Verifying the extended loop braid relations for the reduced Burau representation of $\mathbf{LB}'_4$

Supplementary material for [M. Palmer, A. Soulié, *The Burau representations of loop braid groups*]

Martin Palmer and Arthur Soulié

31st January 2022

Abstract

We explicitly verify that the matrices given in Table 1 of [PS] in the case $n = 4$ satisfy all of the relations of the extended loop braid group $\mathbf{LB}'_4$, as described for example in [BH]. We note that, in [BH], braid words are written from left to right, whereas matrix multiplication is written from right to left, so we in fact verify that these matrices satisfy the *reverse* of the relations described by Brendle and Hatcher. In addition, the matrices in Table 1 of [PS] are in fact defined over the ring $S = R/(t^2 - 1) = \mathbb{Z}[t^{\pm 1}]/(t^2 - 1)$, so $t^2 = 1$ everywhere. Moreover, the bottom row of each matrix lies in $S/(t-1)$, so $t = 1$ on the bottom row of each matrix. For more details of this representation, including its topological construction, see [PS].

[BH] = T. Brendle, A. Hatcher, *Configuration spaces of rings and wickets*, Comment. Math. Helv. 88.1 (2013), pp. 131–162.
[PS] = M. Palmer, A. Soulié, *The Burau representations of loop braid groups*, to appear in Comptes Rendus Math. (2022) (see also arXiv:2109.11468)

```
In [1]: R.<t> = LaurentPolynomialRing(ZZ); R

Out[1]: Univariate Laurent Polynomial Ring in t over Integer Ring

In [2]: Ta = matrix(R, [[-1,1,0,0] , [0,1,0,0] , [0,0,1,0] , [0,0,0,1]]); Ta

Out[2]: [-1  1  0  0]
        [ 0  1  0  0]
        [ 0  0  1  0]
        [ 0  0  0  1]

In [3]: Tb = matrix(R, [[1,0,0,0] , [1,-1,1,0] , [0,0,1,0] , [0,0,0,1]]); Tb

Out[3]: [ 1  0  0  0]
        [ 1 -1  1  0]
        [ 0  0  1  0]
        [ 0  0  0  1]

In [4]: Tc = matrix(R, [[1,0,0,0] , [0,1,0,0] , [0,1,-1,-1-t] , [0,0,0,1]]); Tc

Out[4]: [     1      0      0      0]
        [     0      1      0      0]
        [     0      1     -1 -1 - t]
        [     0      0      0      1]
```

```
In [5]: Sa = matrix(R, [[-t,1,0,0] , [0,1,0,0] , [0,0,1,0] , [0,0,0,1]]); Sa

Out[5]: [-t  1  0  0]
        [ 0  1  0  0]
        [ 0  0  1  0]
        [ 0  0  0  1]

In [6]: Sb = matrix(R, [[1,0,0,0] , [t,-t,1,0] , [0,0,1,0] , [0,0,0,1]]); Sb

Out[6]: [ 1  0  0  0]
        [ t -t  1  0]
        [ 0  0  1  0]
        [ 0  0  0  1]

In [7]: Sc = matrix(R, [[1,0,0,0] , [0,1,0,0] , [0,t,-t,-1-t] , [0,0,0,1]]); Sc

Out[7]: [     1      0      0      0]
        [     0      1      0      0]
        [     0      t     -t -1 - t]
        [     0      0      0      1]

In [8]: Da = matrix(R, [[-t,0,0,0] , [-1-t,1,0,0] , [-1-t,0,1,0] , [1,0,0,1]]); Da

Out[8]: [    -t      0      0      0]
        [-1 - t      1      0      0]
        [-1 - t      0      1      0]
        [     1      0      0      1]

In [9]: Db = matrix(R, [[1,0,0,0] , [1+t,-t,0,0] , [1+t,-1-t,1,0] , [-1,1,0,1]]); Db

Out[9]: [     1      0      0      0]
        [ 1 + t     -t      0      0]
        [ 1 + t -1 - t      1      0]
        [    -1      1      0      1]

In [10]: Dc = matrix(R, [[1,0,0,0] , [0,1,0,0] , [0,1+t,-t,0] , [0,-1,1,1]]); Dc

Out[10]: [    1     0     0     0]
         [    0     1     0     0]
         [    0 1 + t    -t     0]
         [    0    -1     1     1]

In [11]: Dd = matrix(R, [[1,0,0,0] , [0,1,0,0] , [0,0,1,0] , [0,0,-1,-1]]); Dd

Out[11]: [ 1  0  0  0]
         [ 0  1  0  0]
         [ 0  0  1  0]
         [ 0  0 -1 -1]

In [12]: Sai = Sa.inverse(); Sai

Out[12]: [ 1/-t -1/-t     0     0]
         [    0     1     0     0]
         [    0     0     1     0]
         [    0     0     0     1]
```

```
In [13]: Sbi = Sb.inverse(); Sbi

Out[13]: [     1      0      0      0]
         [     1   1/-t  -1/-t      0]
         [     0      0      1      0]
         [     0      0      0      1]

In [14]: Sci = Sc.inverse(); Sci

Out[14]: [          1           0           0           0]
         [          0           1           0           0]
         [          0           1        1/-t  (t + 1)/-t]
         [          0           0           0           1]

In [15]: I = matrix(R, [[1,0,0,0] , [0,1,0,0] , [0,0,1,0] , [0,0,0,1]]); I

Out[15]: [1 0 0 0]
         [0 1 0 0]
         [0 0 1 0]
         [0 0 0 1]

In [16]: Da^2 - I

Out[16]: [-1 + t^2         0         0         0]
         [-1 + t^2         0         0         0]
         [-1 + t^2         0         0         0]
         [   1 - t         0         0         0]

In [17]: Db^2 - I

Out[17]: [       0         0         0         0]
         [ 1 - t^2  -1 + t^2        0         0]
         [ 1 - t^2  -1 + t^2        0         0]
         [  -1 + t     1 - t        0         0]

In [18]: Dc^2 - I

Out[18]: [       0         0         0         0]
         [       0         0         0         0]
         [       0   1 - t^2  -1 + t^2        0]
         [       0    -1 + t     1 - t        0]

In [19]: Dd^2 - I

Out[19]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [20]: Ta^2 - I

Out[20]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
```

```
In [21]: Tb^2 - I

Out[21]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [22]: Tc^2 - I

Out[22]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [23]: Sa * Sb * Sa - Sb * Sa * Sb

Out[23]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [24]: Sb * Sc * Sb - Sc * Sb * Sc

Out[24]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [25]: Ta * Tb * Ta - Tb * Ta * Tb

Out[25]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [26]: Tb * Tc * Tb - Tc * Tb * Tc

Out[26]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [27]: Da * Db - Db * Da

Out[27]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [28]: Da * Dc - Dc * Da

Out[28]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
```

```
In [29]: Da * Dd - Dd * Da

Out[29]: [    0      0      0      0]
         [    0      0      0      0]
         [    0      0      0      0]
         [1 - t      0      0      0]

In [30]: Db * Dc - Dc * Db

Out[30]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [31]: Db * Dd - Dd * Db

Out[31]: [     0       0       0       0]
         [     0       0       0       0]
         [     0       0       0       0]
         [-1 + t   1 - t       0       0]

In [32]: Dc * Dd - Dd * Dc

Out[32]: [     0       0       0       0]
         [     0       0       0       0]
         [     0       0       0       0]
         [     0  -1 + t   1 - t       0]

In [33]: Sa * Dc - Dc * Sa

Out[33]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [34]: Sa * Dd - Dd * Sa

Out[34]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [35]: Sb * Dd - Dd * Sb

Out[35]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [36]: Sc * Da - Da * Sc

Out[36]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
```

```
In [37]: Ta * Dc - Dc * Ta

Out[37]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [38]: Ta * Dd - Dd * Ta

Out[38]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [39]: Tb * Dd - Dd * Tb

Out[39]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [40]: Tc * Da - Da * Tc

Out[40]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [41]: Ta * Sc - Sc * Ta

Out[41]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [42]: Tc * Sa - Sa * Tc

Out[42]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [43]: Ta * Da - Db * Ta

Out[43]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [44]: Tb * Db - Dc * Tb

Out[44]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
```

```
In [45]: Tc * Dc - Dd * Tc

Out[45]: [    0     0     0     0]
         [    0     0     0     0]
         [    0     0     0     0]
         [    0     0     0 1 - t]

In [46]: Sa * Da - Db * Sa

Out[46]: [-1 + t^2        0        0        0]
         [-1 + t^2        0        0        0]
         [-1 + t^2        0        0        0]
         [   1 - t        0        0        0]

In [47]: Sb * Db - Dc * Sb

Out[47]: [       0        0        0        0]
         [ 1 - t^2 -1 + t^2        0        0]
         [ 1 - t^2 -1 + t^2        0        0]
         [  -1 + t    1 - t        0        0]

In [48]: Sc * Dc - Dd * Sc

Out[48]: [       0        0        0        0]
         [       0        0        0        0]
         [       0  1 - t^2 -1 + t^2        0]
         [       0   -1 + t    1 - t    1 - t]

In [49]: Sa * Db - Da * Ta * Sai * Ta

Out[49]: [            0             0             0             0]
         [(-t^2 + 1)/-t             0             0             0]
         [(-t^2 + 1)/-t             0             0             0]
         [   (t - 1)/-t             0             0             0]

In [50]: Sb * Dc - Db * Tb * Sbi * Tb

Out[50]: [            0             0             0             0]
         [            0             0             0             0]
         [ (t^2 - 1)/-t (-t^2 + 1)/-t             0             0]
         [  (-t + 1)/-t    (t - 1)/-t             0             0]

In [51]: Sc * Dd - Dc * Tc * Sci * Tc

Out[51]: [           0            0            0            0]
         [           0            0            0            0]
         [           0            0            0     -t^2 + 1]
         [           0 (-t + 1)/-t   (t - 1)/-t        t - 1]

In [52]: Sb * Ta * Tb - Ta * Tb * Sa

Out[52]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
```

```
In [53]: Sc * Tb * Tc - Tb * Tc * Sb

Out[53]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [54]: Tb * Sa * Sb - Sa * Sb * Ta

Out[54]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]

In [55]: Tc * Sb * Sc - Sb * Sc * Tb

Out[55]: [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
         [0 0 0 0]
```