rocchini,
Angela Romoli, and Cecilia Verri

Dipartimento di Sistemi e Informatica
via Lombroso 6/17, 50134, Firenze, Italy
{merlini, verri}@dsi.unifi.it

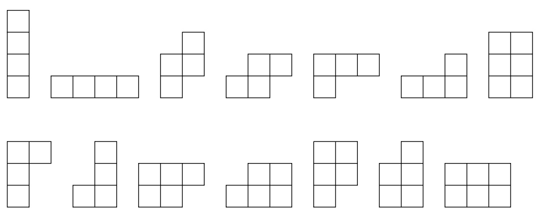
Abstract. We present a C++ program which implements an algorithm able to solve *strip tiling problems*. The program transforms a specified problem into a regular grammar and then finds the generating function counting the number of different ways the strip can be tiled with the given pieces. Some examples and applications are illustrated.

1 Introduction

A *p-strip tiling problem* consists in counting the number of ways a $p \times n$ strip ($p \in \mathbb{N}$ fixed, $n \in \mathbb{N}$) can be tiled with some sort of *pieces*, i.e., sets of simply connected *cells* (squares of one unit length sides).

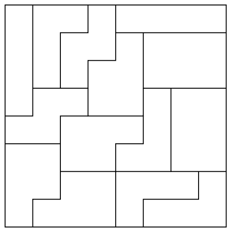
A simple problem of this class (see, e.g., Graham, Knuth and Patashnik [3]) consists in determining the number of ways a strip having width 2 and length n can be tiled with *dimers*, i.e. rectangles having a size of 1×2 units. The solution is F_{n+1} where F_n is the n^{th} Fibonacci number. Dimers have been used as a simple model for gas molecules when they assume two different directions (horizontal and vertical) with respect to the direction of the tube (modelled as a strip) containing the gas (see e.g. [4]).

More complex pieces are taken into consideration by Woan, Rogers and Shapiro in [7]. They consider pieces obtained by *path pairs* of length m (also known as *parallelogram polyominoes*[1] of perimeter $2m$.), i.e., pairs of paths that start at the origin of $\mathbb{Z}^2$, consist of m horizontal and vertical unit steps and meet again for the first time after m steps. If C_m denotes the m^{th} Catalan number, there are exactly C_{m-1} pieces, when m is fixed. For example, for $m = 5$ we have the following 14 pieces:



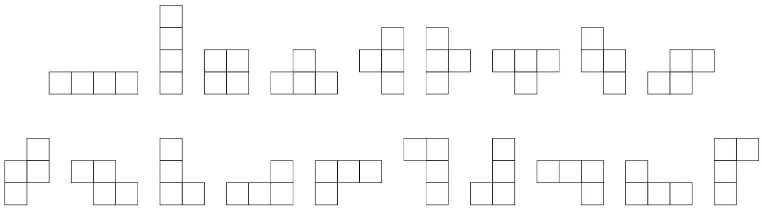
[1] The term *polyomino* comes from [2]

It is well known that the total number of cells of all the C_{m-1} pieces is 4^{m-2}. An unsolved problem is whether or not the C_{m-1} pieces of length m can be used to tile a $2^{m-2} \times 2^{m-2}$ checkboard (each piece must be used exactly once). For example, when $m = 5$ a possible solution is:

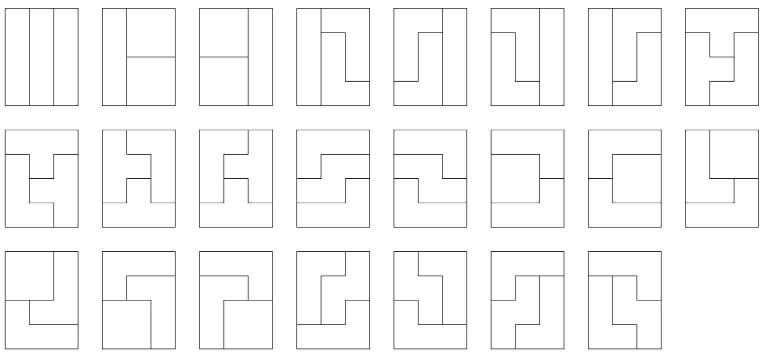


Questions of this kind can be stated as a set of strip tiling problems with $p = n = 2^{m-2}$, and we will call them *Catalan problems*. Obviously, in this way, we also find the tilings of the $2^{m-2} \times 2^{m-2}$ checkboard in which each piece is not used exactly once, but it suffices to determine, among all the possible solutions, those satisying the above restrictions.

An amusing puzzle-problem is to establish in how many different ways a $p \times n$ strip can be tiled with the 19 Tetris pieces:



Typically, we have $p = 10$ but also a smaller value of p leads to non trivial puzzles, due to the large number of solutions as p increases. As a curiosity, here are the 23 solutions corresponding to $p = 4$ and $n = 3$.



We will call this set of strip tiling problems as *Tetris problems*.

For fixed p, if we denote by $T(t) = \sum_n T_n t^n$, the generating function that counts the number T_n of n-length strip tilings, then we can say that this tiling problem is solved if we are able to find an exact or an asymptotic value for T_n and a method for building the tilings.

Recently, Merlini, Sprugnoli and Verri [5] studied the general p-strip tiling problem and proved the following basic results:

1) every tiling problem is equivalent to a regular grammar (i.e., the set of tilings is a regular language);
2) an algorithm exists that can find the regular grammar corresponding to a tiling problem;
3) as a consequence of 1) and 2), it is possible to find the rational function $T(t) = \sum_n T_n t^n$, that counts the n-length strip tilings made up of the pieces assigned. If we also use the indeterminate w to count the number of pieces then it is possible to find the rational function $T(t,w) = \sum_n T_{n,k} t^n w^k$ counting the n-length strip tilings with exactly k pieces;
4) it is possible to find out if there is at least one solution ($T_{n_0} = [t^{n_0}]T(t) \neq 0$) for any value n_0 of n; the number of possible solutions can also be determined ($[t^n]T(t)$ denotes the coefficient of t^n in the generating function $T(t)$).

When the number of pieces and/or the width of the strip is high, the regular grammar becomes very large and therefore $T(t)$ is almost intractable. Anyway, the method gives an automatic way of solving medium-sized problems in a constructive manner, expecially if some effort is done to make an efficient software realisation.

A first implementation was realised by using Maple, due to the symbolic nature of the algorithm. Actually this program is able to solve strip tiling problems with "reasonably little p" and/or "few" pieces. But when the problem complexity grows, the Maple program needs a quantity of time and space which is not available.

In this paper, we describe a more efficient C++ program which realises the algorithm in [5] by taking care of the time and space complexity of the problem. Due to the exponential nature of the algorithm, this new implementation can solve more complex problems than the Maple one, but obviously can not solve all kinds of tiling problem.

2 The C++ program

The C++ program can run on a PC with Windows (95, 98, NT) operating system. It takes as input the definition of a p-strip tiling problem, automatically generates the corresponding regular grammar and applies Schützenberger's methodology (see e.g., [6]) to find out the linear system associated to the grammar; then, two possible choices can be done:

- in some cases, the system can be exactly solved by using Maple;
- some other times, the system is so large that more sophisticated methods are needed, for example methods based on Gröbner bases (see, e.g., [1]). We limited ourselves to use a second C++ program which implements an

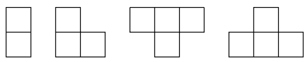
iterative method to find out the series expansion, up to a fixed order, of the desired generating function. We note explicitly that this program is more efficient than the corresponding series expansion in Maple.

We now give a running example of the program. The main menu gives the possibility to choose between a predefined tiling problem or to define a completely new one:

1) TETRIS problem
2) CATALAN problem with p=2
3) CATALAN problem with p=4
4) CATALAN problem with p=8
5) Other

Choice 1) solves the Tetris problem taking p as input. Choices 2), 3) and 4) solve Catalan problem with $m = 3, 4, 5$ and consequently $p = n = 2, 4, 8$. Choice 5) solves a general strip tiling problem taking the set of pieces and p as input. The way to enter pieces is specified during the program execution.

If we select option 5), input $p = 3$ and the pieces:



we find the following regular grammar

$$\begin{aligned}
N1 &::= \epsilon \mid t3N5 \mid t1N4\\
N2 &::= t1N1\\
N3 &::= t4N2\\
N4 &::= t4N6\\
N5 &::= t2N7 \mid t1N3\\
N6 &::= t3N7\\
N7 &::= t2N2 \mid t1N1
\end{aligned}$$

where, $t1, t2, t3, t4$ correspond to the previous pieces and $N1, \ldots, N7$ denote the non-terminal symbols. Then, the program finds the following linear system:

$$\begin{bmatrix}
1 & 0 & 0 & -w & -w & 0 & 0\\
-wt & 1 & 0 & 0 & 0 & 0 & 0\\
0 & -wt^2 & 1 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & 0 & -wt & 0\\
0 & 0 & -wt & 0 & 1 & 0 & -wt^2\\
0 & 0 & 0 & 0 & 0 & 1 & -wt^2\\
-wt & -wt & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix} N1\\ N2\\ N3\\ N4\\ N5\\ N6\\ N7 \end{bmatrix}
=
\begin{bmatrix} 1\\ 0\\ 0\\ 0\\ 0\\ 0\\ 0 \end{bmatrix}$$

where the indeterminate t counts the length of the strip and the indeterminate w the number of pieces. The system obtained is simple and can be solved by using Maple, which gives the generating function:

$$T(t, w) = \frac{1}{1 - w^3t^3 - 3w^4t^4 - w^5t^5} =$$

$$= 1 + w^3t^3 + 3w^4t^4 + w^5t^5 + w^6t^6 + 6w^7t^7 + O(t^8).$$

Anyway, we can find the series expansion of the generating function by using the C++ program. For example, if we ask for 10 iterations we find the following result:

$$T(t) = 1 + t^3 + 3t^4 + t^5 + t^6 + 6t^7 + 11t^8 + 7t^9 + 10t^{10} + \cdots,$$

in terms of the only indeterminate t.

The C++ program described in this paper allowed us to solve a lot of tiling problems in a very efficient way. In particular, by running the program on a Pentium II, 400 MHz, with 64 Mb of RAM, we were able to solve the Tetris problem up to $p = 8$ (with $p = 8$ we obtained a grammar having 49206 non-terminal symbols and 154366 productions) and the Catalan problem with $p = 8$, that is $m = 5$. In this case, the program found a grammar with $145,128$ non-terminal symbols, $374,684$ productions and a generating function having the following series expansion:

$$T(t) = 1 + t + 7t^2 + 169t^3 + 2780t^4 + 31172t^5 +$$

$$+ 460697t^6 + 7866763t^7 + 119195600t^8 + \cdots$$

Among the $119,195,600$ possible tilings of the 8×8 checkboard, the program was also able to determine the 3204 different solutions which use all the 14 pieces.

Obviously, better results should be obtained by running the program on a more efficient computer.

Acknowledgments

We wish to thank Paul Zimmermann for useful discussions about the solution of large sparse linear systems.

References

1. Faugere, J. C.: The GB and FGb systems. Available at `http://posso.lip6.fr/GB.html`.
2. Golomb, S. W.: Polyominoes: puzzles, patterns, problems. Princeton Science Library, Second Edition, 1996.
3. Graham, R. L., Knuth, D. E., Patashnik., O.: Concrete Mathematics. Addison-Wesley, 1989.
4. Lieb, E. H.: Exactly soluble models. Physica, 73:226–236, 1974.
5. Merlini, D., Sprugnoli, R., Verri, M. C.: Strip tiling and regular grammar. To appear in Theoretical Computer Science.
6. Schützenberger, M. P.: Context-free language and pushdown automata. Information and Control, 6:246–264, 1963.
7. Woan, W. J., Rogers, D., Shapiro, L.: The Catalan numbers, the Lebesgue integral and 4^{n-2}. American Mathematical Monthly, December 1997.