from this generating function is $(1+\cos n\pi)2^{n-1}+o(2^n)$. What happens is that although valid for all positive n, this expression reduces to $o(2^n)$ when n is odd. Better precision necessitates to look for further terms in the expansion. The ideal algorithm outputs a list of asymptotic expansions depending on arithmetic properties of n. Cancellation in this context is not a trivial problem. For instance, no algorithm is known to determine whether a linear recurrent sequence takes the value 0 for some index. It is known that when such a sequence cancels infinitely often, the indices where it cancels asymptotically form a finite union of arithmetic progressions that can be computed [1], but our problem is different since we are only concerned with indefinite cancellation of the dominant part, which does not satisfy a linear recurrence in general.

Acknowledgement

This work was supported in part by the ESPRIT III Basic Research Action Programme of the E.C. under contract ALCOM II (#7141).

References

[1] Berstel, J., and Mignotte, M. Deux propriétés décidables des suites récurrentes linéaires. *Bulletin de la Société Mathématique de France*, 104 (1976), 175–184.

[2] Bronstein, M., and Salvy, B. Full partial fraction decomposition of rational functions. Preprint, Dec. 1992. To appear in Proceedings ISSAC'93.

[3] Cerlienco, L., Mignotte, M., and Piras, F. Suites récurrentes linéaires. Propriétés algébriques et arithmétiques. *L'Enseignement Mathématique XXXIII* (1987), 67–108. Fascicule 1-2.

[4] Comtet, L. *Advanced Combinatorics*. Reidel, Dordrecht, 1974.

[5] Conway, J. H. The weird and wonderful chemistry of audioactive decay. In *Open Problems in Communication and Computation* (1987), T. M. Cover and B. Gopinath, Eds., Springer-Verlag, pp. 173–188.

[6] Coste, M., and Roy, M.-F. Thom's lemma, the coding of real algebraic numbers and the topology of semi-algebraic sets. *Journal of Symbolic Computation 5* (1988), 121–129.

[7] Dieudonné, J. *Calcul Infinitésimal*. Hermann, Paris, 1968.

[8] Gourdon, X. Algorithmique du théorème fondamental de l'algèbre. Tech. rep. 1852, Institut National de Recherche en Informatique et en Automatique, February 1993.

[9] Jenkins, M. A., and Traub, J. F. A three-stage variable-shift iteration for polynomial zeros and its relation to generalized Rayleigh iteration. *Numerische Mathematik 14* (1970), 252–263.

[10] Mahler, K. An application of Jensen's formula to polynomials. *Mathematika* 7 (1960), 98–100.

[11] Mahler, K. An inequality for the discriminant of a polynomial. *Michigan Mathematical Journal 11* (1964), 257–262.

[12] Pan, V. Algebraic complexity of computing polynomial zeros. *Computers and Mathematics with Applications 14*, 4 (1987), 285–304.

[13] Plouffe, S. Approximations de séries génératrices et quelques conjectures. Master's thesis, Université du Québec à Montréal, Sept. 1992. Also available as Research Report 92-61, Laboratoire Bordelais de Recherche en Informatique, Bordeaux, France.

[14] Rioboo, R. Real algebraic closure of an ordered field. Implementation in Axiom. In *Symbolic and Algebraic Computation* (1992), P. S. Wang, Ed., ACM Press, pp. 130–137. Proceedings of ISSAC'92, Berkeley, July 1992.

[15] Roy, M.-F., and Szpirglas, A. Complexity of the computation on real algebraic numbers, 1988. Preliminary Version.

[16] Schmidt, H. Beiträge zu einer Theorie der allgemeinen asymptotischen Darstellungen. *Mathematische Annalen 113* (1936), 629–656.

[17] Schönhage, A. The fundamental theorem of algebra in terms of computational complexity. Tech. rep., Mathematisches Institut der Universität Tübingen, 1982. Preliminary report.

[18] Sloane, N. J. A. *A Handbook of Integer Sequences*. Academic Press, 1973.

[19] Sturm, C. Mémoire sur la résolution des équations numériques. *Institut de France de Sciences Mathématiques et Physiques 6* (1835), 271–318.

[20] Titchmarsh, E. C. *The Theory of Functions*, second ed. Oxford University Press, 1939.

Generating Convex Polyominoes at Random

Winfried Hochstättler[*] Köln
Martin Loebl, Praha
Christoph Moll[†] Köln

Abstract

We give a new recursion formula for the number of convex polyominoes with fixed perimeter. From this we derive a bijection between an intervall of natural numbers and the polyominoes of given perimeter. This provides a possibility to generate such polyominoes at random in polynomial time. Our method also applies for fixed area and even when fixing both, perimeter and area.

In the second part of the paper we present a simple linear time probabilistic algorithm which uniformly generates convex polyominoes of given perimeter with asymptotic probability 0.5.

1 Introduction

Unit squares having their vertices at integer points in the Cartesian plane are called **cells**. A **polyomino** is a connected finite subset of cells in the grid. The number of cells is the **area** of the polyomino. The length of the boundary is called its **perimeter**.

Polyominoes have been studied – essentially – from two points of view. The question how to tile the plane with polyominoes is adressed (among others) in [Ber81], [Con90], [Mar91]. On the other hand several authors treat enumerative questions concerning polyominoes (cf. [Ben74], [Pol69]). There, two polyominoes are indentical if there is a translation of the plane mapping one to the other. The asymptotic behaviour of certain types of polyominoes has several applications to problems arising in percolation theory in statistical physic (cf. [Gutt88], [Lin88]). For a survey on recent results concerning the enumeration of polyominoes see [Del91].

A polyomino is called **convex**, if the intersection with any horizontal or vertical line is connected. A few years ago M.-P. Delest and G. Viennot ([Del84]) found that the number of convex polyominoes of perimeter $2n+8$ is given by $P_{2n+8}=(2n+11)4^n-4(2n+1)\binom{2n}{n}$. The proof of this formula is not bijective and thus does not provide a possibility to generate them at random.

In this paper we present a new recursion formula for the number P_n of convex polyominoes of given perimeter n. We will construct a bijection from $\{1,\ldots,P_n\}$ into the set

[*]supported by the German Research Association (Deutsche Forschungsgemeinschaft, SFB 303)
[†]supported by the German Research Association (Deutsche Forschungsgemeinschaft)

of polyominoes with perimeter n. This bijection leads to a polynomial time algorithm that generates polyominoes with fixed perimeter with uniform probability. The same method can be used to produce random polyominoes with fixed area or fixed area and fixed perimeter. Furthermore, we suggest a simple probabilistic algorithm with linear running time for this task.

2 Counting Polyominoes

A polyomino P of perimeter n determines a smallest rectangle $r(P)$ of perimeter n which circumscribes it. A rectangle s of height 1 is called a **strip**. We consider each polyomino as a sequence of strips $(s_1, \ldots, s_k)$, where s_1 is the topmost strip and s_k is the downmost strip of the polyomino. Fixing an embedding of a strip s we denote by $L(s), R(s)$ the horizontal coordinates of the leftmost, resp. rightmost point of s.

Definition 1 *Let $P = (s_1, \ldots, s_k)$ be a polyomino, s_l the downmost strip with $L(s_l)$ minimal and s_r the downmost strip with $R(s_r)$ maximal. We partition the set $(s_1, .., s_k)$ as follows (cf. figure 1):*

i. the **top** $s_1, \ldots, s_{min(l,r)}$,

ii. the **interior** $s_{min(l,r)+1}, \ldots, s_{max(l,r)}$,

iii. the **bottom** $s_{max(l,r)+1}, \ldots, s_k$.

We say the interior is of **eastern type** *if $l < r$ and of* **western type** *if $l > r$.*

Note that the top is always nonempty while interior and bottom may be empty.

Given a polyomino it is an easy task to construct its partition into top, interior and bottom.

We consider four different types of polyominoes.

(t) Polyominoes with empty interior and empty bottom

(i^w) Polyominoes with empty bottom and western interior

(i^e) Polyominoes with empty bottom and eastern interior

(b) Polyominoes with non-empty bottom

Polyominoes of type t are called stack polyominoes and have been studied in [Del84].

In our bijection we will construct a convex polyomino P strip by strip starting from the top.

Given a partially constructed polyomino $\tilde{P}$, determined by a sequence of strips $s_1, .., s_j$ we consider the set of all possible extensions of $\tilde{P}$ to a convex polyomino P.

Lemma 1 *Let $\tilde{P} = (s_1, \ldots, s_j)$ be a convex polyomino. Then the set of all extensions $(s_{j+1}, \ldots, s_k)$ of $\tilde{P}$ to a convex polyomino $P = (s_1, \ldots, s_k)$ is determined by the type of $\tilde{P}$ and the last strip s_j.*

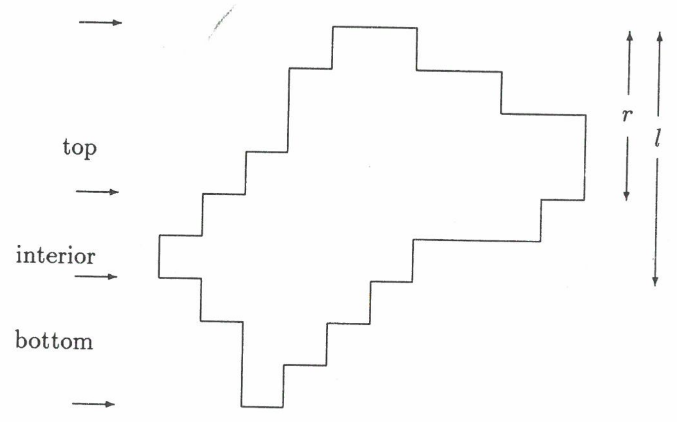


Figure 1: Polyomino of western type

Proof. If $\tilde{P}$ is of type t, then P is a convex polyomino if and only if $(s_j, \ldots, s_k)$ is a convex polyomino. For polyominoes of type i^w (i^e) in addition $R(s_i) \leq R(s_j)$ ($L(s_i) \geq L(s_j)$) must hold for $i > j$. If $\tilde{P}$ is of type b every extension has to respect both inequalities.

□

These facts were used in [Ben74],[Del84] and [Lin88] to count polyominoes by cutting them into two or three parts. We apply this lemma to develop a recursion formula for the number of possible extensions of partially constructed polyominoes to polyominoes of a fixed perimeter, fixed interior or fixing both. In this paper we only argue the case of fixed perimeter n. The other cases can be treated by the same method.

The number of possible extensions of $\tilde{P}$ (of perimeter $\tilde{n}$) to a convex polyomino P of perimeter n only depends on $\tilde{m}$, the length of the bottom line of $\tilde{P}$, and $\tilde{l} = n - \tilde{n} + \tilde{m}$, the remaining length, that is the length of the path replacing the bottom line in P and the type of $\tilde{P}$. Note, that $\tilde{m} + \tilde{l}$ is always even.

Using symmetry in the cases i^e and i^w we define $N_b(\tilde{m}, \tilde{l})$, $N_i(\tilde{m}, \tilde{l})$ and $N_t(\tilde{m}, \tilde{l})$ to be the number of possible extensions in the remaining three cases.

If $\tilde{l} = \tilde{m}$ then the only extension is the empty polyomino. This yields the start up conditions for our recursion.

$$N_b(\tilde{m}, \tilde{l}) = N_i(\tilde{m}, \tilde{l}) = N_t(\tilde{m}, \tilde{l}) = \begin{cases} 1 & \text{for} \quad \tilde{l} = \tilde{m} \\ 0 & \text{for} \quad \tilde{l} < \tilde{m} \end{cases}$$

Obviously, the simplest case is the one with the least liberties. Thus we first consider the case, where we count the number of extensions of $\tilde{P}$ of type b with bottom length $\tilde{m}$ and perimeter $\tilde{n}$ to convex polyominoes with perimeter n ($n = \tilde{n} - \tilde{m} + \tilde{l}$ for a given remaining length $\tilde{l}$) (cf. figure 2).

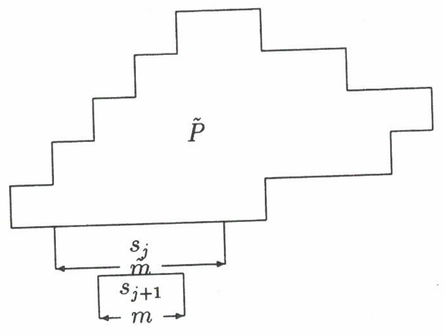


Figure 2: The bottom case

We already are in the bottom case and hence the extending strips s_i must be monotonic increasing with respect to $L(s_i)$ and monotonic decreasing in $R(s_i)$ (cf. Lemma 1).

Now, fixing $m = L(s_{j+1}) - R(s_{j+1})$ we have $\tilde{m} - m + 1$ possibilities to attach s_{j+1} to $\tilde{P}$. Let l denote the length of the path replacing the bottom line of $(s_1, .., s_{j+1})$. $\tilde{l}$ denotes the length of the path replacing the bottomline of $\tilde{P}$. Hence $\tilde{l} = \tilde{m} - m + 2 + l$ and

$$N_b(\tilde{m}, \tilde{l}) = \sum_{m=1}^{\tilde{m}} (\tilde{m} - m + 1) N_b(m, \tilde{l} - 2 - \tilde{m} + m) \qquad \text{for } \tilde{l} > \tilde{m}, \tilde{l} + \tilde{m} \text{ even.} \tag{1}$$

Next, we count the number of extensions $N_i(\tilde{m}, \tilde{l})$ of polyominoes with empty bottom and non empty interior (type i). We have to consider all strips with $R(s_{j+1}) \leq R(s_j)$ in the western case ($L(s_{j+1}) \geq L(s_j)$ in the eastern case). By symmetry we may restrict our considerations to the western case.

If $L(s_{j+1}) > L(s_j)$ then the polyomino $(s_1, .., s_{j+1})$ is of type b and analogous to the first case we get

$$\sum_{m=1}^{\tilde{m}-1} (\tilde{m} - m) N_b(m, \tilde{l} - 2 - \tilde{m} + m)$$

extensions with $L(s_{j+1}) > L(s_j)$.

For the second part of the interior recursion we have $L(s_{j+1}) \leq L(s_j)$. We abbreviate $a = R(s_j) - R(s_{j+1})$ and $b = L(s_j) - L(s_{j+1})$ (see figure 3). Then $0 \leq a < \tilde{m}$. The parameter a will become one running index of the sum in our recursion. Fixing a, we have to determine the feasible values of m.

Following the path from the left endpoint to the right endpoint of $\tilde{P}$ in the polyomino P we get

$$\tilde{l} = b + 1 + l + 1 + a.$$

Furthermore we have $a + m = \tilde{m} + b$, and hence

$$l + m = \tilde{l} + \tilde{m} - 2a - 2.$$

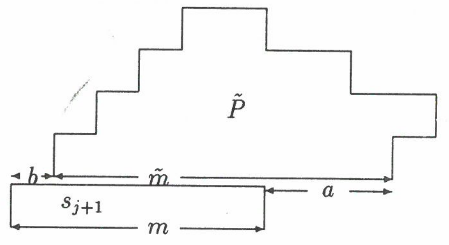


Figure 3: In the interior case

From $m \leq l$ we conclude $m \leq \frac{\tilde{l}+\tilde{m}}{2} - a - 1$. On the other hand $m \geq \tilde{m} - a$, which leads to the boundary conditions.

$$N_i(\tilde{m}, \tilde{l}) = \sum_{m=1}^{\tilde{m}-1} (\tilde{m} - m) N_b(m, \tilde{l} - 2 - \tilde{m} + m)$$

$$+ \sum_{a=0}^{\tilde{m}-1} \sum_{\tilde{m}-a}^{\frac{\tilde{l}+\tilde{m}}{2}-a-1} N_i(m, \tilde{l} + \tilde{m} - 2a - 2 - m) \tag{2}$$

Finally, we determine the number $N_t(\tilde{m}, \tilde{l})$ of extensions of polyominoes $\tilde{P}$ of type t.

As before the formulas induced by a change to a different type only differ in the boundaries and the coefficients of the sums. Hence $N_t(\tilde{m}, \tilde{l}) = N_t^b + N_t^i + N_t^t$. Counting the extensions to polyominoes with $L(s_{j+1}) > L(s_j)$ and $R(s_{j+1}) < R(s_j)$ (switching to the bottom case) yields

$$N_t^b = \sum_{m=1}^{\tilde{m}-2} (\tilde{m} - m - 1) N_b(m, \tilde{l} - 2 - \tilde{m} + m)$$

adding the extensions with $L(s_{j+1}) \leq L(s_j)$ and $R(s_{j+1}) < R(s_j)$ or $L(s_{j+1}) > L(s_j)$ and $R(s_{j+1}) \geq R(s_j)$ (interior - western or eastern type)

$$N_t^i = 2 \sum_{a=1}^{\tilde{m}-1} \sum_{\tilde{m}-a}^{\frac{\tilde{l}+\tilde{m}}{2}-a-1} N_i(m, \tilde{l} + \tilde{m} - 2a - 2 - m)$$

and at last, using the same arguments, the extensions with $L(s_{j+1}) \leq L(s_j)$ and $R(s_{j+1}) \geq R(s_j)$ (staying in the top) are

$$N_t^t = \sum_{m=\tilde{m}}^{\frac{\tilde{m}+\tilde{l}}{2}-1} (m - \tilde{m} + 1) N_t(m, \tilde{l} - 2 + \tilde{m} - m). \tag{3}$$

From this relations we derive the number of polyominoes with fixed perimeter n considering all possible strips s_1 and counting their extensions. Thus, we have proven the following theorem

Theorem 1 *The total number of convex polyominoes of perimeter n is*

$$P_n = \sum_{\tilde{m}+2+\tilde{l}=n} N_t(\tilde{m},\tilde{l}). \tag{4}$$

□

Remark 1 *It is easy to see that $N_b(m,l) = \binom{l-1}{m-1}$. Explicit formulas for N_i and N_t should provide a bijective proof of the total number of polyominoes.*

3 Constructing the bijection

The considerations in the last chapter induce a tree, where the leaves are the polyominoes of a given perimeter, interior nodes are partially constructed polyominoes and the root is the empty polyomino.

The numbers N_b, N_i, N_t and P_n associated with the interior nodes, resp. the root, give the number of leaves of the corresponding subtrees. Now, consider an embedding of such a tree in the euclidian plane. This gives rise to a linear order on the leaves and hence a numbering of the polyominoes of perimeter n. Obviously, for any given node, the set of numbers of the leaves in its subtree is an intervall.

In order to compute the number of a given polyomino and vice versa we have to fix an embedding. To do so, we fix some ordering of the summands in the equations (1) – (4). Furthermore, we compute the table of the numbers $N_d(\tilde{m},\tilde{l})$, $N_i(\tilde{m},\tilde{l})$, $N_t(\tilde{m},\tilde{l})$ for $\tilde{m}+\tilde{l}$ even and $\tilde{m}+\tilde{l} \leq n-2$, and the total number P_n of polyominoes using formulas (1) – (4). Note that this table can be computed in $O(n^5)$ and requires $O(n^3)$ memory, for the size of the numbers is bounded by $\ln(P_n) = O(n)$.

Our algorithm to compute the bijection between the polyominoes of perimeter n and the intervall $[1, P_n]$ proceeds as follows:

Given a polyomino $p = (s_1, \ldots, s_k)$ we compute the partition of the polyominoes of perimeter n induced by their first strip. The ordering of the terms in (4) yields a partition of the interval $[1, P_n]$. Thus s_1 determines an interval I_1. Applying this procedure recursively – always taking into account the type of the already touched part of p – we end up in a leaf of the tree with interval $[j, j]$.

On the other hand, given a number j in $[1, P_n]$ we compute the partition of the interval $[1, P_n]$, fix the corresponding strip s_1 and proceed recursively.

The depth of the tree and its maximum degree is bounded by $\frac{n-2}{2}$. Hence the above computations can be performed in $O(n^3)$, $(O(n^2)$ using uniform measure).

An example computation can be found in figure 4. There, we fixed the ordering of the summands as they occur in the formulas.

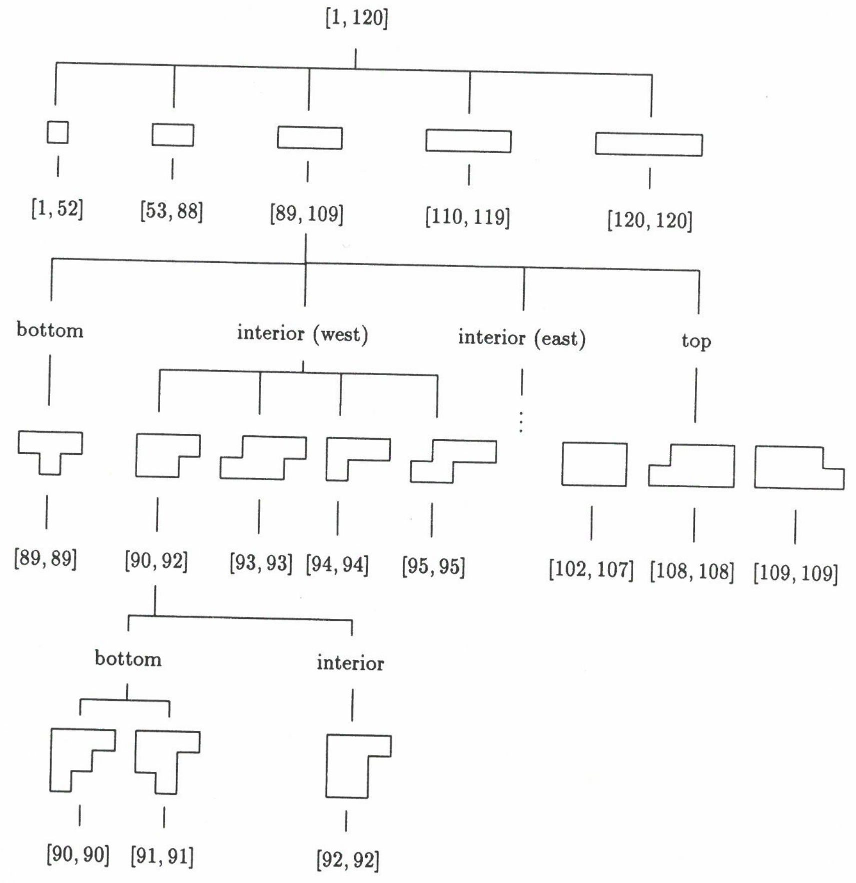
To generate random polyominoes of perimeter n we proceed as follows:

Calculate the table and P_n, compute a random number $j \leq P_n$ and apply the above procedure.

Theorem 2 *A random polyomino of perimeter n can be computed in $O(n^3)$ with $O(n^5)$ preprocessing time and $O(n^3)$ memory requirement.*

□

The same technique can be used to compute random polyominoes with fixed area A or fixed area and fixed perimeter. In the first case the preprocessing time is bounded by $O(A^4)$ and $O(A^3)$ memory is required. In the second case we need $O(An^4)$ time and $O(An^3)$ memory.



Nontrivial values used in the example :

- $N_t(1,9) = 52, N_t(2,8) = 36, N_t(3,7) = 21, N_t(4,6) = 10$
- $N_b(3,5) = 1, N_i(2,4) = 3, N_i(1,3) = 1, N_b(3,5) = 6$

Figure 4: Computing the 91st polyomino of perimeter 12

4 A probabilistic algorithm

The algorithm developed in the last chapter has a huge memory requirement. Furthermore, an exact long integer arithmetic is necessary to compute random polyominoes of large perimeter with this approach. Thus, our method is hardly applicable for practical use. In this section we develop a very simple probabilistic algorithm with linear time and space requirement. Using the formula for the total number of convex polyominoes with fixed perimeter, we prove that this algorithm succeeds with asymptotic probability 0.5.

A well known approach to count polyominoes is to study the language, which is defined by the boundaries of convex polyominoes encoded as sequences of 'L'eft, 'R'ight, 'U'p and 'D'own moves. To develop a probabilistic algorithm we first collect some obvious properties of these cyclic 'LURD'- sequences. Then, we will use these properties to encode convex polyominoes of perimeter n as a pair of a number k in $[0, \frac{n}{2}]$ and a 0-1 string s of size $n-6$. To generate a random polyomino we first guess a pair (k, s). Then, we check if this pair corresponds to a word of the language defined by the convex polyominoes. If it is, we decode it.

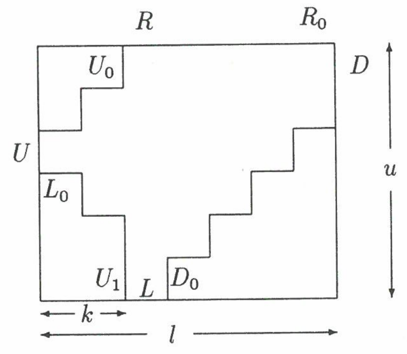


Figure 5: Some properties of an $LURD$ encoding

Lemma 2 *Let P be a convex polyomino of perimeter n, T the cyclic 'LURD'-sequence encoding the boundary of P, $l(r,u,d)$ the number of 'L's ('R's, 'U's, 'D's resp.) in T. Furthermore, let $T_{UD}(T_{LR})$ denote the cyclic 'UD' ('LR') subsequences of T, then the following holds (cf. figure 5):*

i. *$l = r > 0$, $u = d > 0$*

ii. *There is a unique $L =: L_0$ $(R =: R_0)$ with successor R (L) in T_{RL} and its successor in T is a U (D).*

iii. *There is a unique $U =: U_0$ $(D =: D_0)$ with successor D (U) in T_{UD} and its successor in T is an R (L).*

iv. *The smallest rectangle Q circumscribing P is of height u and width l.*

□

Note, that every LURD-sequence with these properties represents a closed walk in the rectangle Q, that touches every side only once in the correct order and has no reverse step. Obviously a LURD-sequence with these properties encodes a convex polyomino if and only if the induced path is simple. In the following we consider LURD-sequences as words starting with the first $U =: U_1$.

Checking simplicity of a $LURD$-sequence can be done in linear time using the algorithm in figure 6. We start at the downmost leftmost point and construct the left and the right side strip by strip.

```
Function CHECKSIMPLICITY
(0) $l = 0$, $r = 0$
(1) Fix the left side: While the first entry is an 'L' ('R') remove the first entry and
        change $l$ to $l-1$   $(l+1)$.
(2) Fix the right side: While the last entry is an 'R' ('L') remove the last entry and
        change $r$ to $r-1$   $(r+1)$.
(3) Start next strip : If $l < r$ then remove first and last entry and go to (1).
(4) Check : If the sequence is empty then return SIMPLE else return NOTSIMPLE.
```

Figure 6: Checking simplicity of an $LURD$-word

We will prove that in most cases LURD-sequences with (i)-(iii) encode simple paths. Thus, to generate convex polyominoes at random we might generate uniformly distributed LURD-sequences with (i)-(iii) until we get a simple path. Instead of trying this, we eliminate the corresponding redundancy.

In order to achieve this we consider words on the alphabet $\{V, H\}$ and map the LURD-sequences to vertical-horizontal-sequences writing an 'V' for every 'U' and 'D' and an 'H' for every 'L' and 'R'. In addition we store the number of 'L's before the first 'R' (the distance of the starting point U_0 to the left side of the rectangle). This enables us to reconstruct the LURD-sequence from the corresponding VH-sequence. We can encode LURD-sequences as a pair of a binary string of length n and a number k in $0, \ldots, \frac{n}{2} - 1$. To eliminate some more redundancy of this VH-k-encoding we observe the following:

Lemma 3 *Let T be a LURD-sequence with properties (i)-(iii), S its VH-k-encoding, h (v) the number of 'H's ('V's resp.) in the VH-sequence then:*

i. *h and v are even.*

ii. *After the k-th and after the $k + \frac{h}{2}$-th 'H' there is a 'V'.*

iii. *After the $\frac{v}{2}$-th and after the last 'V' there is an 'H'.*

iv. *The first symbol is a 'V'.*

□

With this information we can reduce our encoding to a binary string of length $n-6$ and a k in $0, \dots, \frac{n}{2}$. To do so, we first remove 4 redundant letters of our encoding using (ii) and (iii). Moreover, (i) allows us to forget about the last letter in the remaining string, the parity bit. To utilize (iv) we have to consider two cases. If $k \neq 0$ the first entry of the remaining string must be a 'U' and we can delete it. If $k = 0$ there is a minor problem. The first 'U' has already been deleted using (ii). To get a binary string of length $n-6$ we delete the first entry and store this information in k. If the first entry is an 'H' we change k to $\frac{n}{2}$, else we leave k unchanged.

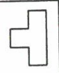
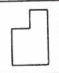
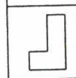
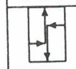
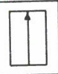
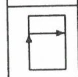
	LURD-sequence	VH-k-sequence	Step 2-3	Encoding
	ULURURDDDL	1 - $\mathrm{VH\underline{V}HV\ \underline{HV}VV\underline{H}}$	1 - $\mathrm{\underline{V}HHVV\underline{V}}$	1 - HHVV
	UURURDDDLL	0 - $\mathrm{\underline{V}VHV\underline{HV}VV\underline{H}H}$	0 - $\mathrm{\underline{V}HVVV\underline{H}}$	0 - HVVV
	URUURDDDLL	0 - $\mathrm{\underline{V}HVV\underline{HV}VV\underline{H}H}$	0 - $\mathrm{\underline{H}VVVV\underline{H}}$	5 - VVVV
	URUURDLDDL	0 - $\mathrm{\underline{V}HVV\underline{HV}HVV\underline{H}}$	0 - $\mathrm{\underline{H}VVHV\underline{V}}$	5 - VVHV
		1 - $\mathrm{VVV\underline{HV}HH\underline{V}V\underline{H}}$	1 - $\mathrm{\underline{V}VVHH\underline{V}}$	1 - VVHH
		0 - $\mathrm{\underline{V}VHH\underline{VH}VVV\underline{H}}$	0 - $\mathrm{\underline{V}HHVV\underline{V}}$	0 - HHVV

Figure 7: Some $k - VH$ sequences and their decodings

Unfortunately, the conditions (i) to (iv) are not sufficient to characterize the VH-sequences induced by $LURD$-sequences. The last two examples in figure 7 can not be the encodings of LURD-sequences with the properties of Lemma 3.

Lemma 4 *Let S be a VH-string of length n, $0 \leq k \leq \frac{n}{2} - 1$ with properties (i)-(iv) then S is a VH-k encoding of an 'LURD'-sequence with properties (i)-(iii) of Lemma 3 if and only if*

v. The k-th 'H' occurs before the $\frac{v}{2}$-th 'V'.

vi. The $k + \frac{h}{2}$-th 'H' occurs before the last 'V'.

vii. The $\frac{v}{2}$-th 'V' occurs before the $k + \frac{h}{2}$-th 'H'.

Proof: (v), (vi) and (vii) correspond to the fact that a path induced by a LURD-sequence hits the sides of the rectangle in the correct order. To prove sufficiency, we construct the path induced by the LURD-sequence. First, we draw a rectangle of lenght $\frac{v}{2}$ and width $\frac{h}{2}$ and start at the bottomline at the point with distance k from the left side. Now, we do left and up moves for each 'H' and 'V' until we hit the left boundary. Then, we switch to right and up moves etc. □

In order to generate a random polyomino of perimeter n we repeat the algorithm in figure 8:

```
Function GUESSPOLYOMINO:

(1) Choose a number $k$ in $0, \dots, \frac{n}{2}$ and a binary string of length $n-6$.
(2) Generate a VH-$k$ encoding of length $n$ using (iv)-(i).
(3) If (v),(vi) or (vii) does not hold, then return $fail$.
(4) Construct the LURD-sequence using the method in the proof of Lemma 4.
(5) If the path induced by this sequence is not simple then return $fail$ else return the
    LURD-sequence.
```

Figure 8: The main loop

Theorem 3 *i. If the function GUESSPOLYOMINO returns an LURD-sequence then this sequence encodes the boundary of a convex polyomino.*

ii. Every convex polyomino is generated with uniform probability.

iii. The function GUESSPOLYOMINO returns a polyomino with asymptotic probability 0.5.

iv. The function GUESSPOLYOMINO has linear running time.

Proof.

i. This is a direct consequence of the Lemmata 1 – 3.

ii. Fixing the perimeter $2n+8$ we select k in $0, \dots, n+4$ and a randomstring of length $2n+2$. A polyomino is output of this algorithm if and only if the correct pair was choosen at the beginning, thus every polyomino has the probability $\frac{1}{4(n+5)4^n}$.

iii. We have $P_{2n+8} = (2n+11)4^n - 4(2n+1)\binom{2n}{n}$. Using $\binom{2n}{n} \geq \frac{4^n}{n}$ for $n \geq 4$ a simple calculation leeds to a probability of less than 0.5. On the other hand, utilizing Sterling's formula we get $\binom{2n}{n} < \frac{4^n}{2\sqrt{n}}$ for large n which implies an asymptotic probability of 0.5.

iv. This is trivial.

The convergence of the probability towards 0.5 is very fast. It is thus possible to generate large random polyominoes effectively with our algorithm. However, a large random polyomino tends to look like a rectangle standing on a corner (see also [Ben74]). This is not surprising, since the interpretation of a random 0-1 sequence as a left-up walk will asymptotically yield a diagonal.

References

[Ben74] E. Bender: Convex n-ominoes *Discr. Math. 8 (1974) 219 – 226*

[Ber81] C. Berge, C. C. Chen, V. Chvatal and C. S. Seow: Combinatorial properties of polyominoes *Combinatorica 1 (1981) 217 – 224*

[Con90] J. H. Conway and J. C. Lagrias: Tiling with polyominoes and combinatorial group theory *J. Comb. Th. A53 (1990) 183 – 208*

[Del84] M. P. Delest and G. Viennot: Algebraic languages and polyominoes enumeration *Theor. Comp. Sci. 34 (1984) 169 – 206*

[Del91] M. Delest: Polyominoes and animals: Some recent results *J. Math. Chem. 8 (1991) 3 – 18*

[Gol89] S. Golomb: Polyominoes while tile rectangles *J. Comb. Th. A51 (1989) 117 – 124*

[Gutt88] A. J. Guttmann: The number of convex polygons on the square and honeycomb lattices *J. Phys. A21 (1988) 467 – 474*

[Lin88] K. Y. Lin and S. J. Chang: Rigorous results for the number of convex polygons on the square and honeycomb lattices *J. Phys. A21 (1988) 2635 – 2642*

[Mar91] G. E. Martin: Polyominoes: A Guide to Puzzles and Problems in Tiling *Mathematical Association of America Spectrum Series 1991*

[Pol69] G. Pólya: On the number of certain lattice polygons *J. Comb. Th. 6 (1969) 102 – 105*

Quantum Letter-Place Algebra

Rosa Q. Huang[1]
Math. Dept., VPI&SU, Blacksburg, VA 24061
and
James J. Zhang[2]
Math. Dept., Univ. of Michigan, Ann Arbor, MI 48109

Abstract

A quantum analogue of the supersymmetric letter-place algebra of Rota and his school is presented. Although it is significantly different than the ordinary letter-place algebra, it still possesses the most important combinatorial structure: standard quantum left (resp. right) bitableaux form a linear basis of the quantum letter-place algebra. The quantum straightening formula holds for quantum letter-place algebras.

0. Introduction

In [DKR] and [GRS], Rota and his school systematically developed the theory of the supersymmetric letter-place algebra. Although the algebraic structure of the letter-place algebra is isomorphic to the algebra generated by all minors of a generic supersymmetric matrix, the invention of the so-called letters and places distinguishes the letter-place algebra with its broad applicability and rich combinatorial structure. For this reason, this algebra has been widely applied, and proved to be an effective algebraic-combinatorial tool so far, to different areas like classical invariant theory ([GRS], [KR], [Hu1]), representation theory ([BPT], [BT]), resolutions of certain algebras and modules ([AR], [BR]), projective geometry ([RS], [Wh]), rigidity theory ([WW]), etc.

In the present paper we use Manin's approach of quantum groups to develop a quantum analogue of supersymmetric letter-place algebra. Generalizing Manin's definition of quantum general linear supergroups, we define (supersymmetric) quantum letter-place algebra $Super[L|P]_q$ by requiring that both the left co-representation T_l from the quantum letter algebra $super[L]_q$ to $Super[L|P]_q \otimes Super[P]_q$ and the right co-representation T_r from quantum place algebra $Super[P]_q$ to $Super[L]_q \otimes Super[L|P]_q$ are algebra homomorphisms. In the case that $L = P$, the super bialgebra $Super[L|L]_q$ coincides with the quantum general linear super(semi-)group E_q defined in [Ma2]. Hence $Super[L|P]_q$ can be also viewed as a supersymmetric analogue of the quantum linear semi-groups $M_n(q)$. Again, as an algebra, $Super[L|P]_q$ is isomorphic to the algebra generated by all (left or right) quantum minors of a generic quantum supersymmetric matrix. However this quantum letter-place algebra is significantly different than the ordinary letter-place algebra. For examples, one has to distinguish between "left" and "right" quantum biproducts, although it turns out that they only differ by a scalar multiple, and only left-sided (resp. right-sided) Laplace expansion holds for left (resp. right) quantum biproducts (Proposition 4). No clear relation has been

[1]The research of R. Huang is supported in part by an NSA grant.
[2]The research of J. Zhang supported in part by an NSF grant.