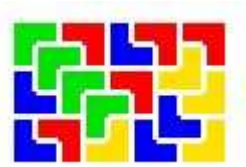
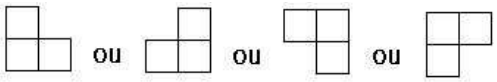
Pavages de rectangles par des formes en équerres

Nous allons utiliser ici la méthode des fonctions génératrices pour calculer le nombre de pavages de rectangles en fonction de la dimension de leurs côtés. Dans les deux exemples qui suivent, les pavés sont dans le premier cas des formes en équerres formées de trois carrés accolés de côté unité, et dans le deuxième cas, soit des équerres formées de trois carrés, soit des équerres allongées, formées de quatre carrés. Seule la longueur des rectangles à paver est quelconque, sa largeur étant fixée, étant entendu que les dimensions des rectangles sont des nombres entiers.

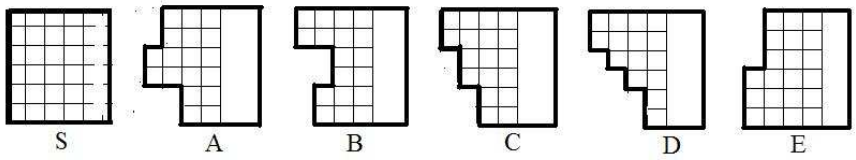
1) Pavages de rectangles 6 sur *N* par des équerres



Ce que nous appelons équerre est plus précisément un *triomino* formé de trois carrés accolés, dans le cas présent à angle droit. En prenant une longueur unité pour chacun de ces trois petits carrés, il s'agit de paver un rectangle de *M* sur *N* unités avec ces équerres, de toutes les façons possibles. Nous traitons ici le cas où le rectangle a pour dimensions 6 verticalement et *N* horizontalement. Les équerres ont quatre dispositions possibles :

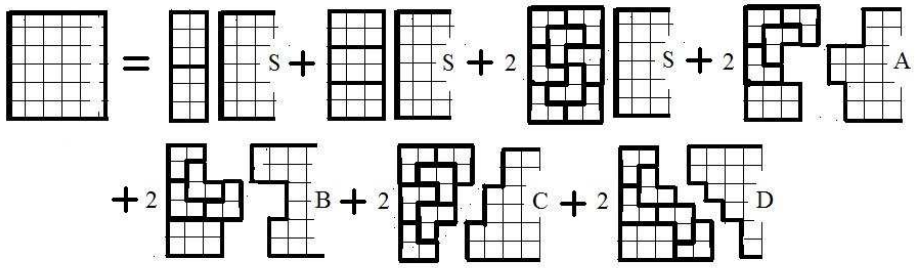


Le classement des pavages suivant leur début fait apparaître plusieurs formes à remplir, dessinées ci-dessous, avec les lettres *S*, *A*, *B*, *C*, *D* correspondant aux fonctions génératrices donnant le nombre de leurs pavages selon le nombre d'équerres constituant le pavage.



Cela nous conduit aux équations suivantes :

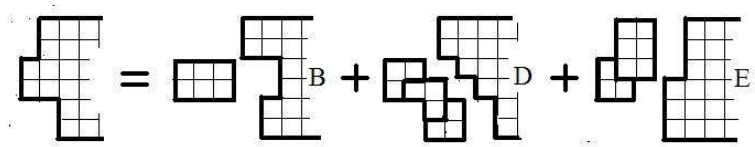
- A partir de *S*



(Remarquons que les rectangles 3 sur 2 peuvent être remplis par deux équerres de deux façons, et que les facteurs 2 présents dans certains cas ci-dessus indiquent que l'on obtient deux formes initiales, la deuxième se déduisant par symétrie d'axe horizontal). L'équation correspondante s'écrit :

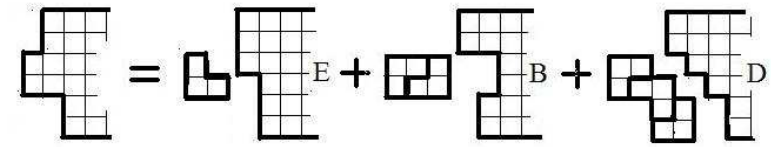
$$S = 1 + 4X^2S + 8X^6S + 2X^8S + 4X^6A + 4X^6B + 2X^6C + 4X^7D$$

- A partir de A :



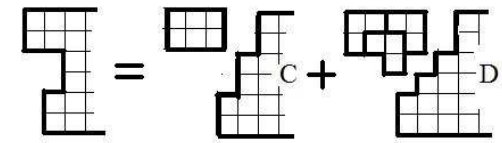
$$A = 2X^2B + X^3D + 2X^3E$$

Remarque : on pouvait aussi bien faire la décomposition suivante :



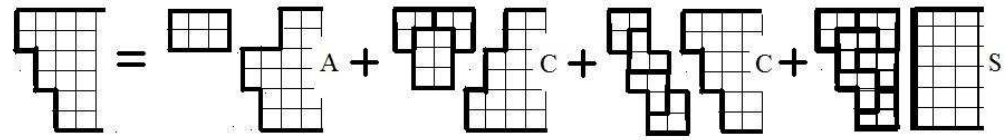
soit $A = XE + X^2B + X^3D$

- A partir de B

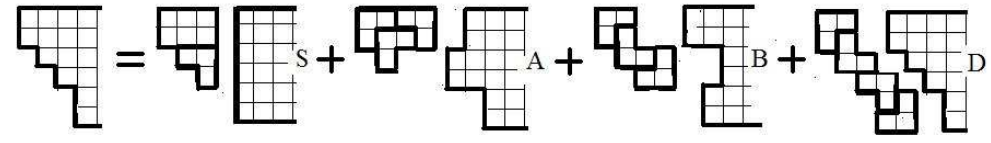


$$B = 2X^2C + X^3D$$

- A partir de C

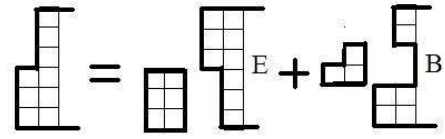


- A partir de D



$$D = 2X^3S + X^3A + X^3B + X^4D$$

- A partir de E



$$E = 2X^2E + XB$$

La résolution du système d'équations conduit à :

$$S = \frac{1 - 2X^2 - 4X^4 - 2X^6 + 13X^8 + 6X^{10} - 6X^{12} - 6X^{14}}{1 - 2X^2 - 8X^4 - 2X^6 + 43X^8 + 42X^{10} - 36X^{12} - 102X^{14} + 44X^{16} + 8X^{20} + 8X^{22}}$$

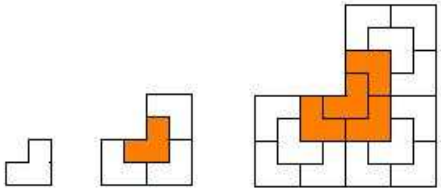
et son développement donne :

$S(X) = 1 + 4\,X^4 + 8\,X^6 + 18\,X^8 + 72\,X^{10} + 162\,X^{12} + + 520\,X^{14} + 1514\,X^{16} + 4312\,X^{18} + 13242\,X^{20} + \ldots$

Pour avoir le nombre de pavages selon la longueur N du rectangle, il suffit de diviser les exposants des puissances de X précédentes par 2, puisque 4 équerres occupent un rectangle de longueur 2.

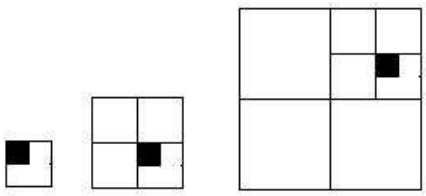
Note sur les triominos à angle droit

On a d'abord cette propriété constructive : toute équerre peut être pavée avec quatre équerres deux fois plus petites. Plus précisément :



A partir d'une équerre découpée en 4 équerres, on peut grossir le dessin en multipliant les longueurs par 2, l'équerre obtenue se trouve divisée en 4 équerres que l'on peut ensuite re-diviser en 4. Et ainsi de suite par ce procédé de montée-descente (*up and down*).

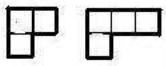
On en déduit ce théorème attribué à S. Golomb (1954) : Tout carré de côté 2^n, amputé d'un carré unité, peut être pavé avec des équerres.



Pour le démontrer procédons par récurrence. On peut paver un carré de côté 2 amputé d'un petit carré unité, évidemment. Puis on double les dimensions du carré. On peut faire un trou dans l'un des quatre carrés obtenus et le paver comme on a su le faire précédemment, puis il reste à paver l'équerre restante avec ses trois carrés. Puis on double les dimensions et l'on recommence…

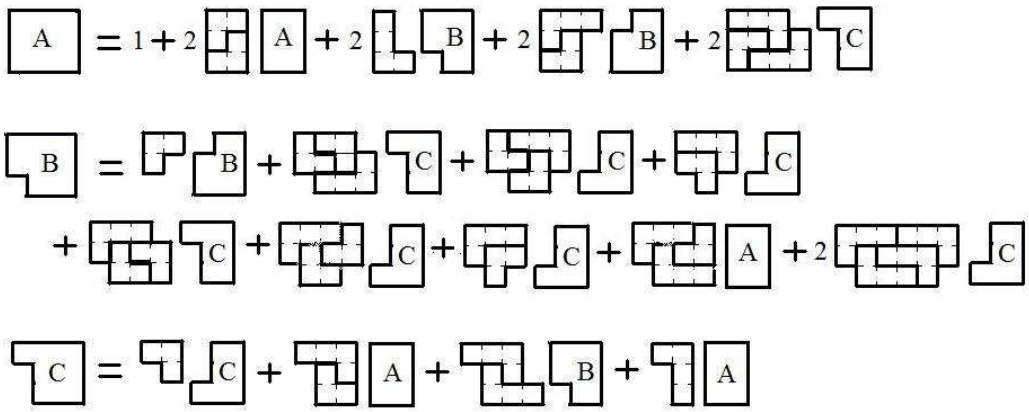
2) Pavages de rectangles de 3 sur *N* avec deux types d'équerres

Prenons maintenant deux types de pavés, soit l'équerre *triomino* précédemment utilisée, soit une équerre longue qui est un *tétramino*, formée de quatre carrés accolés avec un angle droit.[1]



Les deux types d'équerres

Les équerres courtes ont quatre orientations possibles dans les pavages, tandis que les équerres allongées en ont 8. Dans un premier temps, nous allons chercher les fonctions génératrices avec deux variables *X* et *Y*, la première correspondant à l'équerre courte, et *Y* à l'équerre longue. On trouve les décompositions suivantes suivant les débuts de pavages :



Les équations en découlent :

$A = 1 + 2\,X^2\,A + (2Y + 2\,X\,Y)\,B + 2\,X\,Y^2\,C$
$B = X\,B + (X^2\,Y + X^2\,Y + X\,Y + X^2\,Y + X^2\,Y + X\,Y + 2\,Y^4)\,C + X\,Y^2\,A$
$C = X\,C + X\,Y\,A + Y^2\,B + Y\,A$

$$\begin{cases} A(1-2X^2)-(2Y+2XY)B-2XY^2C=1 \\ B(1-X)-XY^2A-(2XY+4X^2Y+2Y^4)C=0 \\ C(1-X)-(Y+XY)A-Y^2B=0 \end{cases}$$

La résolution du système donne :

$$A=\frac{1-2X+X^2-2XY^3-4X^2Y^3-2Y^6}{1-2X-X^2+4X^3-2X^4-10XY^3-20X^2Y^3-12X^3Y^3-6Y^6-8XY^6-2X^2Y^6}$$

En développant :

$$A(X)=4Y^6+8XY^3+16XY^6+2X^2+32X^2Y^3+130X^2Y^6+104X^3Y^3+4X^4 \\ +248X^4Y^3+552X^5Y^3+8X^6+16X^8+\ldots$$

[1] J'ai essentiellement repris ici le travail de Y. Naciri dans le cadre de sa maîtrise et de son DEA sur les pavages. J'ai seulement un peu simplifié les calculs.

Si l'on veut avoir le nombre des pavages selon la longueur L du rectangle, il convient de remplacer chaque terme de la forme $X^n Y^p$ par Z^L où $L = (3n + 4p)/3 = n + (4/3)p$. Les premières valeurs obtenues sont :

- pour $L = 2$: 2 pavages, tous du type X^2
- pour $L = 4$: 4 pavages, tous du type X^4
- pour $L = 5$: 8 pavages, tous du type $X Y^3$)
- pour $L = 6$: 40 pavages, 32 du type X^2Y^3 et 8 du type X^6
- pour $L = 7$: 104 pavages, tous du type X^3Y^3
- pour $L = 8$: 268 pavages, 4 du type Y^6, 248 du type X^4Y^3 et 16 du type X^8
- pour $L = 9$: 568 pavages, 16 du type $X Y^6$ et 552 du type X^5Y^3

Comment faire pour avoir directement le nombre de pavages en fonction de la longueur du rectangle ? Le moyen le plus simple consiste à reprendre les fonctions génératrices avec une seule variable X, dont l'exposant sera égal au nombre de petits carrés unités formant le rectangle à paver. Ainsi, une équerre courte sera remplacée par X^3, et une équerre longue par X^4 dans les équations initiales avec A, B et C. Une fois la fonction génératrice $A(X)$ obtenue, il suffira de diviser les exposants par 3 pour avoir le nombre des pavages suivant la longueur des rectangles.

Dans ce nouveau contexte, les fonctions génératrices obtenues auparavant avec les deux variables X et Y deviennent :

$$(1-X^6)A-(2X^4+2X^7)B-2X^{11}C=1$$

$$(1-X^3)B-X^{11}A-(2X^7+4X^{10}+2X^{16})C=0$$

$$(1-X^3)C-(X^4+X^7)A-X^8B=0$$

On aboutit à :

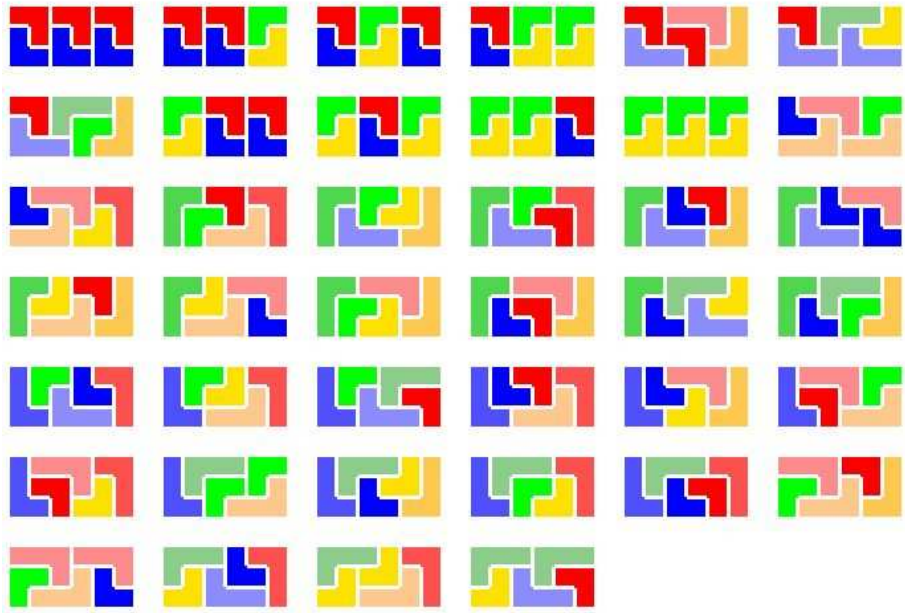
$$A=\frac{1-2X^3+X^6-2X^{15}-4X^{18}-2X^{24}}{1-2X^3-X^6+4X^9-2X^{12}-10X^{15}-20X^{18}-12X^{21}-6X^{24}-8X^{27}-2X^{30}}$$

En développant :

$$A(X)=1+2X^6+4X^{12}+8X^{15}+40X^{18}+104X^{21}+\ldots$$

Finalement le nombre de pavages en fonction de la longueur N des rectangles est :

N	2	4	5	6	7	8	9	10	11
Nombre de pavages	2	4	8	40	104	268	568	1242	2812



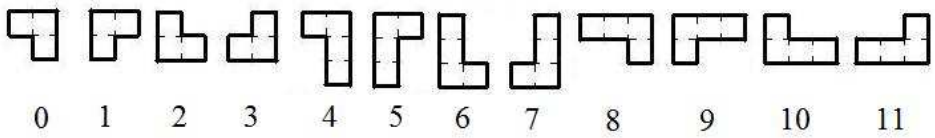
Les 40 pavages du rectangle 6 sur 3

Références :

* Y. Naciri, Pavages à base de polyominos, maîtrise informatique Université Paris 8, 2005.
* R. Graham, D. Knuth, O. Patashnik, *Concrete Mathematics*, Addison Wesley, 1990.
* P. Audibert, Combien ? Mathématiques appliquées à l'informatique, éditions Hermès-Lavoisier, 2008 (en anglais, *Mathematics for Computer Science and Informatics*, Wiley 2010)

Annexe : Programme pour le comptage des pavages avec les deux types d'équerres (seulement en mode texte ici, sans dessins), dans un rectangle de *M* sur *N*.

Les équerres ont 12 dispositions possibles, numérotées ainsi :




```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define M 11
#define N 3
#define MN (M*N)
void explore(int E, int level);
int fait[MN],count,pred[MN];  /* le tableau pred|] n'est utile que si l'on veut en plus afficher tous les
                                 pavés de chaque pavage, ce que nous ne faisons pas ici */
int main()
{  count=0;
   explore(0,1);
```

```
  explore(1,1);
  explore(2,1);
  explore(5,1);
  explore(6,1);
  explore(8,1);
  explore(9,1);
  explore(10,1);
  printf("\n\nNumber of tilings (experimental) =  %d ",count);
  getchar(); return 0;
}

void explore(int E, int level)
{ int i,nbsuccessor,x1,x2,x3,x4,type,newx1,successor[12],nbcasesoccupees=0;
  nbcasesoccupees=0;
  for(i=0;i<MN;i++) if (fait[i]==1) nbcasesoccupees++;
  type=E%12;
  if ((type<4 && nbcasesoccupees==MN-3) || (type>=4 && nbcasesoccupees==MN-4))
   { count++;
   printf("\n%d: ",count);
     for(i=2;i<=level;i++) printf("(%d %d) ",pred[i]/12,pred[i]%12);
     printf("(%d %d) ",E/12,E%12);
    }
  else
   { nbsuccessor=0;
     x1=E/12;type=E%12;

     if (type==0) { x2=x1+1;x3=x1+1+M; fait[x1]=1; fait[x2]=1;fait[x3]=1;  }
    else if (type==1){ x2=x1+1;x3=x1+M;  fait[x1]=1; fait[x2]=1;fait[x3]=1;   }
    else if (type==2){ x2=x1+M;x3=x1+M+1;  fait[x1]=1; fait[x2]=1;fait[x3]=1;  }
    else if (type==3){ x2=x1+M;x3=x1+M-1; fait[x1]=1; fait[x2]=1;fait[x3]=1;  }
    else if (type==4) { x2=x1+1;x3=x1+1+M; x4=x1+1+2*M;
                        fait[x1]=1; fait[x2]=1;fait[x3]=1; fait[x4]=1; }
    else if (type==5){ x2=x1+1;x3=x1+M;  x4=x1+2*M;
                        fait[x1]=1; fait[x2]=1;fait[x3]=1;  fait[x4]=1; }
    else if (type==6){ x2=x1+M;x3=x1+2*M;  x4=x1+2*M+1;
                        fait[x1]=1; fait[x2]=1;fait[x3]=1; fait[x4]=1;  }
    else if (type==7){ x2=x1+M;x3=x1+2*M; x4=x1+2*M-1;
                        fait[x1]=1; fait[x2]=1;fait[x3]=1; fait[x4]=1;  }
    else if (type==8) { x2=x1+1;x3=x1+2; x4=x1+2+M;
                       fait[x1]=1; fait[x2]=1;fait[x3]=1; fait[x4]=1; }
    else if (type==9){ x2=x1+1;x3=x1+2;  x4=x1+M;
                        fait[x1]=1; fait[x2]=1;fait[x3]=1;  fait[x4]=1; }
    else if (type==10){ x2=x1+M;x3=x1+M+1;  x4=x1+M+2;
                        fait[x1]=1; fait[x2]=1;fait[x3]=1; fait[x4]=1;  }
    else if (type==11){ x2=x1+M;x3=x1+M-1; x4=x1+M-2;
                        fait[x1]=1; fait[x2]=1;fait[x3]=1; fait[x4]=1;  }

     newx1=x1+1;     while(fait[newx1]==1) newx1++;

     /**type=0;*/
     if ((newx1+1)%M!=0 &&  newx1+M<MN &&
         fait[newx1+1]==0  &&
         fait[newx1+M+1]==0  )
```

```
        { successor[nbsuccessor]=newx1*12;pred[level+1]=E; nbsuccessor++;}
 /**type=1 */
 if ((newx1+1)%M!=0 &&  newx1+M<MN &&
     fait[newx1+1]==0  &&
     fait[newx1+M]==0  )
       { successor[nbsuccessor]=newx1*12+1;pred[level+1]=E; nbsuccessor++;}

  /**type=2 */
 if ((newx1+1)%M!=0 &&  newx1+M<MN &&
     fait[newx1+M]==0  &&
     fait[newx1+M+1]==0  )
       { successor[nbsuccessor]=newx1*12+2;pred[level+1]=E; nbsuccessor++;}
 /**type=3 */
 if ( newx1+M<MN &&  newx1%M!=0  &&
     fait[newx1+M]==0  &&
     fait[newx1+M-1]==0  )
  { successor[nbsuccessor]=newx1*12+3;pred[level+1]=E; nbsuccessor++;}
/**type=4 */
 if ((newx1+1)%M!=0 &&  newx1+M<MN &&  newx1+2*M<MN &&
     fait[newx1+1]==0  && fait[newx1+1+M]==0
     && fait[newx1+2*M+1]==0  )
       { successor[nbsuccessor]=newx1*12+4;pred[level+1]=E; nbsuccessor++;}
 /**type=5 */
 if ( (newx1+1)%M!=0 &&  newx1+M<MN &&  newx1+2*M<MN &&
     fait[newx1+1]==0  && fait[newx1+M]==0
     && fait[newx1+2*M]==0  )
       { successor[nbsuccessor]=newx1*12+5;pred[level+1]=E; nbsuccessor++;}
 /**type=6 */
 if ( (newx1+1)%M!=0 &&  newx1+M<MN &&  newx1+2*M<MN &&
     fait[newx1+M]==0  && fait[newx1+2*M]==0
     && fait[newx1+2*M+1]==0  )
       { successor[nbsuccessor]=newx1*12+6;pred[level+1]=E; nbsuccessor++;}
 /**type=7 */
 if ( newx1%M!=0  &&  newx1+M<MN && newx1+2*M<MN &&
     fait[newx1+M]==0  && fait[newx1+2*M]==0
     && fait[newx1+2*M-1]==0  )
  { successor[nbsuccessor]=newx1*12+7;pred[level+1]=E; nbsuccessor++;}
/**type=8 */
 if ( (newx1+1)%M!=0 &&  (newx1+2)%M!=0 &&  newx1+M<MN &&
     fait[newx1+1]==0  && fait[newx1+2]==0   && fait[newx1+2+M]==0  )
       { successor[nbsuccessor]=newx1*12+8;pred[level+1]=E; nbsuccessor++;}
 /**type=9 */
 if ( (newx1+1)%M!=0 && (newx1+2)%M!=0 &&  newx1+M<MN &&
     fait[newx1+1]==0  && fait[newx1+2]==0    && fait[newx1+M]==0  )
       { successor[nbsuccessor]=newx1*12+9;pred[level+1]=E; nbsuccessor++;}
 /**type=10 */
 if ( (newx1+1)%M!=0 && (newx1+2)%M!=0 &&  newx1+M<MN &&
     fait[newx1+M]==0  && fait[newx1+M+1]==0     && fait[newx1+M+2]==0  )
       { successor[nbsuccessor]=newx1*12+10;pred[level+1]=E; nbsuccessor++;}
 /**type=11 */
 if ( newx1+M<MN &&  newx1%M!=0  &&  newx1%M!=1 &&
     fait[newx1+M]==0  && fait[newx1+M-1]==0     && fait[newx1+M-2]==0  )
  { successor[nbsuccessor]=newx1*12+11;pred[level+1]=E; nbsuccessor++;}
```

```
      if(nbsuccessor>0)
      for(i=0;i<nbsuccessor;i++) explore(successor[i],level+1);

      if (type<4) {fait[x1]=0; fait[x2]=0;fait[x3]=0;}
      else if (type>=4) {fait[x1]=0; fait[x2]=0;fait[x3]=0;  fait[x4]=0;}
     }
}
```