3. Some Computations

Pensieve header: The primary program accompanying “Over then Under Tangles”, by Dror Bar-Natan, Zsuzsanna Dancso, and Roland van der Veen.

This notebook also generates the LaTeX file SomeComputations.tex, that corresponds to the section “Some Computations” in the paper.

Mathematica and LaTeX Initialization

In[•]:=
```
SetDirectory["C:\\drorbn\\AcademicPensieve\\Projects\\OU"]
```

Out[•]=
```
C:\drorbn\AcademicPensieve\Projects\OU
```

tex
```
{
\def\face{\input{figs/face.pdf_t}}
\def\human{\input{figs/human.pdf_t}}
\def\machine{\input{figs/machine.pdf_t}}

\def\cellscale{1}
\newsavebox\pdfbox
\newlength{\pdfheight}

\def\nbpdfInput#1{{%
 \savebox{\pdfbox}{\includegraphics[scale=\cellscale]{#1}}%
 \settoheight{\pdfheight}{\usebox{\pdfbox}}%
 %\uselengthunit{mm}\printlength{\pdfheight}%
 \noindent\imagetop{\ifdim\pdfheight<10mm\face\else\human\fi}\ %
 \imagetop{\usebox{\pdfbox}}%
 \vskip 2mm%
}}

\def\nbpdfOutput#1{{\noindent{\imagetop{\machine}\
\imagetop{\includegraphics[scale=\cellscale]{#1}}\vskip 2mm}}}

\def\m#1{\text{\tt #1}}
```

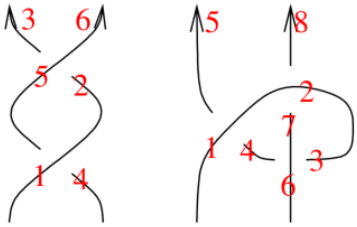
Section Introduction

tex
```
\section{Some Computations} \label{sec:comp}
```

When mathematics is computable, we feel it is appropriate and necessary to include an implementation. In this case, the implementation is concise and follows the notation and logical structure of a paper, so we choose to include it as an integral part of that paper. We hope that the programs presented here serve as an illustration of the overall simplicity and validity of the ideas within the paper, and that they encourage others to play and further discover. The implementations also lead to new enumerations -- the tables in Sections~\ref{ssec:VB}~and~\ref{ssec:CB} -- and to intriguing and appealing graph-valued invariants -- Section~\ref{ssec:EG} -- which cannot be computed otherwise, and which may lead to further study.

All code here is written in {\sl Mathematica}~\cite{Wolfram:Mathematica} and is available as the {\sl Mathematica} notebook {\sl SomeComputations.nb} at~\cite{Self}.

Virtual Diagrams



tex

\parpic[r]{\input{figs/VDExample.pdf_t}}
\subsection{Implementing virtual OU tangles, virtual braids, and $\Ch$}
To represent a virtual tangle diagram D on the computer, we order its strands and traverse each of them in order, marking each ``O'' point, each ``U'' point, and each end of strand, with the integers $1,2,3,\ldots$, in the order in which they are encountered. See examples on the right. For each crossing x of D we form a {\sl Mathematica} expression $\m{X}_s[i,j]$, where s is the sign of the crossing and i and j are the markings next to the O side and the U side of x, respectively. We also form an expression $\m{EOS}[k]$ for each end-of-strand marked k. We toss all this information into a container \m{VD}, and the result is our computer representation of D. Below, \m{vd1} and \m{vd2} are the results of this process for the two example tangles shown here.

pdf

In[•]:=
```
SetAttributes[VD, Orderless]
```

pdf

In[•]:=
```
vd1 = VD[X+1[1, 4], X+1[5, 2], EOS[3], EOS[6]];
vd2 = VD[X+1[1, 4], X+1[2, 7], X+1[6, 3], EOS[5], EOS[8]];
```

tex

Sometimes in a \m{VD} we allow to label O/U/\m{EOS} points by arbitrary real numbers, for in fact, only the ordering of these points matter. The routine \m{Tidy} takes a real-ordered \m{VD} and converts it to a sequentially ordered one. Thus it brings a \m{VD} to a ``canonical form'':

pdf

```
In[•]:= Tidy[vd_VD] := Module[{ps = Union @@ (List @@@ vd)},
          Replace[vd, Thread[ps → Range@Length@ps], {2}] ]
```

pdf

```
In[•]:= VD[X+1[0.9, 4.2], X+1[5, e], EOS[π], EOS[60]] // Tidy
```

pdf

```
Out[•]= VD[EOS[4], EOS[6], X1[1, 2], X1[3, 5]]
```

tex

The routine \m{R12Reduce1} reduces a virtual diagram by performing one R2 or R1 move, if such a move is available, and otherwise it does nothing. The routine \m{R12Reduce} finds the fixed point of \m{R12Reduce1} --- in other words, it reduces a virtual diagram using all available R1 and R2 moves.

pdf

```
In[•]:= R12Reduce1[vd_VD] := Tidy@Module[{R2s, R2}, Which[
          Length[R2s = Cases[vd, Xs_[i_, j_] :> X-s[i + 1, j + 1]] ⋂ (List @@ vd)] > 0,
          Complement[vd, VD[R2 = First@R2s, R2 /. Xs_[i_, j_] :> X-s[i - 1, j - 1]]],
          Length[R2s = Cases[vd, Xs_[i_, j_] :> X-s[i + 1, j - 1]] ⋂ (List @@ vd)] > 0,
          Complement[vd, VD[R2 = First@R2s, R2 /. Xs_[i_, j_] :> X-s[i - 1, j + 1]]],
          True, DeleteCases[vd, X_[i_, j_] /; Abs[i - j] == 1] ]];
        R12Reduce[vd_VD] := FixedPoint[R12Reduce1, vd]
```

tex

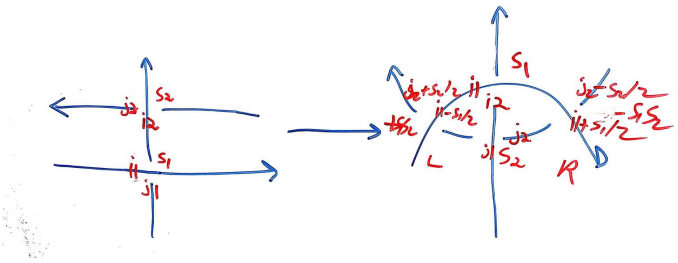
Here's a very minor example:

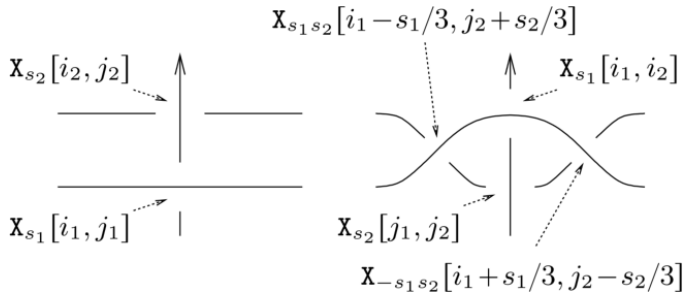
pdf

```
In[•]:= VD[X+1[1, 4], X-1[2, 5], EOS[3], EOS[6]] // R12Reduce
```

pdf

```
Out[•]= VD[EOS[1], EOS[2]]
```





tex

\Needspace{30mm} % 29mm is not enough.
\parpic[r]{

\def\Xa{$\m{X}_{s_1}[i_1,j_1]$} \def\Xb{$\m{X}_{s_2}[i_2,j_2]$}
\def\Xc{$\m{X}_{s_2}[j_1,j_2]$} \def\Xd{$\m{X}_{s_1}[i_1,i_2]$}
\def\Xe{$\m{X}_{s_1s_2}[i_1\!-\!s_1/3,j_2\!+\!s_2/3]$} \def\Xf{$\m{X}_{-s_1s_2}[i_1\!+\!s_1/3,j_2\!-\!s_2/3]$}
\input{figs/Gamma1.pdf_t}

In a similar manner, \m{Γ1} performs one glide move if one is available, and \m{$\barGamma$} fully reduces under both glide moves and R1 and R2 moves. Here we bound the number of iterations by 2^{24}, to artificially stop runaway reductions such as the one in Figure~\ref{fig:swirls}.

pdf

In[•]:=
```
Γ1[vd_VD] := Module[{js, s1, i1, j1, s2, i2, j2},
  js = Cases[vd, X_[_, j_] :> j] ⋂ Cases[vd, X_[i_, _] :> i - 1];
  If[Length[js] == 0, vd,
   j1 = RandomChoice[js]; i2 = j1 + 1;
   Cases[vd, X_s_[i_, j1] :> (s1 = s; i1 = i)];
   Cases[vd, X_s_[i2, j_] :> (s2 = s; j2 = j)];
   Tidy@Join[Complement[vd, VD[X_s1[i1, j1], X_s2[i2, j2]]],
     VD[X_s2[j1, j2], X_s1[i1, i2], X_s1s2[i1 - s1 / 3, j2 + s2 / 3], X_-s1s2[i1 + s1 / 3, j2 - s2 / 3]]
    ]]]
```

In[•]:=
```
Γ[vd_VD] := FixedPoint[Γ1, vd, 2^24]
```

In[•]:=
```
Γ[T_] /; Head[T] =!= VD := Γ[VD[T]]
```

pdf

In[•]:=
```
Γ̄[vd_VD] := FixedPoint[Γ1@*R12Reduce, vd, 2^24];
Γ̄[T_] /; Head[T] =!= VD := Γ̄[VD[T]]
```

tex

As expected, $\barGamma(\m{vd1})=\m{vd2}$:

pdf

In[•]:=
```
Γ̄[vd1] == vd2
```

pdf

Out[•]=
```
True
```

tex

Next we define the composition operation \m{d1**d2} of virtual tangle diagrams. The implementation works by ``shrinking'' \m{d2} so that each of its strands would fit between the last crossing in the corresponding strand of \m{d1} and the \m{EOS} at the end of that strand of \m{d1}, then taking the union of \m{d1} and the shrank \m{d2}, and then applying \m{Tidy} to the result:

pdf

```
In[•]:= VD /: d1_VD ** d2_VD := Tidy@Module[{es1, es2, m2},
           es1 = Cases[d1, EOS[i_] :> i];
           m2 = Max[es2 = Cases[d2, EOS[i_] :> i]];
           d1 ⋃
            Replace[DeleteCases[d2, _EOS], i_ :> i / m2 - 1 + es1[[1 + Count[es2, e_ /; i > e]]], {2}]]
```

tex

For example, ``our'' \m{vd2} has 3 crossings yet is equivalent to a 2-twist braid. So $\m{vd1}\cdot\m{vd2}$ ought to have 6 crossings while its reduced OU form, $\barGamma(\m{vd1}\cdot\m{vd2})$ should be the Cinnamon Roll $\CR_4$, which has 7 crossings. The computer agrees:

pdf

```
In[•]:= {vd2 ** vd2, Γ̄[vd2 ** vd2]}
```

pdf

```
Out[•]= {VD[EOS[9], EOS[14], X1[1, 4], X1[2, 11], X1[5, 8], X1[6, 13], X1[10, 3], X1[12, 7]], VD[
          EOS[9], EOS[16], X1[1, 8], X1[2, 15], X1[3, 6], X1[4, 13], X1[10, 7], X1[11, 14], X1[12, 5]]}
```

Virtual Pure Braids

tex

Next we implement virtual pure braids, and it is best to start with an example. We represent the 3-strand virtual pure braid $\beta=\sigma_{21}^{-1}\sigma_{13}\sigma_{31}\sigma_{13}\sigma_{31}\sigma_{13}\sigma_{23}\sigma_{21}$ of Example~\ref{exa:SCR} by the {\sl Mathematica} expression below:

pdf

```
In[•]:= β = VPB[3, σ̄2,1, σ1,3, σ3,1, σ1,3, σ3,1, σ1,3, σ2,3, σ2,1];
```

tex

The conversion of \m{VPB}s into \m{VD}s is quite easy. We just need to define it on the generators and then use the already-available composition of \m{VD}s to extend the definition to products of generators:

pdf

```
In[•]:= VPB[n_] // VD := VD @@ (EOS /@ Range[n]);
        VPB[n_, σi_,j_] // VD := Tidy@Append[VD @@ (EOS /@ Range[n]), X+1[i - 0.5, j - 0.5]];
        VPB[n_, σ̄i_,j_] // VD := Tidy@Append[VD @@ (EOS /@ Range[n]), X-1[i - 0.5, j - 0.5]];
        VPB[n_, σ_, σs__] // VD := VD[VPB[n, σ]] ** VD[VPB[n, σs]]
```

tex

We can compute $\Ch(\beta)=\barGamma_v(\bariota_v(\beta))$ (count that it has 18 \m{X} symbols, just as Figure~\ref{fig:SCR}~(A) has 18 crossings!):

pdf

```
In[•]:= β // VD // Γ̄
```

pdf

```
Out[•]= VD[EOS[14], EOS[24], EOS[39], X-1[20, 6], X-1[21, 31], X-1[22, 10], X-1[23, 35],
         X1[1, 38], X1[2, 13], X1[3, 34], X1[4, 9], X1[5, 30], X1[15, 37], X1[16, 12],
         X1[17, 33], X1[18, 8], X1[19, 29], X1[25, 36], X1[26, 11], X1[27, 32], X1[28, 7]]
```

tex

We can even verify Equation~\eqref{eq:SCRBraids1}:

pdf

In[•]:=
```
(β // VD // Γ̄) == (VPB[3, σ2,3, σ1,3, σ3,1, σ1,3, σ3,1, σ1,3] // VD // Γ̄)
```

pdf

Out[•]= True

Tabulating Virtual Pure Braids

tex

\subsection{Tabulating Virtual Pure Braids} \label{ssec:VB}
Our next task is to tabulate virtual pure braids with a given number of strands n and a bound m on the number of crossings. The first routine, \m{VPBGens}, outputs the list of all generators of $\vcalPB_n$:

pdf

In[•]:=
```
A_ \ B_ := Complement[A, B];
VPBGens[n_] := VPBGens[n] = Flatten@Table[{σi,j, σ̄i,j}, {i, n}, {j, Range[n] \ {i}}];
```

pdf

In[•]:=
```
VPBGens[3]
```

pdf

Out[•]= $\{\sigma_{1,2}, \overline{\sigma}_{1,2}, \sigma_{1,3}, \overline{\sigma}_{1,3}, \sigma_{2,1}, \overline{\sigma}_{2,1}, \sigma_{2,3}, \overline{\sigma}_{2,3}, \sigma_{3,1}, \overline{\sigma}_{3,1}, \sigma_{3,2}, \overline{\sigma}_{3,2}\}$

tex

Next we'd like to generate all words in the generators we just computed, and separate them using $\Ch$ and Chterental's Theorem (\ref{thm:vinj}). To save some computer effort, we generate only ``proud'' words --- words that do not contain a letter followed by its inverse, or adjacent commuting letters that are not in lexicographic order. The ``Proud Followers'' \m{PF} of a generator are those generators that can follow it without ruining the pride of a word:

pdf

In[•]:=
```
PF[n_, σi_,j_] := PF[n, σi,j] = Module[{p, q, s},
    Flatten@{σi,j, σj,i, σ̄j,i,
      Table[{σp,q, σq,p, σ̄p,q, σ̄q,p}, {p, {i, j}}, {q, Range[n] \ {i, j}}],
      Table[{σp,q, σ̄p,q}, {p, Range[i + 1, n] \ {j}}, {q, Range[n] \ {i, j, p}}]}];
PF[n_, σ̄i_,j_] := PF[n, σ̄i,j] = PF[n, σi,j] /. σi,j → σ̄i,j
```

pdf

In[•]:=
```
PF[4, σ2,3]
```

pdf

Out[•]= $\{\sigma_{2,3}, \sigma_{3,2}, \overline{\sigma}_{3,2}, \sigma_{2,1}, \sigma_{1,2}, \overline{\sigma}_{2,1}, \overline{\sigma}_{1,2}, \sigma_{2,4}, \sigma_{4,2},$
$\overline{\sigma}_{2,4}, \overline{\sigma}_{4,2}, \sigma_{3,1}, \sigma_{1,3}, \overline{\sigma}_{3,1}, \overline{\sigma}_{1,3}, \sigma_{3,4}, \sigma_{4,3}, \overline{\sigma}_{3,4}, \overline{\sigma}_{4,3}, \sigma_{4,1}, \overline{\sigma}_{4,1}\}$

tex

And then $\m{PVPBDs}[n,m]$ computes all Proud Virtual Pure Braid Diagrams on n strands and with m crossings:

pdf

```
In[•]:= PVPBDs[n_, 0] := {VPB[n]};
        PVPBDs[n_, 1] := VPB[n, #] & /@ VPBGens[n];
        PVPBDs[n_, m_] :=
         Flatten[PVPBDs[n, m - 1] /. VPB[n, σs___, σ_] :> (VPB[n, σs, σ, #] & /@ PF[n, σ])]
```

pdf

```
In[•]:= PVPBDs[2, 2]
```

pdf

```
Out[•]= {VPB[2, σ1,2, σ1,2], VPB[2, σ1,2, σ2,1], VPB[2, σ1,2, σ̄2,1], VPB[2, σ̄1,2, σ̄1,2],
         VPB[2, σ̄1,2, σ2,1], VPB[2, σ̄1,2, σ̄2,1], VPB[2, σ2,1, σ2,1], VPB[2, σ2,1, σ1,2],
         VPB[2, σ2,1, σ̄1,2], VPB[2, σ̄2,1, σ̄2,1], VPB[2, σ̄2,1, σ1,2], VPB[2, σ̄2,1, σ̄1,2]}
```

tex

These sets grow very rapidly:

pdf

```
In[•]:= PVPBDs[4, 4] // Length
```

pdf

```
Out[•]= 219 560
```

tex

$\m {AllVPBs}[n, m] $ finds representatives for all virtual braids on n strands with at most m crossings, by using $\m{PVPBDs}[n,m]$ and then deleting duplicates by $\barGamma_v$:

pdf

```
In[•]:= AllVPBs[n_, m_] := DeleteDuplicatesBy[r̄]@Flatten@Table[b, {k, 0, m}, {b, PVPBDs[n, k]}]
```

Here are the 17 2-strand virtual pure braids with up to 2 crossings:

pdf

```
In[•]:= AllVPBs[2, 2]
```

pdf

```
Out[•]= {VPB[2], VPB[2, σ1,2], VPB[2, σ̄1,2], VPB[2, σ2,1], VPB[2, σ̄2,1],
         VPB[2, σ1,2, σ1,2], VPB[2, σ1,2, σ2,1], VPB[2, σ1,2, σ̄2,1], VPB[2, σ̄1,2, σ̄1,2],
         VPB[2, σ̄1,2, σ2,1], VPB[2, σ̄1,2, σ̄2,1], VPB[2, σ2,1, σ2,1], VPB[2, σ2,1, σ1,2],
         VPB[2, σ2,1, σ̄1,2], VPB[2, σ̄2,1, σ̄2,1], VPB[2, σ̄2,1, σ1,2], VPB[2, σ̄2,1, σ̄1,2]}
```

tex

There are 15,156 virtual pure braids with 3 strands and precisely 4 crossings (meaning, braids in $\m {AllVPBs}[3,4] $ but excluding those in $\m {AllVPBs}[3,3]$). It took our computer about 86 seconds to figure that out:

pdf

```
In[•]:= Length@AllVPBs[3, 4] - Length@AllVPBs[3, 3] // Timing
```

pdf

```
Out[•]= {85.9844, 15 156}
```

tex

\Needspace{65mm} % 64mm is not enough.
\parpic[r]{\setlength{\tabcolsep}{3pt}\begin{tabular}{|c|c|c|c|c|c|}
\hline
$m\backslash n$ & 2 & 3 & 4 & 5 & 6 \\
\hline

```
0 & 1 & 1 & 1 & 1 & 1 \\
1 & 4 & 12 & 24 & 40 & 60 \\
2 & 12 & 132 & 504 & 1,320 & 2,820 \\
3 & 36 & 1,416 & 10,344 & 41,760 & 124,140 \\
4 & 108 & 15,156 & 211,416 & 1,308,360 & 5,357,700 \\
5 & 324 & 162,156 & 4,317,912 & & \\
6 & 972 & 1,734,864 & & & \\
\hline
\end{tabular}}
```

tex

In our spare time we have tabulated the numbers of n-strand pure virtual braids with precisely m crossings for some small values of n and m. The results are on the right, and data files containing the actual braids are at~\cite{Self}. As a test of the integrity of our programs we also computed most of the numbers in this table by generating all braid words and reducing modulo all relations\footnotemark. The numbers match. See the {\sl Mathematica} notebook {\sl VPBByGensAndRels.nb} at~\cite{Self}.
%
\footnotetext{Sometimes two braid words of length m_1 are related by a chain of relations that pass through words of length m_2, where $m_2>m_1$, and we do not know in advance a bound on m_2. Hence the computation using generators and relations is slow (as we have to raise m_2 and the number of words to consider grows very big) and unreliable (strictly speaking, we only get upper bounds on the braid counts).}

Tabulating Classical Braids

tex

```
\subsection{Tabulating Classical Braids} \label{ssec:CB}
```

It is a bit odd that we have not seen a table such as the one above, but for classical braids. As the classical braid group is automatic~\cite{Epstein:WordProcessing} and hence the word problem in it is very easy, there are much better in-theory tools than ours to produce such a table. Yet our tools are implemented in practice, and we may as well use them.

First, we need to be able to convert from a standard classical braid notation~\cite{Bar-NatanMorrison:KnotTheory} to the \m{VPB} notation used here.

pdf

In[ ]:=

```
VPB[BR[n_, is_List]] := VPB[n, Module[{π = Range@n, i}, Sequence @@ Table[
      If[i > 0,
       π⟦{i, i + 1}⟧ = π⟦{i + 1, i}⟧; σ_{π⟦i+1⟧,π⟦i⟧},
       (* else *) π⟦{-i, -i + 1}⟧ = π⟦{-i + 1, -i}⟧; σ̄_{π⟦-i⟧,π⟦-i+1⟧} ],
      {i, is}] ]];
VD[br_BR] := VD[VPB@br]
```

pdf

In[•]:=
```
BR[3, {1, 2, 1}] // VPB
```

pdf

Out[•]=
$$\text{VPB}[3, \sigma_{1,2}, \sigma_{1,3}, \sigma_{2,3}]$$

tex

After that, we repeat the same steps as in the virtual case:

pdf

In[•]:=
```
PF[n_, i_Integer] :=
  (Range[Max[Abs[i] - 1, 1], n - 1] ⋃ (-Range[Max[Abs[i] - 1, 1], n - 1])) \ {-i};
```

pdf

In[•]:=
```
PF[7, -4]
```

pdf

Out[•]=
```
{-6, -5, -4, -3, 3, 5, 6}
```

pdf

In[•]:=
```
ProdBs[n_, 0] := {BR[n, {}]};
ProdBs[n_, 1] := BR[n, {#}] & /@ (Range[n - 1] ⋃ (-Range[n - 1]));
ProdBs[n_, m_] /; m > 1 :=
 Flatten[ProdBs[n, m - 1] /. BR[n, {σs___, σ_}] :> (BR[n, {σs, σ, #}] & /@ PF[n, σ])]
```

pdf

In[•]:=
```
AllBs[n_, m_] := DeleteDuplicatesBy[Γ̄]@Flatten@Table[b, {k, 0, m}, {b, ProdBs[n, k]}]
```

tex

For example, here are all the distinct positive 3-strand braids:

pdf

In[•]:=
```
PositiveQ[BR[_, σs_]] := And @@ (# > 0 & /@ σs);
Select[AllBs[3, 3], PositiveQ]
```

pdf

Out[•]=
```
{BR[3, {}], BR[3, {1}], BR[3, {2}], BR[3, {1, 1}], BR[3, {1, 2}],
 BR[3, {2, 1}], BR[3, {2, 2}], BR[3, {1, 1, 1}], BR[3, {1, 1, 2}], BR[3, {1, 2, 1}],
 BR[3, {1, 2, 2}], BR[3, {2, 1, 1}], BR[3, {2, 2, 1}], BR[3, {2, 2, 2}]}
```

tex

On our computer, it takes about 20 seconds to find that there are 1,110 classical braids with 4 strands and crossing number equal to 5:

pdf

In[•]:=
```
Length@AllBs[4, 5] - Length@AllBs[4, 4] // Timing
```

pdf

Out[•]=
```
{20.1875, 1110}
```

tex

\parpic[r]{\setlength{\tabcolsep}{3pt}\begin{tabular}{|c|c|c|c|c|c|}
\hline
$m\backslash n$ & 2 & 3 & 4 & 5 & 6 \\
\hline
0 & 1 & 1 & 1 & 1 & 1 \\
1 & 2 & 4 & 6 & 8 & 10 \\
2 & 2 & 12 & 26 & 44 & 66 \\

```
3  &  2 &  30 &    98 &    206 &    362  \\
4  &  2 &  68 &   338 &    884 &  1,794  \\
5  &  2 & 148 & 1,110 &  3,600 &  8,370  \\
6  &  2 & 314 & 3,542 & 14,198 & 37,606  \\
7  &  2 & 656 & 11,098 & 54,876 & 164,910 \\
8  &  2 & 1,356 & 34,362 & 209,348 & 711,746 \\
9  &  2 & 2,782 & 105,546 & 791,798 & 3,039,546 \\
\hline
\end{tabular}}
```

And here's a table of the numbers of n-strand pure virtual braids with precisely m crossings, for small values of n and m. The data files containing the actual braids are at~\cite{Self}.

Note that the entries in the $n=3$ column of this table fit with the sequence $6\cdot 2^m-2F_{m+3}-2$, where F_m is the mth Fibonacci number:

exec

```
nb2tex$PDFWidth-=2.8;
```

pdf

In[•]:=
```
Table[6 × 2^m - 2 Fibonacci[m + 3] - 2, {m, 15}]
```

pdf

Out[•]=
```
{4, 12, 30, 68, 148, 314, 656, 1356, 2782, 5676, 11532, 23354, 47176, 95108, 191438}
```

exec

```
nb2tex$PDFWidth+=2.8;
```

tex

```
\picskip{0}
```

The fit persists at least up to $m=12$. We do not know why this is so.

Extraction Graphs

tex

```
\subsection{Extraction Graphs} \label{ssec:EG}
```

We can now write a short program \m{EG}, to compute and display Extraction Graphs as in Discussion~\ref{disc:EG}.

pdf

In[•]:=
```
Options[EG] =
  {Labels → False, GraphLayout → "SpringElectricalEmbedding", EdgeStyle → Automatic};
```

pdf

```
In[•]:= EG[𝒪_, opts___] := Module[{vd, n, gs, vs, es = {}, e, p = 0, m1, m2, g, q, k, lbl},
  lbl = Labels /. {opts} /. Options[EG];
  gs = VPBGens[n = Count[vd = Γ̄[𝒪], _EOS]]; vs = {vd};
  While[p < Length[vs], m1 = Count[vd = vs[[++p]], X_[_, _]];
   Do[m2 = Count[q = Γ̄[VD[VPB[n, g /. {σ → σ̄, σ̄ → σ}]] ** vd], X_[_, _]];
    If[m2 < m1, If[! MemberQ[vs, q], AppendTo[vs, q]];
     e = p ↔ Position[vs, q][[1, 1]];
     AppendTo[es, If[lbl, Labeled[e, g], e] ] ],
    {g, gs} ] ];
  Graph[Table[If[lbl, Labeled[k, Length[vs[[k]]] - n], k], {k, p}], es,
   FilterRules[Join[{opts}, Options[EG]], Options[Graph]]] ]
```

tex

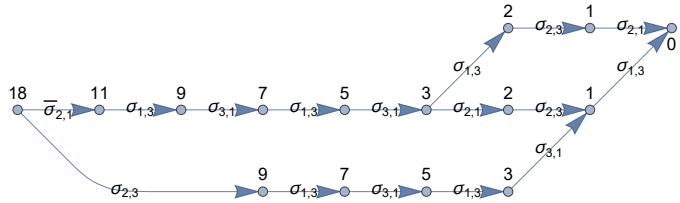
Note that the diamond in Figure~\ref{fig:SCR} is genuine, but it is not an extraction graph, because the full extraction graph of the initial OU tangle of that figure contains two further edges:

pdf

```
In[•]:= EG[VPB[3, σ2,3, σ1,3, σ3,1, σ1,3, σ3,1, σ1,3], Labels → True,
  GraphLayout → {"LayeredDigraphEmbedding", "Orientation" → Left}]
```

pdf

Out[•]=



tex

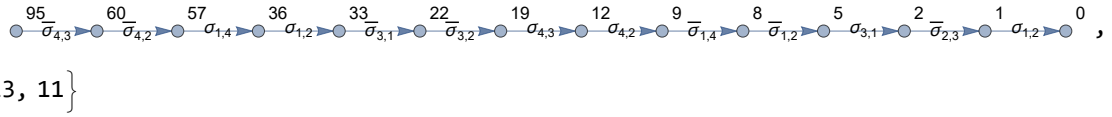
The braid below, suggested to us by B.~Wiest, has a linear extraction graph and hence a unique ``special word'' (see Discussion~\ref{disc:EG}), but that word is of length 13, whereas the braid can be presented by a shorter word β, of length 11:

pdf

```
In[•]:= β = BR[4, {-2, -3, -2, 1, -2, 1, -2, 1, 3, -2, 1}];
{g = EG[β, Labels → True, GraphLayout → Automatic, ImageSize → Large],
 VertexCount@g - 1, Length@β[[2]]}
```

pdf

Out[•]=



13, 11}

tex

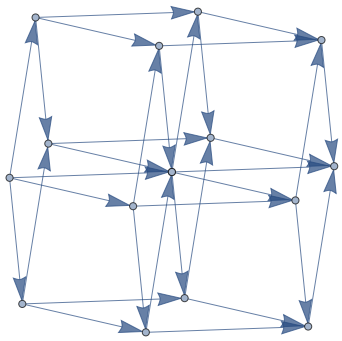
It is easy to see that the extraction graph of the 4-crossing 8-strand braid $\overcrossing{\overcrossing{\overcrossing{\overcrossing}}}$ is the tesseract:

pdf

In[•]:=
```
EG[BR[8, {1, 3, 5, 7}], GraphLayout → "HighDimensionalEmbedding", ImageSize → Small]
```

pdf

Out[•]=



tex

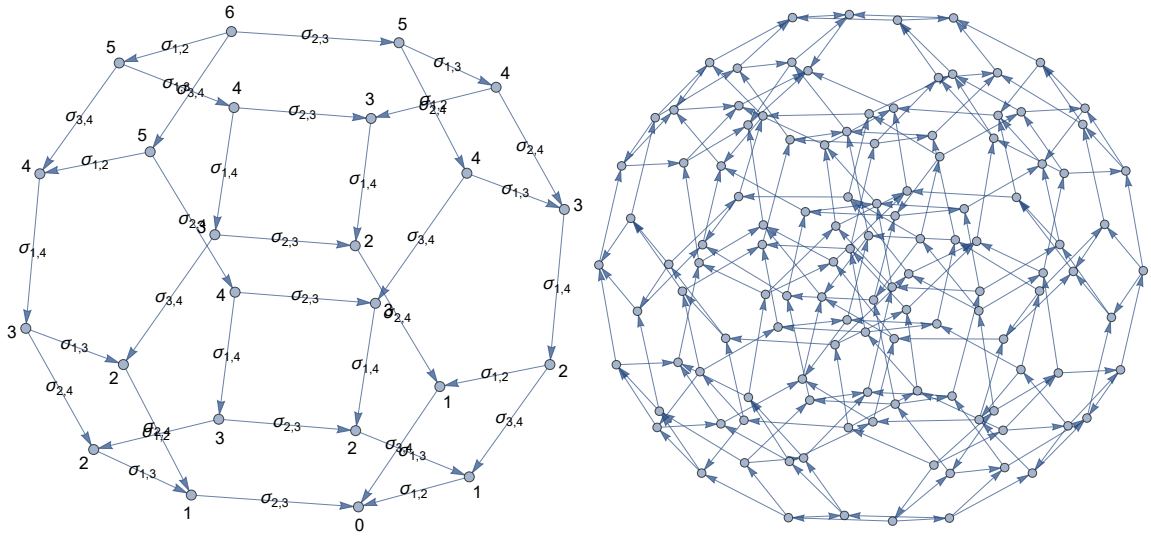
The extraction graphs of Garside braids seem to be permutahedra (we did not attempt to prove this in general):

pdf

In[•]:=
```
Row[{EG[BR[4, {1, 2, 3, 1, 2, 1}], Labels → True, ImageSize → 300],
  EG[BR[5, {1, 2, 3, 4, 1, 2, 3, 1, 2, 1}], ImageSize → 300] }]
```

pdf

Out[•]=



tex

Sometimes extraction graphs can be amusing. In no particular order, here are a lifesaver, an impressionistic map of the US state of Iowa, a torch flame, a legless bird, a feather, a ladder, a tennis racket, and a mouse trap:

pdf

In[•]:=
```
SetOptions[EG, EdgeStyle → Thick];
```

pdf

In[•]:=
```
g1 = EG[BR[9, {2, -1, -1, -1, 4, 6, 8}]] (* lifesaver *);
g2 = EG[β2 = BR[5, {3, -4, -3, 4, -1, 4, -1, 4, 3, 4, -2, 1}]] (* Iowa *);
g3 = EG[β2, GraphLayout → Automatic] (* torch flame *);
g4 = EG[VPB[6, σ̄6,2, σ2,4, σ4,3, σ̄1,4, σ3,1, σ4,5, σ3,2, σ̄5,2, σ̄3,2, σ̄6,2]] (* bird *);
```

pdf

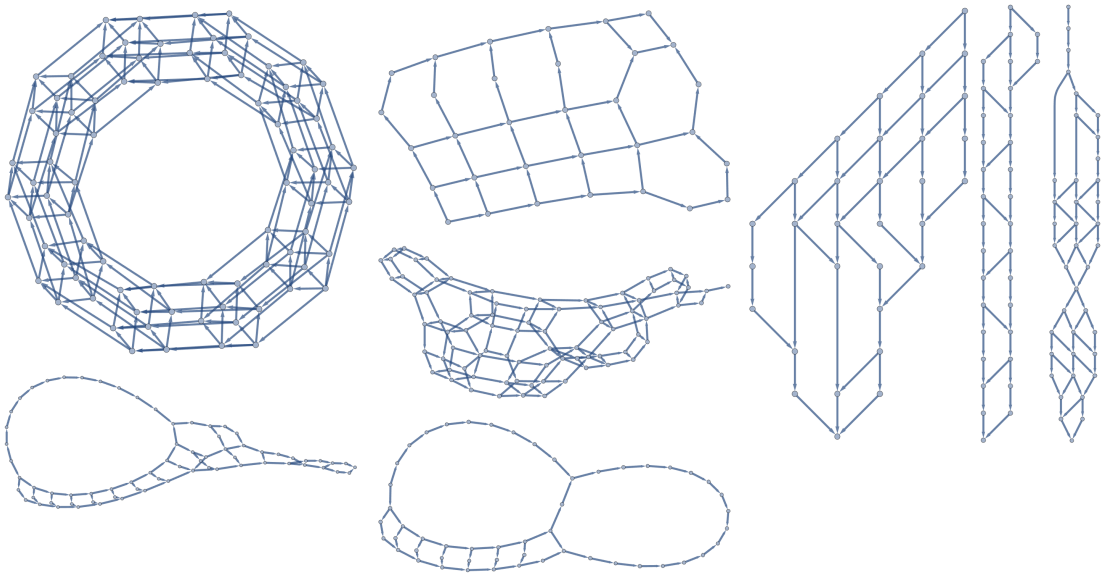
```
(* feather, ladder, tennis racket, mouse trap *)
g5 = EG[VPB[6, σ5,2, σ2,5, σ5,6, σ5,4, σ̄4,3, σ1,3, σ̄4,6, σ4,2,
     σ̄4,6, σ4,3, σ̄6,1, σ̄5,3, σ2,6, σ4,5, σ4,3, σ̄2,5], GraphLayout → Automatic];
g6 = EG[BR[3, {1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2}], GraphLayout → Automatic];
g7 = EG[BR[3, {2, -1, -1, -1, -1, -1, -1, 2, 1, 1, 2, 2, 1, -2, 1, 1, 2, 1}]];
g8 = EG[BR[3, {2, -1, -1, -1, -1, -1, -1, -1, 2, -1, -1, -1, -1, -1, -1, -1, -1}]];
```

pdf

In[•]:=
```
ImageCollage[(Scaled[1] → Show[#]) & /@ {g1, g2, g3, g4, g5, g6, g7, g8},
 ImagePadding → 10, Background → White]
```

pdf

Out[•]=



tex

We don’t know what, if any, can be learned about braids from these graphs, and we can only hope the referee will forgive us for having a bit of fun.

Computational Complexity

tex

```
\subsection{Computational Complexity} \label{ssec:CC}
Looking again at Figure~\ref{fig:divquo}~(C), we see that in the worst case, if the crossing number
$\xi(T)$ of an OU tangle $T$ is $p$, the crossing number of the OU version of $\sigma_{ij}^{\pm 1}T$
might be as big as $3p+1$, and hence the complexity of computing $\Ch$ grows exponentially. Here are
the ``worst'' classical and virtual braids with 8 crossings. A bit more is in the {\sl Mathematica}
notebook {\sl TheWorstBraids.nb} at~\cite{Self}.
```

pdf

In[•]:=
```
Length@Γ̄@BR[3, {-1, 2, -1, 2, -1, 2, -1, 2}] - 3
```

pdf

Out[•]= 172

pdf

In[•]:=
```
Length@T̄@VPB[2, σ1,2, σ̄2,1, σ1,2, σ̄2,1, σ1,2, σ̄2,1, σ1,2, σ̄2,1] - 2
```

pdf

Out[•]= 984

LaTeX Epilogue

tex

}