

THE MATHEMATICA PACKAGE `KnotTheory`

DROR BAR-NATAN

ABSTRACT. We describe the Mathematica package `KnotTheory`, the main tool used to produce The Knot Atlas.

Web version: <http://www.math.toronto.edu/~drorbn/KAtlas/Manual/>

CONTENTS

1. Acknowledgement	2
2. Setup	3
3. Naming and Enumeration	4
4. Presentations	6
4.1. Planar Diagrams	6
4.1.1. Some further details	7
4.2. Gauss Codes	8
4.3. DT (Dowker-Thistlethwaite) Codes	9
4.4. Braid Representatives	10
5. Graphical Output	11
5.1. Drawing Planar Diagrams	11
5.1.1. How does it work?	13
5.2. Drawing Braids	14
6. Structure and Operations	16
7. Invariants	17
7.1. Invariants from Braid Theory	17
7.2. Three Dimensional Invariants	18
7.3. The Alexander-Conway Polynomial	21
7.4. The Determinant and the Signature	23
7.5. The Jones Polynomial	23
7.5.1. How is the Jones polynomial computed?	24
7.6. The Coloured Jones Polynomials	27
7.7. The A2 Invariant	30
7.8. The HOMFLY-PT Polynomial	31
7.9. The Kauffman Polynomial	32
7.10. Finite Type (Vassiliev) Invariants	33
7.11. Khovanov Homology	34

Date: This Edition: September 14, 2005, 14:03:21; *Revision History:* Section 11.
This work was partially supported by NSERC grant RGPIN 262178.

8. Extras	37
8.1. Drawing with TubePlot	37
8.1.1. Standalone TubePlot	40
9. Lightly Documented Features	40
10. Further Knot Theory Software	41
10.1. KnotPlot	41
10.2. Knotscape	41
11. About this Manual...	41
References	42
Index	44

1. ACKNOWLEDGEMENT

I wish to thank the following individuals for help with various aspects of this project:

- Jana Archibald, for writing the program `Alexander[K, r]`.
- Sergei Chmutov, for spotting a typo.
- David De Wit, for a bug report.
- Ralph Furmaniak, for help with our link to Knotilus (Section 4.2).
- Stavros Garoufalidis, for jointly writing the program `ColouredJones` (Section 7.6).
- Thomas Gittings, for the minimum braid representatives for the knots with up to 10 crossings (Section 4.4).
- Jeremy Green, for his java implementation of `Kh` (Section 7.11).
- Thang Le, for supplying some of the formulas used in the program `ColouredJones` (Section 7.6).
- Rick Litherland, for spotting a sneaky bug in the program `KnotSignature`.
- Charles Livingston, for allowing me to bundle data from his Table of Knot Invariants.
- Scott Morrison, for a bug report and for writing the programs to compute the HOMFLY-PT and Kauffman polynomials (Sections 7.8 and 7.9).
- Bertrand Patureau-Mirand, for informing me of some mismatches in the link tables (now corrected).
- Jozef Przytycki, for correcting a typo.
- Stuart Rankin, for help with our link to Knotilus (Section 4.2).
- Emily Redelmeier, for writing the program `DrawPD` (Section 5.1).
- Siddarth Sankaran, for writing the conversion program between Gauss codes and PD codes and for writing `MorseLink` and `DrawMorseLink`.
- Alexander Shumakovitch, for his help with signature computations (Section 7.4).
- Alexander Stoimenow, for the knot presentations for the knots in the Rolfsen table and some further remarks.
- Z-X. Tao, for noticing problems with the knots 10_{83} and 10_{86} .
- Morwen Thistlethwaite, for the pictures of links and of 11 crossing knots.
- Dylan Thurston, for writing a routine to translate from Hoste-Thistlethwaite's DT codes to my "PD Presentations" (Section 4.1).

2. SETUP

You can download the Mathematica package `KnotTheory`' from the web version of this manual. This done, no installation is required (though you may wish to check out "Further Data Files" below). Start Mathematica and you're ready to go:

```
In[1]:= << KnotTheory'
```

```
Loading KnotTheory' (version of September 14, 2005, 13:37:36)...
```

Let us check that everything is working well:

```
In[2]:= Alexander[Knot[6, 2]] [t]
```

```
Out[2]=      -2  3      2
      -3 - t  + - + 3 t - t
              t
```

```
In[3]:= ?KnotTheoryVersion
```

```
KnotTheoryVersion[] returns the date of the current version of the
package KnotTheory'. KnotTheoryVersion[k] returns the kth field in
KnotTheoryVersion[].
```

```
In[4]:= ?KnotTheoryVersionString
```

```
KnotTheoryVersionString[] returns a string containing the date
and time of the current version of the package KnotTheory'. It
is generated from KnotTheoryVersion[].
```

```
In[5]:= ?KnotTheoryWelcomeMessage
```

```
KnotTheoryWelcomeMessage[] returns a string containing the welcome
message printed when KnotTheory' is first loaded.
```

Thus on the day this manual page was last changed, we had:

```
In[6]:= {KnotTheoryVersion[], KnotTheoryVersionString[]}
```

```
Out[6]= {{2005, 9, 14, 13, 37, 36}, September 14, 2005, 13:37:36}
```

```
In[7]:= ?KnotTheoryDirectory
```

```
KnotTheoryDirectory[] returns the best guess KnotTheory' has for its
location on the host computer. It can be reset by the user.
```

Thus with my setup,

```
In[8]:= KnotTheoryDirectory[]
```

```
Out[8]= ./KnotTheory
```

`KnotTheoryDirectory` may not work under some operating systems/environments. Please let me know if you encounter any difficulties.

Notes.

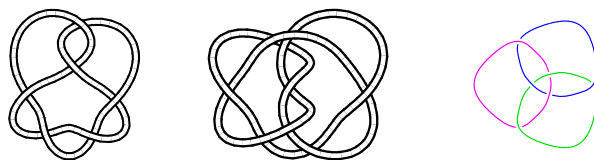


Figure 1. The knots 6_1 and 9_{46} and the link $L6a4$.

- **Precomputed Data.** `KnotTheory` comes with a certain amount of precomputed data which is loaded “on demand” just when it is needed. When a precomputed data file is read by `KnotTheory`, a notification message is displayed. To prevent these messages from appearing execute the command `Off[KnotTheory::loading]`.
- **Further Data Files.** To access the Hoste-Thistlethwaite enumeration of knots with 12 to 16 crossings (Section 3), also download either the file `DTCodes4Knots12To16.tar.gz` (8252Kb) or the file `DTCodes4Knots12To16.zip` (8244Kb), and unpack either one into the directory `KnotTheory/`.

3. NAMING AND ENUMERATION

`KnotTheory` comes loaded with some knot tables; currently, the Rolfsen table of prime knots with up to 10 crossings [Ro], the Hoste-Thistlethwaite tables of prime knots with up to 16 crossings and the Thistlethwaite table of prime links with up to 11 crossings (Section 10.2): (for `In[1]` see Section 2.)

```
In[2]:= ?Knot
Knot[n, k] denotes the kth knot with n crossings in the Rolfsen
table. Knot[11, Alternating, k] denotes the kth alternating
11-crossing knot in the Hoste-Thistlethwaite table. Knot[11,
NonAlternating, k] denotes the kth non alternating 11-crossing knot
in the Hoste-Thistlethwaite table.

In[3]:= ?Link
Link[n, Alternating, k] denotes the kth alternating n-crossing link
in the Thistlethwaite table. Link[n, NonAlternating, k] denotes the
kth non alternating n-crossing link in the Thistlethwaite table.
```

Thus, for example, let us verify that the knots 6_1 and 9_{46} have the same Alexander polynomial:

```
In[4]:= Alexander[Knot[6, 1]][t]
```

```
Out[4]=      2
          5 - - - 2 t
              t
```

```
In[5]:= Alexander[Knot[9, 46]][t]
```

```
Out[5]=      2
          5 - - - 2 t
              t
```

We can also check that the Borromean rings, $L6a4$ in the Thistlethwaite table, is a 3-component link:

```
In[6]:= Length[Skeleton[Link[6, Alternating, 4]]]
```

```
Out[6]= 3
```

```
In[7]:= ?AllKnots
AllKnots[] return a list of all the named knots known to
KnotTheory.m.

In[8]:= ?AllLinks
AllLinks[] return a list of all the named links known to
KnotTheory.m.
```

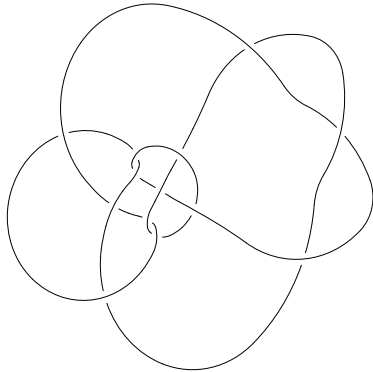
Thus at the moment there are 802 knots and 1424 links known to `KnotTheory`:

```
In[9]:= Length /@ {AllKnots[], AllLinks[]}
```

```
Out[9]= {802, 1424}
```

Note though that if you have also loaded the further files `DTCodes4Knots12To16.tar.gz` (8252Kb) or `DTCodes4Knots12To16.zip` (8244Kb), the contents of `AllKnots[]` does not change but higher knots in the Hoste-Thistlethwaite enumeration become available:

```
In[10]:= Show[DrawPD[Knot[13, NonAlternating, 5016], {Gap -> 0.025}]]
```



```
Out[10]= -Graphics-
```

(Shumakovitch had noticed that this nice knot has interesting Khovanov homology; see [Sh, Section A.4]).

In addition to the tables, `KnotTheory` also knows about torus knots:

```
In[11]:= ?TorusKnot
TorusKnot[m, n] represents the (m,n) torus knot.
```

For example, the torus knots $T(5,3)$ and $T(3,5)$ have different presentations with different numbers of crossings, but they are in fact isotopic, and hence they have the same invariants (and in particular the same type 3 Vassiliev invariant V_3):

```
In[12]:= Crossings /@ {TorusKnot[5,3], TorusKnot[3, 5]}
```

```
Out[12]= {10, 12}
```

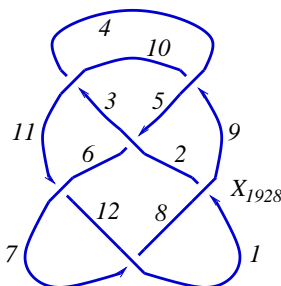


Figure 2. Our notation for planar diagrams

```
In[13]:= Vassiliev[3] /@ {TorusKnot[5,3], TorusKnot[3, 5]}
```

```
Out[13]= {20, 20}
```

`KnotTheory'` knows how to plot torus knots; see Section 8.1.

4. PRESENTATIONS

4.1. Planar Diagrams. For `KnotTheory'`, we present every knot or link diagram (every *Planar Diagram* or just PD) by labeling its edges (with natural numbers, $1, \dots, n$, and with increasing labels as we go around each component) and by a list crossings presented as symbols X_{ijkl} where i, j, k and l are the labels of the edges around that crossing, starting from the incoming lower edge and proceeding counterclockwise. Thus for example, the PD presentation of the knot in Figure 2 is:

$$X_{1928}X_{3,10,4,11}X_{5362}X_{7,1,8,12}X_{9,4,10,5}X_{11,7,12,6}.$$

(This of course is the Miller Institute knot, the mirror image of the knot 6_2 .)

(for `In[1]` see Section 2.)

```
In[2]:= ?PD
```

```
PD[v1, v2, ...] represents a planar diagram whose vertices are v1,
v2, .... PD also acts as a "type caster", so for example, PD[K]
where K is is a named knot (or link) returns the PD presentation of
that knot.
```

```
In[3]:= PD::about
```

```
The PD to GaussCode conversion was written by Siddarth Sankaran at
the University of Toronto in the summer of 2005.
```

```
In[4]:= ?X
```

```
X[i,j,k,l] represents a crossing between the edges labeled i,
j, k and l starting from the incoming lower strand i and going
counterclockwise through j, k and l. The (sometimes ambiguous)
orientation of the upper strand is determined by the ordering of
{j,l}.
```

Thus, for example, let us compute the determinant of the above knot:

```
In[5]:= K = PD[
      X[1,9,2,8], X[3,10,4,11], X[5,3,6,2],
      X[7,1,8,12], X[9,4,10,5], X[11,7,12,6]
    ];
In[6]:= Alexander[K] [-1]
Out[6]= -11
```

4.1.1. Some further details.

```
In[7]:= ?Xp
Xp[i,j,k,l] represents a positive (right handed) crossing between
the edges labeled i, j, k and l starting from the incoming lower
strand i and going counter clockwise through j, k and l. The upper
strand is therefore oriented from l to j regardless of the ordering
of {j,l}. Presently Xp is only lightly supported.

In[8]:= ?Xm
Xm[i,j,k,l] represents a negative (left handed) crossing between
the edges labeled i, j, k and l starting from the incoming lower
strand i and going counter clockwise through j, k and l. The upper
strand is therefore oriented from j to l regardless of the ordering
of {j,l}. Presently Xm is only lightly supported.

In[9]:= ?P
P[i,j] represents a bivalent vertex whose adjacent edges are i
and j (i.e., a "Point" between the segment i and the segment j).
Presently P is only lightly supported.
```

For example, we could add an extra “point” on the Miller Institute knot, splitting edge 12 into two pieces, labeled 12 and 13:

```
In[10]:= K1 = PD[
      X[1,9,2,8], X[3,10,4,11], X[5,3,6,2],
      X[7,1,8,13], X[9,4,10,5], X[11,7,12,6], P[12,13]
    ];
```

At the moment, many of our routines do not know to ignore such “extra points”. But some do:

```
In[11]:= Jones[K] [q] == Jones[K1] [q]
Out[11]= True
```

```
In[12]:= ?Loop
Loop[i] represents a crossingsless loop labeled i.
```

Hence we can verify that the A2 invariant of the unknot is $q^{-2} + 1 + q^2$:

```
In[13]:= A2Invariant[Loop[1]] [q]
Out[13]=      -2    2
            1 + q  + q
```

4.2. **Gauss Codes.** The *Gauss Code* of an n -crossing knot or link L is obtained as follows:

- Number the crossings of L from 1 to n in an arbitrary manner.
- Order the components of L in some arbitrary manner.
- Start “walking” along the first component of L , taking note of the numbers of the crossings you’ve gone through. If in a given crossing you cross on the “over” strand, write down the number of that crossing. If you cross on the “under” strand, write down the negative of the number of that crossing.
- Do the same for all other components of L (if any).

The resulting list of signed integers (in the case of a knot) or list of lists of signed integers (in the case of a link) is called the *Gauss Code* of L . `KnotTheory` has some rudimentary support for Gauss codes:

(for `In[1]` see Section 2.)

```
In[2]:= ?GaussCode
GaussCode[i1, i2, ...] represents a knot via its Gauss Code
following the conventions used by the knotilus website,
http://srankin.math.uwo.ca/cgi-bin/retrieve.cgi/html/start.html.
Likewise GaussCode[l1, l2, ...] represents a link, where each of
l1, l2, ... is a list describing the code read along one component
of the link. GaussCode also acts as a "type caster", so for
example, GaussCode[K] where K is is a named knot (or link) returns
the Gauss code of that knot.
```

Thus for example, the Gauss codes for the trefoil knot and the Borromean link are:

```
In[3]:= GaussCode /@ {Knot[3, 1], Link[6, Alternating, 4]}
```

```
Out[3]= {GaussCode[-1, 3, -2, 1, -3, 2],
```

```
> GaussCode[{1, -6, 5, -3}, {4, -1, 2, -5}, {6, -4, 3, -2}]}
```

Ralph Furmaniak, working under the guidance of Stuart Rankin and Ortho Flint at the University of Western Ontario, wrote a web-based server called “Knotilus” that takes Gauss codes and outputs pictures of the desired knots and links in several standard image formats.

```
In[4]:= ?KnotilusURL
KnotilusURL[K_] returns the URL of the knot/link K on the knotilus
website,
http://srankin.math.uwo.ca/cgi-bin/retrieve.cgi/html/start.html.
```

Thus,

```
In[5]:= KnotilusURL /@ {Knot[3, 1], Link[6, Alternating, 4]}
```

```
Out[5]= {http://srankin.math.uwo.ca/cgi-bin/retrieve.cgi/-1,3,-2,1,-3,2/goTop.html,
```

```
> http://srankin.math.uwo.ca/cgi-bin/retrieve.cgi/1,-6,5,-3:4,-1,2,-5:6,-4,
```

```
> 3,-2/goTop.html}
```

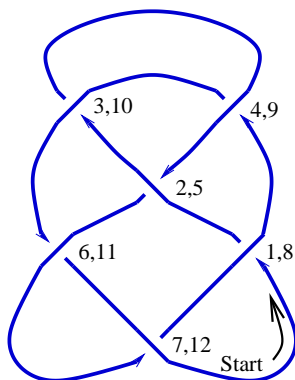


Figure 3. A knot with DT code 8, 10, 2, 12, 4, 6

4.3. DT (Dowker-Thistlethwaite) Codes. The *DT Code* (“DT” after Clifford Hugh Dowker and Morwen Thistlethwaite) of a knot K is obtained as follows:

- Start “walking” along of K , count every crossing you pass through. If K has n crossings and given that every crossing is visited twice, the count ends at $2n$. Label each crossing with the values of the counter when it is visited, though when labeling by an even number, take it with a minus sign if you are walking “under” the crossing.
- Every crossing is now labeled with two integers whose absolute values run from 1 to $2n$. It is easy to see that each crossing is labeled with one odd integer and one even integer. The DT code of K is the list of even integers paired with the odd integers 1, 3, 5, \dots , taken in this order. See Figure 3.

`KnotTheory` has some rudimentary support for DT codes:
(for `In[1]` see Section 2.)

```
In[2]:= ?DTCode
DTCode[i1, i2, ...] represents a knot via its DT
(Dowker-Thistlethwaite) code. DTCode also acts as a "type caster",
so for example, DTCode[K] where K is a named knot returns the DT
code of that knot.
```

Thus for example, the DT codes for the last 9 crossing alternating knot 9_{41} and the first 9 crossing non alternating knot 9_{42} are:

```
In[3]:= dts = DTCode /@ {Knot[9, 41], Knot[9, 42]}
```

```
Out[3]= {DTCode[6, 10, 14, 12, 16, 2, 18, 4, 8],
```

```
> DTCode[4, 8, 10, -14, 2, -16, -18, -6, -12]}
```

(The DT code of an alternating knot is always a sequence of positive numbers but the DT code of a non alternating knot contains both signs.)

DT codes and Gauss codes carry the same information and are easily convertible:

```
In[4]:= gcs = GaussCode /@ dts
```

```

Out[4]= {GaussCode[1, -6, 2, -8, 3, -1, 4, -9, 5, -2, 6, -4, 7, -3, 8, -5, 9, -7],
        > GaussCode[1, -5, 2, -1, 3, 8, -4, -2, 5, -3, -6, 9, -7, 4, -8, 6, -9, 7]}
In[5]:= DTCode /@ gcs
Out[5]= {DTCode[6, 10, 14, 12, 16, 2, 18, 4, 8],
        > DTCode[4, 8, 10, -14, 2, -16, -18, -6, -12]}

```

Conversion between DT codes and/or Gauss codes and PD codes is more complicated; the harder side, going from DT/Gauss to PD, was written by Siddarth Sankaran at the University of Toronto:

```

In[6]:= PD[DTCode[4, 6, 2]]
Out[6]= PD[X[4, 2, 5, 1], X[6, 4, 1, 3], X[2, 6, 3, 5]]

```

4.4. Braid Representatives. Every knot and every link is the closure of a braid. `KnotTheory` can also represent knots and links as braid closures:
(for `In[1]` see Section 2.)

```

In[2]:= ?BR
BR stands for Braid Representative. BR[k,l] represents a braid on
k strands with crossings l={i1,i2,...}, where a positive index i
within the list l indicates a right-handed crossing between strand
number i and strand number i+1 and a negative i indicates a left
handed crossing between strands numbers |i| and |i|+1. Each ij can
also be a list of non-adjacent (i.e., commuting) indices. BR also
acts as a "type caster": BR[K] will return a braid whose closure is
K if K is given in any format that KnotTheory' understands. BR[K]
where K is is a named knot with up to 10 crossings returns a minimum
braid representative for that knot.

In[3]:= BR::about
The minimum braids representing the knots with up to 10 crossings
were provided by Thomas Gittings. See his article on the subject
at arXiv:math.GT/0401051. Vogel's algorithm was implemented by Dan
Carney in the summer of 2005 at the University of Toronto.

In[4]:= ?Mirror
Mirror[br] return the mirror braid of br.

```

Thus for example,

```

In[5]:= br1 = BR[2, {-1, -1, -1}];
In[6]:= PD[br1, q]
Out[6]= PD[BR[2, {-1, -1, -1}], q]
In[7]:= Jones[br1][q]
Out[7]=  -4   -3   1
         -q   + q   + -
                q

```

```
In[8]:= Mirror[br1]
```

```
Out[8]= BR[2, {1, 1, 1}]
```

`KnotTheory` has the braid representatives of some knots and links pre-loaded. Thus for example,

```
In[9]:= BR[TorusKnot[5, 4]]
```

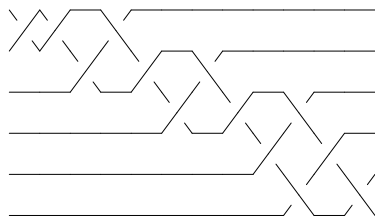
```
Out[9]= BR[4, {1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3}]
```

The *minimum braid representative* of a given knot is a braid representative for that knot which has a minimal number of braid crossings and within those braid representatives with a minimal number of braid crossings, it has a minimal number of strands (full details are in Gittings' [Gi]). Thomas Gittings kindly provided us the minimum braid representatives for all knots with up to 10 crossings. Thus for example, the minimum braid representative for the knot 10_1 has length (number of crossings) 13 and width (number of strands, also see Section 7.1) 6:

```
In[10]:= br2 = BR[Knot[10, 1]]
```

```
Out[10]= BR[6, {-1, -1, -2, 1, -2, -3, 2, -3, -4, 3, 5, -4, 5}]
```

```
In[11]:= Show[BraidPlot[CollapseBraid[br2]]]
```



```
Out[11]= -Graphics-
```

(Check Section 5.2 for information about the command `BraidPlot` and the related command `CollapseBraid`.)

5. GRAPHICAL OUTPUT

5.1. Drawing Planar Diagrams. My summer student Emily Redelmeier is in the process of writing a program that uses circle packing to draw an arbitrary object given as a PD as in Section 4.1. At the moment her program is still slow, limited and sometimes buggy, but it is already quite useful, as the following lines show:

(for `In[1]` see Section 2.)

```

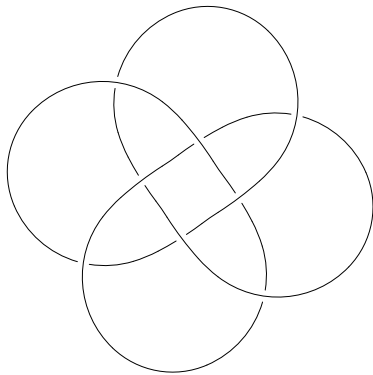
In[2]:= ?DrawPD
DrawPD[pd] takes the planar diagram description pd and
creates a graphics object containing a picture of the knot.
DrawPD[pd,options], where options is a list of rules, allows the
user to control some of the parameters. OuterFace->n sets the face
at infinity to the face numbered n. OuterFace->{e_1,e_2,...,e_n}
sets the face at infinity to a face which has edges e_1, e_2, ...,
e_n in the planar diagram description. Gap->g sets the size of the
gap around a crossing to length g.

In[3]:= DrawPD::about
DrawPD was written by Emily Redelmeier at the University of Toronto
in the summers of 2003 and 2004.

```

Thus, for example, here's the torus knot $T(4,3)$:

```
In[4]:= Show[DrawPD[TorusKnot[4, 3]]]
```



```
Out[4]= -Graphics-
```

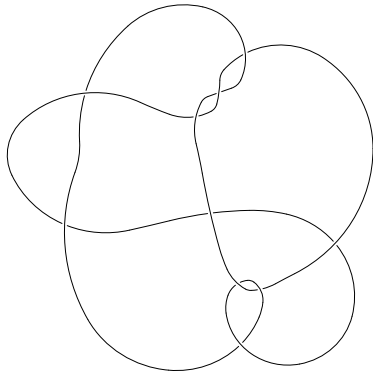
One problem we currently have is that crossings come out at non-uniform sizes, hence in the picture below you may need magnifying glasses to decide who's over and who's under:

```

In[5]:= MillettUnknot = PD[
  X[1,10,2,11], X[9,2,10,3], X[3,7,4,6], X[15,5,16,4], X[5,17,6,16],
  X[7,14,8,15], X[8,18,9,17], X[11,18,12,19], X[19,12,20,13], X[13,20,14,1]
];

```

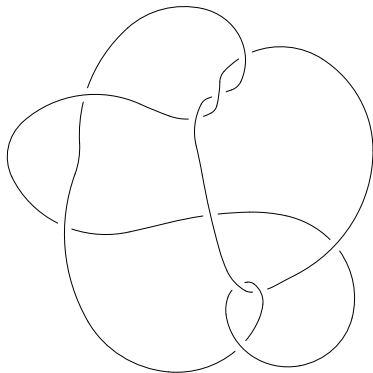
```
In[6]:= Show[DrawPD[MillettUnknot]]
```



Out[6]= -Graphics-

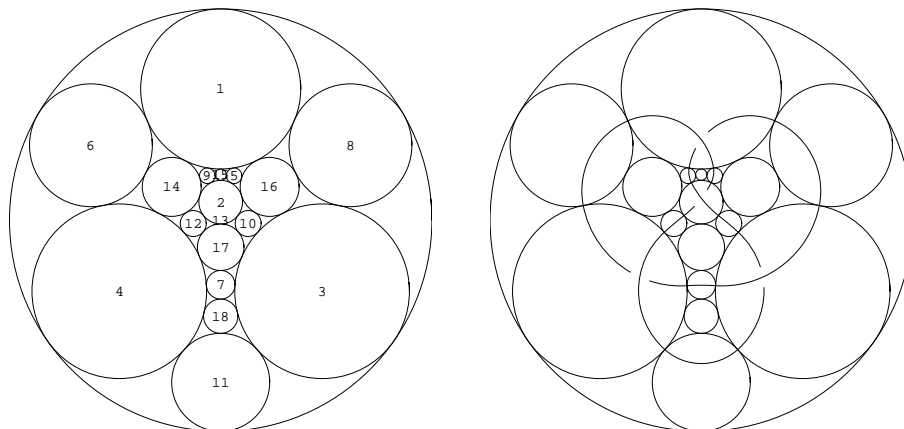
In such a situation, the option `Gap` is sometimes handy:

```
In[7]:= Show[DrawPD[MillettUnknot, {Gap -> 0.03}]]
```



Out[7]= -Graphics-

5.1.1. *How does it work?* `DrawPD` uses Andreev's theorem [An1, An2], which states that every planar graph can be realized, nearly uniquely, as the graph of tangencies of circles drawn within the unit disk. That is, to every vertex of G one may associate a disk within the unit disk, so that the interiors of these disks are disjoint and they are tangent iff the corresponding vertices are connected by an edge. The Andreev "circle packing" corresponding to the knot 4_1 is the left half of the picture below (circle 13 is the unit disk itself).



But now every ingredient of the original knot (every arc, crossing and face) has a disk in the plane in which it can be cleanly drawn and clashes are guaranteed not to occur. Furthermore, knowing the precise coordinates of all the tangency points allows us to represent each ingredient by some nice smooth arcs that meet smoothly. The result is the right half of the picture above. Removing all the circles, what remains is the desired clean planar picture of 4_1 .

5.2. Drawing Braids. (for In[1] see Section 2.)

```
In[2]:= ?BraidPlot
BraidPlot[br, opts] produces a plot of the braid br. Possible
options are Mode, HTMLOpts and Images.

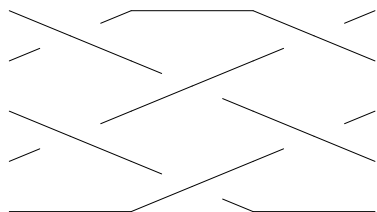
In[3]:= Options[BraidPlot]
{Mode -> "Graphics", Images -> {"0.gif", "1.gif", "2.gif", "3.gif",
"4.gif"}, HTMLOpts -> ""}
```

Thus for example,

```
In[4]:= br = BR[5, {{1,3}, {-2,-4}, {1, 3}}]
```

```
Out[4]= BR[5, {{1, 3}, {-2, -4}, {1, 3}}]
```

```
In[5]:= Show[BraidPlot[br]]
```



```
Out[5]= -Graphics-
```

The `Mode` option to `BraidPlot` defaults to "Graphics", which produces output as above. An alternative is setting `Mode -> "HTML"`, which produces an HTML `<table>` that can be readily inserted into HTML documents:

```
In[6]:= BraidPlot[br, Mode -> "HTML"]
```

```
Out[6]= <table cellspacing=0 cellpadding=0 border=0>
  <tr><td><img src=1.gif><img src=0.gif><img src=1.gif></td></tr>
  <tr><td><img src=2.gif><img src=3.gif><img src=2.gif></td></tr>
  <tr><td><img src=1.gif><img src=4.gif><img src=1.gif></td></tr>
  <tr><td><img src=2.gif><img src=3.gif><img src=2.gif></td></tr>
  <tr><td><img src=0.gif><img src=4.gif><img src=0.gif></td></tr>
</table>
```

The table produced contains an array of image inclusions that together draws the braid using 5 fundamental building blocks: a horizontal "unbraided" line (`0.gif` above), the upper and lower halves of an overcrossing (`1.gif` and `2.gif` above) and the upper and lower halves of an undercrossing (`3.gif` and `4.gif` above).

To see how a web browser displays the above table, check the web version of this manual.

The meaning of the `Images` option to `BraidPlot` should be clear from reading its default definition:

```
In[7]:= Images /. Options[BraidPlot]
```

```
Out[7]= {0.gif, 1.gif, 2.gif, 3.gif, 4.gif}
```

The `HTMLOpts` option to `BraidPlot` allows to insert options within the HTML `` tags. Thus

```
In[8]:= BraidPlot[ BR[2, {1, 1}], Mode -> "HTML", HTMLOpts -> "border=1" ]
```

```
Out[8]= <table cellspacing=0 cellpadding=0 border=0>
  <tr><td><img border=1 src=1.gif><img border=1 src=1.gif></td></tr>
  <tr><td><img border=1 src=2.gif><img border=1 src=2.gif></td></tr>
</table>
```

The web version of this manual contains a rendered example.

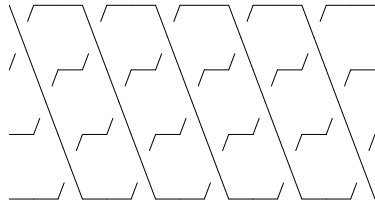
```
In[9]:= ?CollapseBraid
CollapseBraid[br] groups together commuting generators in the braid
br. Useful in conjunction with BraidPlot to produce compact braid
plots.
```

Thus compare the plots of `br1` and `br2` below:

```
In[10]:= br1 = BR[TorusKnot[5, 4]]
```

```
Out[10]= BR[4, {1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3}]
```

```
In[11]:= Show[BraidPlot[br1]]
```

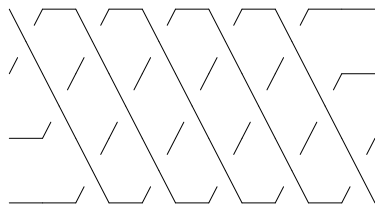


Out[11]= -Graphics-

In[12]:= br2 = CollapseBraid[BR[TorusKnot[5, 4]]]

Out[12]= BR[4, {{1}, {2}, {3, 1}, {2}, {3, 1}, {2}, {3, 1}, {2}, {3, 1}, {2}, {3}}]

In[13]:= Show[BraidPlot[br2]]



Out[13]= -Graphics-

6. STRUCTURE AND OPERATIONS

(for In[1] see Section 2.)

```
In[2]:= ?Crossings
Crossings[L] returns the number of crossings of a knot/link L (in
its given presentation).
In[3]:= ?PositiveCrossings
PositiveCrossings[L] returns the number of positive (right handed)
crossings in a knot/link L (in its given presentation).
In[4]:= ?NegativeCrossings
NegativeCrossings[L] returns the number of negative (left handed)
crossings in a knot/link L (in its given presentation).
```

Thus here's one tautology and one easy example:

In[5]:= Crossings /@ {Knot[0, 1], TorusKnot[11,10]}

Out[5]= {0, 99}

And another easy example:

In[6]:= K=Knot[6, 2]; {PositiveCrossings[K], NegativeCrossings[K]}

Out[6]= {2, 4}

```
In[7]:= ?PositiveQ
PositiveQ[xing] returns True if xing is a positive (right handed)
crossing and False if it is negative (left handed).

In[8]:= ?NegativeQ
NegativeQ[xing] returns True if xing is a negative (left handed)
crossing and False if it is positive (right handed).
```

For example,

```
In[9]:= PositiveQ /@ {X[1,3,2,4], X[1,4,2,3], Xp[1,3,2,4], Xp[1,4,2,3]}
Out[9]= {False, True, True, True}
```

```
In[10]:= ?ConnectedSum
ConnectedSum[K1, K2] represents the connected sum of the knots K1
and K2 (ConnectedSum may not work with links).
```

The connected sum $K = 4_1 \# 4_1$ of the knot 4_1 with itself has 8 crossings (unsurprisingly):

```
In[11]:= K = ConnectedSum[Knot[4,1], Knot[4,1]]
Out[11]= ConnectedSum[Knot[4, 1], Knot[4, 1]]
In[12]:= Crossings[K]
Out[12]= 8
```

It is also nice to know that, as expected, the Jones polynomial of K is the square of the Jones polynomial of 4_1 :

```
In[13]:= Jones[K][q] == Expand[Jones[Knot[4,1]][q]^2]
Out[13]= True
```

It is less nice to know that the Jones polynomial cannot tell K apart from the knot 8_9 :

```
In[14]:= Jones[K][q] == Jones[Knot[8,9]][q]
Out[14]= True
```

But $4_1 \# 4_1$ isn't equivalent to 8_9 ; indeed, their Alexander polynomials are different:

```
In[15]:= {Alexander[K][t], Alexander[Knot[8,9]][t]}
Out[15]= {11 + t^-2 - 6/t + t^2, 7 - t^-3 + 3/t^2 - 5/t + 3t^2 - t^3}
```

7. INVARIANTS

7.1. Invariants from Braid Theory. The *braid length* of a knot or a link K is the smallest number of crossings in a braid whose closure is K . `KnotTheory` has some braid lengths preloaded: (for `In[1]` see Section 2.)

```
In[2]:= ?BraidLength
BraidLength[K] returns the braid length of the knot K, if known to
KnotTheory'.
```

Note that the braid length of K is simply the length of the minimum braid representing K (see Section 4.4):

```
In[3]:= K = Knot[9, 49]; {BraidLength[K], Crossings[BR[K]]}
Out[3]= {11, 11}
```

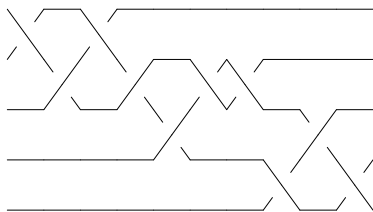
The *braid index* of a knot or a link K is the smallest number of strands in a braid whose closure is K . `KnotTheory'` has some braid indices preloaded:

```
In[4]:= ?BraidIndex
BraidIndex[K] returns the braid index of the knot K, if known to
KnotTheory'.

In[5]:= BraidIndex::about
The braid index data known to KnotTheory' is taken
from Charles Livingston's "Table of Knot Invariants",
http://www.indiana.edu/ knotinfo/.
```

Of the 250 knots with up to 10 crossings, only 10_{136} has braid index smaller than the width of its minimum braid:

```
In[6]:= K = Knot[10, 136]; {BraidIndex[K], First@BR[K]}
Out[6]= {4, 5}
In[7]:= Show[BraidPlot[BR[K]]]
```



```
Out[7]= -Graphics-
```

7.2. Three Dimensional Invariants. (for In[1] see Section 2.)

```
In[2]:= ?SymmetryType
SymmetryType[K] returns the symmetry type of the knot K, if
known to KnotTheory'. The possible types are: Reversible,
FullyAmphicheiral, NegativeAmphicheiral and Chiral.

In[3]:= SymmetryType::about
The symmetry type data known to KnotTheory' is taken
from Charles Livingston's "Table of Knot Invariants",
http://www.indiana.edu/ knotinfo/.
```

The *unknotting number* of a knot K is the minimal number of crossing changes needed in order to unknot K .

```
In[4]:= ?UnknottingNumber
UnknottingNumber[K] returns the unknotting number of the knot K, if
known to KnotTheory'. If only a range of possible values is known,
a list of the form {min, max} is returned.

In[5]:= UnknottingNumber::about
The unknotting numbers of torus knots are due to ???. All
other unknotting numbers known to KnotTheory' are taken
from Charles Livingston's "Table of Knot Invariants",
http://www.indiana.edu/ knotinfo/.
```

Of the 512 knots whose unknotting number is known to `KnotTheory`', 197 have unknotting number 1, 247 have unknotting number 2, 54 have unknotting number 3, 12 have unknotting number 4 and 1 has unknotting number 5:

```
In[6]:= Plus @@ u /@ Cases[UnknottingNumber /@ AllKnots[], _Integer]
```

```
Out[6]= u[0] + 197 u[1] + 247 u[2] + 54 u[3] + 12 u[4] + u[5]
```

There are 4 knots with up to 9 crossings whose unknotting number is unknown:

```
In[7]:= Select[AllKnots[], Crossings[#] <= 9 && Head[UnknottingNumber[#]] === List &]
```

```
Out[7]= {Knot[9, 10], Knot[9, 13], Knot[9, 35], Knot[9, 38]}
```

```
In[8]:= ?ThreeGenus
ThreeGenus[K] returns the 3-genus of the knot K, if known to
KnotTheory'.

In[9]:= ThreeGenus::about
The 3-genus data known to KnotTheory' is taken from
Charles Livingston's "Table of Knot Invariants",
http://www.indiana.edu/ knotinfo/.
```

The *bridge index* of a knot K is the minimal number of local maxima (or local minima) in a generic smooth embedding of K in \mathbb{R}^3 .

```
In[10]:= ?BridgeIndex
BridgeIndex[K] returns the bridge index of the knot K, if known to
KnotTheory'.

In[11]:= BridgeIndex::about
The bridge index data known to KnotTheory' is taken
from Charles Livingston's "Table of Knot Invariants",
http://www.indiana.edu/ knotinfo/.
```

An often studied class of knots is the class of 2-bridge knots, knots whose bridge index is 2. Of the 49 9-crossings knots, 24 are 2-bridge:

```
In[12]:= Select[AllKnots[], Crossings[#] == 9 && BridgeIndex[#] == 2 &]
Out[12]= {Knot[9, 1], Knot[9, 2], Knot[9, 3], Knot[9, 4], Knot[9, 5], Knot[9, 6],
  > Knot[9, 7], Knot[9, 8], Knot[9, 9], Knot[9, 10], Knot[9, 11],
  > Knot[9, 12], Knot[9, 13], Knot[9, 14], Knot[9, 15], Knot[9, 17],
  > Knot[9, 18], Knot[9, 19], Knot[9, 20], Knot[9, 21], Knot[9, 23],
  > Knot[9, 26], Knot[9, 27], Knot[9, 31]}
```

The *super bridge index* of a knot K is the minimal number, in a generic smooth embedding of K in \mathbb{R}^3 , of the maximal number of local maxima (or local minima) in a rigid rotation of that projection.

```
In[13]:= ?SuperBridgeIndex
SuperBridgeIndex[K] returns the super bridge index of the knot K, if
known to KnotTheory'. If only a range of possible values is known,
a list of the form {min, max} is returned.

In[14]:= SuperBridgeIndex::about
The super bridge index data known to KnotTheory' is taken
from Charles Livingston's "Table of Knot Invariants",
http://www.indiana.edu/ knotinfo/.
```

```
In[15]:= ?NakanishiIndex
NakanishiIndex[K] returns the Nakanishi index of the knot K, if
known to KnotTheory'.

In[16]:= NakanishiIndex::about
The Nakanishi index data known to KnotTheory' is taken
from Charles Livingston's "Table of Knot Invariants",
http://www.indiana.edu/ knotinfo/.
```

```

In[17]:= Profile[K_] := Profile[
  SymmetryType[K], UnknottingNumber[K], ThreeGenus[K]
  BridgeIndex[K], SuperBridgeIndex[K], NakanishiIndex[K]
]
In[18]:= Profile[Knot[9,24]]
Out[18]= Profile[Reversible, 1, 9, {4, 6}, 1]
In[19]:= Ks = Select[
  AllKnots[],
  (Crossings[#] == 9 && Profile[#]==Profile[Knot[9,24]])&
]
Out[19]= {Knot[9, 24], Knot[9, 28], Knot[9, 30], Knot[9, 34]}
In[20]:= Alexander[#][t]& /@ Ks
Out[20]=
  {13 - t-3 +  $\frac{5}{2}$  t-5 -  $\frac{10}{t}$  - 10 t + 5 t2 - t3,
  > -15 + t-3 -  $\frac{5}{2}$  t-5 +  $\frac{12}{t}$  + 12 t - 5 t2 + t3,
  > 17 - t-3 +  $\frac{5}{2}$  t-5 -  $\frac{12}{t}$  - 12 t + 5 t2 - t3,
  > 23 - t-3 +  $\frac{6}{2}$  t-6 -  $\frac{16}{t}$  - 16 t + 6 t2 - t3}

```

7.3. The Alexander-Conway Polynomial. (for In[1] see Section 2.)

```

In[2]:= ?Alexander
Alexander[K][t] computes the Alexander polynomial of a knot K as a
function of the variable t. Alexander[K, r][t] computes a basis of
the r'th Alexander ideal of K in Z[t].

In[3]:= Alexander::about
The program Alexander[K, r] to compute Alexander ideals was written
by Jana Archibald at the University of Toronto in the summer of
2005.

In[4]:= ?Conway
Conway[K][z] computes the Conway polynomial of a knot K as a
function of the variable z.

```

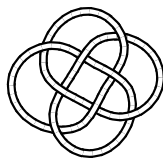


Figure 4. The knot 8_{18} .

The Alexander polynomial $A(K)$ and the Conway polynomial $C(K)$ of a knot K always satisfy $A(K)(t) = C(K)(\sqrt{t} - 1/\sqrt{t})$. Let us verify this relation for the knot 8_{18} :

```
In[5]:= alex = Alexander[Knot[8, 18]] [t]
```

```
Out[5]=      -3  5  10      2  3
          13 - t  + -- - -- - 10 t + 5 t  - t
                   2  t
                  t
```

```
In[6]:= Expand[Conway[Knot[8, 18]] [Sqrt[t] - 1/Sqrt[t]]]
```

```
Out[6]=      -3  5  10      2  3
          13 - t  + -- - -- - 10 t + 5 t  - t
                   2  t
                  t
```

The *determinant* of a knot K is $|A(K)(-1)|$. Hence for 8_{18} it is

```
In[7]:= Abs[alex /. t -> -1]
```

```
Out[7]= 45
```

Alternatively (see Section 7.4):

```
In[8]:= KnotDet[Knot[8, 18]]
```

```
Out[8]= 45
```

$V_2(K)$, the (standardly normalized) type 2 Vassiliev invariant of a knot K is the coefficient of z^2 in its Conway polynomial

```
In[9]:= Coefficient[Conway[Knot[8, 18]] [z], z^2]
```

```
Out[9]= 1
```

Alternatively (see Section 7.10),

```
In[10]:= Vassiliev[2][Knot[8, 18]]
```

```
Out[10]= 0
```

Sometimes two knots have the same Alexander polynomial but different Alexander ideals. An example is the pair K11a99 and K11a277. They have the same Alexander polynomial, but the second Alexander ideal of the first knot is the whole ring $\mathbb{Z}[t]$ while the second Alexander ideal of the second knot is the smaller ideal generated by 3 and by $1 + t$:

```
In[11]:= {K1, K2} = {Knot[11, Alternating, 99], Knot[11, Alternating, 277]};
```

```
In[12]:= Alexander[K1] == Alexander[K2]
```

```
Out[12]= True
```

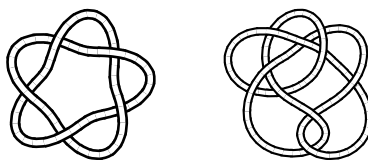


Figure 5. The knots 5_1 and 10_{132} .

```
In[13]:= Alexander[K1, 2][t]
```

```
Out[13]= {1}
```

```
In[14]:= Alexander[K2, 2][t]
```

```
Out[14]= {3, 1 + t}
```

Finally, the Alexander polynomial attains 551 values on the 802 knots known to `KnotTheory`:

```
In[15]:= Length /@ {Union[Alexander[#]& /@ AllKnots[], AllKnots[]}
```

```
Out[15]= {551, 802}
```

7.4. The Determinant and the Signature. (for In[1] see Section 2.)

```
In[2]:= ?KnotDet
```

```
KnotDet[K] returns the determinant of a knot K.
```

```
In[3]:= ?KnotSignature
```

```
KnotSignature[K] returns the signature of a knot K.
```

Thus, for example, the knots 5_1 and 10_{132} have the same determinant (and even the same Alexander and Jones polynomials), but different signatures:

```
In[4]:= KnotDet /@ {Knot[5, 1], Knot[10, 132]}
```

```
Out[4]= {5, 5}
```

```
In[5]:= {
```

```
  Equal @@ (Jones[#][q]& /@ {Knot[5, 1], Knot[10, 132]}),
```

```
  Equal @@ (Alexander[#][t]& /@ {Knot[5, 1], Knot[10, 132]})
```

```
}
```

```
Out[5]= {True, True}
```

```
In[6]:= KnotSignature /@ {Knot[5, 1], Knot[10, 132]}
```

```
Out[6]= {-4, 0}
```

7.5. The Jones Polynomial. (for In[1] see Section 2.)

```
In[2]:= ?Jones
```

```
Jones[L][q] computes the Jones polynomial of a knot or link L as a function of the variable q.
```

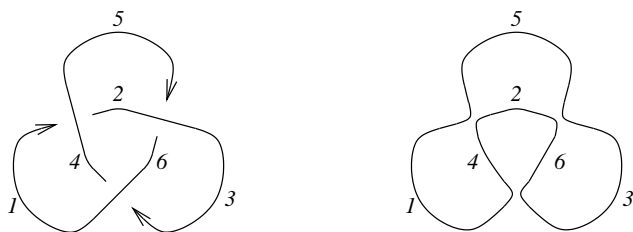


Figure 6. PD[X[1,4,2,5], X[3,6,4,1], X[5,2,6,3]] and P[1,4] P[1,5] P[2,4] P[2,6] P[3,5] P[3,6].

In Section 3 we checked that the knots 6_1 and 9_{46} have the same Alexander polynomial. Their Jones polynomials are different, though:

```
In[3]:= Jones[Knot[6, 1]] [q]
```

```
Out[3]=      -4   -3   -2   2       2
           2 + q  - q  + q  - - - q + q
                        q
```

```
In[4]:= Jones[Knot[9, 46]] [q]
```

```
Out[4]=      -6   -5   -4   2       -2   1
           2 + q  - q  + q  - -- + q  - -
                        3       q
                        q
```

The Jones polynomial attains 2110 values on the 2226 knots and links known to `KnotTheory`:

```
In[5]:= all = Join[AllKnots[], AllLinks[]];
```

```
In[6]:= Length /@ {Union[Jones[#] [q]& /@ all], all}
```

```
Out[6]= {2110, 2226}
```

7.5.1. *How is the Jones polynomial computed?* The Jones polynomial is so simple to compute using Mathematica that it's worthwhile pause and see how this is done, even for readers with limited prior programming experience. First, recall (say from [Ka1]) the definition of the Jones polynomial using the Kauffman bracket $\langle \cdot \rangle$:

$$(1) \quad \langle \emptyset \rangle = 1; \quad \langle \bigcirc L \rangle = (-A^2 - B^2) \langle L \rangle; \quad \langle \times \rangle = A \langle \smile \rangle + B \langle \frown \rangle;$$

$$J(L) = (-A^3)^{w(L)} \frac{\langle L \rangle}{\langle \bigcirc \rangle} \Big|_{A \rightarrow q^{1/4}},$$

here A is a commutative variable, $B = A^{-1}$, and $w(L)$ is the *writhe* of L , the difference $n_+ - n_-$ where n_+ and n_- count the positive (\times) and negative (\smile) crossings of L respectively.

Just for concreteness, let us start by fixing L to be the trefoil knot (see Figure 6):

```
In[7]:= L = PD[Knot[3, 1]]
```

```
Out[7]= PD[X[1, 4, 2, 5], X[3, 6, 4, 1], X[5, 2, 6, 3]]
```

Our first task is to perform the replacement $\langle \times \rangle \rightarrow A \langle \smile \rangle + B \langle \frown \rangle$ on all crossings of L . By our conventions (see Section 4.1) the edges around a crossing X_{abcd} are labeled a , b , c and d : $\begin{matrix} c & b \\ & \times \\ d & a \end{matrix}$. Labeling \smile and \frown in the same way, $\begin{matrix} c & b \\ \smile & \\ d & a \end{matrix}$ and $\begin{matrix} c & b \\ \frown & \\ d & a \end{matrix}$, we are lead to the symbolic replacement rule

$X_{abcd} \rightarrow AP_{ad}P_{bc} + BP_{ab}P_{cd}$. Let us apply this rule to L , switch to a multiplicative notation and expand:

In[8]:= t1 = L /. X[a_,b_,c_,d_] :> A P[a,d] P[b,c] + B P[a,b] P[c,d]

Out[8]= PD[A P[1, 5] P[2, 4] + B P[1, 4] P[2, 5],

> B P[1, 4] P[3, 6] + A P[1, 3] P[4, 6],

> A P[2, 6] P[3, 5] + B P[2, 5] P[3, 6]]

In[9]:= t2 = Expand[Times @@ t1]

Out[9]= 2

A B P[1, 4] P[1, 5] P[2, 4] P[2, 6] P[3, 5] P[3, 6] +

> A² B P[1, 4] P[2, 5] P[2, 6] P[3, 5] P[3, 6] +

> A B P[1, 4] P[1, 5] P[2, 4] P[2, 5] P[3, 6]² +

> B P[1, 4]³ P[2, 5]² P[3, 6]² +

> A P[1, 3]³ P[1, 5] P[2, 4] P[2, 6] P[3, 5] P[4, 6] +

> A B P[1, 3]² P[1, 4] P[2, 5] P[2, 6] P[3, 5] P[4, 6] +

> A B P[1, 3] P[1, 5] P[2, 4] P[2, 5] P[3, 6] P[4, 6]² +

> A B P[1, 3]² P[1, 4] P[2, 5] P[3, 6] P[4, 6]²

In the above expression the product $P[1,4] P[1,5] P[2,4] P[2,6] P[3,5] P[3,6]$ represents a path in which 1 is connected to 4, 1 is connected to 5, 2 is connected to 4, etc. (see Figure 6). We simplify such paths by repeatedly applying the rules $P_{ab}P_{bc} \rightarrow P_{ac}$ and $P_{ab}^2 \rightarrow P_{aa}$:

In[10]:= t3 = t2 //. {P[a_,b_]P[b_,c_] :> P[a,c], P[a_,b_]^2 :> P[a,a]}

Out[10]= 3

B P[1, 1] P[2, 2] P[3, 3] + A B P[2, 2] P[4, 4] + A P[3, 3] P[4, 4] +

> A B P[3, 3] P[4, 4]² + 3 A B P[5, 5]² + A B P[1, 1] P[5, 5]²

To complete the computation of the Kauffman bracket, all that remains is to replace closed cycles (paths of the form P_{aa} by $-A^2 - B^2$, to replace B by A^{-1} , and to simplify:

In[11]:= t4 = Expand[t3 /. P[a_,a_] -> -A^2-B^2 /. B -> 1/A]

Out[11]= -9 1 3 7

-A + - + A + A
A



Figure 7. The link L11a548.

We could have, of course, combined the above four lines to a single very short program, that computes the Kauffman bracket from the beginning to the end:

```
In[12]:= KB0[pd_] := Expand[
  Expand[Times @@ pd /. X[a_,b_,c_,d_] :=> A P[a,d] P[b,c] + 1/A P[a,b] P[c,d]]
  //. {P[a_,b_]P[b_,c_] :=> P[a,c], P[a_,b_]^2 :=> P[a,a], P[a_,a_] -> -A^2-1/A^2}
]
```

```
In[13]:= t4 = KB0[PD[Knot[3, 1]]]
```

```
Out[13]=  -9  1  3  7
          -A  + - + A  + A
                A
```

We will skip the uninteresting code for the computation of the writhe here; it is a linear time computation, and if that's all we ever wanted to compute, we wouldn't have bothered to purchase a computer. For our L the result is -3 , and hence the Jones polynomial of L is given by

```
In[14]:= (-A^3)^(-3) * t4 / (-A^2-1/A^2) /. A -> q^(1/4) // Simplify // Expand
```

```
Out[14]=  -4  -3  1
          -q  + q  + -
                q
```

At merely 3 lines of code, our program is surely nice and elegant. But at 12.59 seconds for an 11 crossing link, it is very slow:

```
In[15]:= Timing[KB0[PD[Link[11, Alternating, 548]]]]
```

```
Out[15]=  {12.59 Second, A-23 +  $\frac{5}{15}$  A10 +  $\frac{10}{7}$  A-3 + 6 A5 + 6 A13 + 5 A17 - 5 A21 + 4 A25 -
```

$$\left. \begin{aligned} & \frac{5}{A} + \frac{13}{A^3} + \frac{17}{A^5} - \frac{5}{A^7} + \frac{21}{A^9} - \frac{25}{A^{11}} \right\}$$

Here's the much faster alternative employed by `KnotTheory`:

```
In[16]:= KB1[pd_PD] := KB1[pd, {}, 1];
KB1[pd_PD, inside_, web_] := Module[
  {pos = First[Ordering[Length[Complement[List @@ #, inside]]& /@ pd]]},
  pd[[pos]] /. X[a_,b_,c_,d_] -> KB1[
    Delete[pd, pos],
    Union[inside, {a,b,c,d}],
    Expand[web*(A P[a,d] P[b,c]+1/A P[a,b] P[c,d])] /. {
      P[e_,f_]P[f_,g_] -> P[e,g], P[e_,_]^2 -> P[e,e], P[e_,e_] -> -A^2-1/A^2
    }
  ]
];
KB1[PD[],_,web_] := Expand[web]
```

```
In[17]:= Timing[KB1[PD[Link[11, Alternating, 548]]]]
```

```
Out[17]=
      -23   5   10   -3           5   13   17   21
      {0.14 Second, A  + --- + -- + A  + 6 A + 6 A + 5 A  - 5 A  + 4 A  -
                    15   7
                    A   A
      >  A  }
      25
```

(So on the link L11a548 KB1 is $12.59/0.14 \sim 90$ times faster than KB0.)

The idea here is to maintain a “computation front”, a planar domain which starts empty and gradually increases until the whole link diagram is enclosed. Within the front, the rules defining the Kauffman bracket, Equation (1), are applied and the result is expanded as much as possible. Outside of the front the link diagram remains untouched. At every step we choose a crossing outside the front with the most legs inside and “conquer” it — apply the rules of (1) and expand again. As our new outpost is maximally connected to our old territory, the length of the boundary is increased in a minimal way, and hence the size of the “web” within our front remains as small as possible and thus quick to manipulate.

In further detail, the routine `KB1[pd, inside, web]` computes the Kauffman bracket assuming the labels of the edges inside the front are in the variable `inside`, the already-computed inside of the front is in the variable `web` and the part of the link diagram yet untouched is `pd`. The single argument `KB1[pd]` simply calls `KB1[pd, inside, web]` with an empty `inside` and with `web` set to 1. The three argument `KB1[pd, inside, web]` finds the position of the crossing maximally connected to the front using the somewhat cryptic assignment

```
pos = First[Ordering[Length[Complement[List @@ #, inside]]& /@ pd]]}
```

`KB1[pd, inside, web]` then recursively calls itself with that crossing removed from `pd`, with its legs added to the `inside`, and with `web` updated in accordance with (1). Finally, when `pd` is empty, the output is simply the value of `web`.

7.6. The Coloured Jones Polynomials. *KnotTheory*' can compute the coloured Jones polynomial of braid closures, using the same formulas as in [GL]:

(for `In[1]` see Section 2.)

```

In[2]:= ?ColouredJones
ColouredJones[br, n][q] computes the coloured Jones polynomial
of the closure of the braid br in colour n (i.e., in the
(n+1)-dimensional representation) and with respect to the variable
q. ColouredJones[K, n][q] does the same for knots for which a braid
representative is known to this program.

In[3]:= ColouredJones::about
The ColouredJones program was written jointly with Stavros
Garoufalidis, based on formulas provided to us by Thang Le.

In[4]:= Options[ColouredJones]
{Compute -> True}

```

Thus, for example, here's the coloured Jones polynomial of the knot 4_1 in the 4-dimensional representation of $sl(2)$:

```
In[5]:= ColouredJones[Knot[4, 1], 3][q]
```

```

Out[5]=
      -12   -11   -10   2   2   3   3       2   4   6   8
      3 + q   - q   - q   + -- - -- + -- - -- - 3 q + 3 q - 2 q + 2 q -
                    8   6   4   2
                    q   q   q   q

      10   11   12
      >  q   - q   + q

```

And here's the coloured Jones polynomial of the same knot in the two dimensional representation of $sl(2)$; this better be equal to the ordinary Jones polynomial of 4_1 !

```
In[6]:= ColouredJones[Knot[4, 1], 1][q]
```

```

Out[6]=
      -2   1   2
      1 + q   - - - q + q
                q

```

```
In[7]:= Jones[Knot[4, 1]][q]
```

```

Out[7]=
      -2   1   2
      1 + q   - - - q + q
                q

```

```

In[8]:= ?CJ'Summand
CJ'Summand[br, n] returned a pair {s, vars} where s is the summand
in the the big sum that makes up ColouredJones[br, n][q] and where
vars is the list of variables that need to be summed over (from 0 to
n) to get ColouredJones[br, n][q]. CJ'Summand[K, n] is the same for
knots for which a braid representative is known to this program.

```

The coloured Jones polynomial of 3_1 is computed via a single summation. Indeed,

```
In[9]:= s = CJ'Summand[Mirror[Knot[3, 1]], n]
```

```

Out[9]=      (3 n)/2 + n CJ'k[1] + (-n + 2 CJ'k[1])/2      1
      {CJ'q      qBinomial[0, 0, ----]
                        CJ'q

>      qBinomial[CJ'k[1], 0, ----] qBinomial[CJ'k[1], CJ'k[1], ----]
                        CJ'q      CJ'q

>      qPochhammer[CJ'q , ----, 0] qPochhammer[CJ'q , ----, CJ'k[1]]
                        CJ'q      CJ'q

>      qPochhammer[CJ'q      , ----, 0], {CJ'k[1]}
                        n - CJ'k[1]  1
                        CJ'q

```

The symbols in the above formula require a definition:

```

In[10]:= ?qPochhammer
qPochhammer[a, q, k] represents the q-shifted factorial of a in base
q with index k. See Eric Weisstein's
  http://mathworld.wolfram.com/q-PochhammerSymbol.html and Axel
Riese's
  www.risc.uni-linz.ac.at/research/combinat/risc/software/qMultiSum/
In[11]:= ?qBinomial
qBinomial[n, k, q] represents the q-binomial coefficient of n and k
in base q. For k<0 it is 0; otherwise it is
  qPochhammer[q^(n-k+1), q, k] / qPochhammer[q, q, k].

```

More precisely, $qPochhammer[a, q, k]$ is

$$(a; q)_k = \begin{cases} (1-a)(1-aq)\dots(1-aq^{k-1}) & k > 0 \\ 1 & k = 0 \\ ((1-aq^{-1})(1-aq^{-2})\dots(1-aq^k))^{-1} & k < 0 \end{cases}$$

and $qBinomial[n, k, q]$ is

$$\binom{n}{k}_q = \begin{cases} \frac{(q^{n-k+1}; q)_k}{(q; q)_k} & k \geq 0 \\ 0 & k < 0. \end{cases}$$

The function `qExpand` replaces every occurrence of a `qPochhammer` symbol or a `qBinomial` symbol by its definition:

```

In[12]:= ?qExpand
qExpand[expr_] replaces all occurrences of qPochhammer and qBinomial
in expr by their definitions as products. See the documentation for
qPochhammer and for qBinomial for details.

```

Hence,

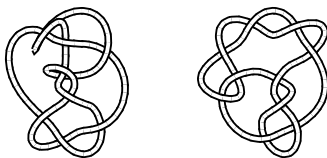


Figure 8. The knots 10_{22} and 10_{35} .

```
In[13]:= qPochhammer[a, q, 6] // qExpand
```

```
Out[13]=
      2      3      4      5
(-1 + a) (-1 + a q) (-1 + a q) (-1 + a q) (-1 + a q) (-1 + a q)
```

```
In[14]:= First[s] /. {n -> 3, CJ'k[1] -> 2} // qExpand
```

```
Out[14]=
      11      2      3
CJ'q (-1 + CJ'q) (-1 + CJ'q)
```

Finally,

```
In[15]:= ?ColoredJones
Type ColoredJones and see for yourself.
```

7.7. The A2 Invariant. We compute the A2 (or quantum $sl(3)$) invariant using the normalization and formulas of [Kh3], which in itself follows [Ku]:

(for In[1] see Section 2.)

```
In[2]:= ?A2Invariant
A2Invariant[L][q] computes the A2 (sl(3)) invariant of a knot or
link L as a function of the variable q.
```

As an example, let us check that the knots 10_{22} and 10_{35} have the same Jones polynomial but different A2 invariants:

```
In[3]:= Jones[Knot[10, 22]][q] == Jones[Knot[10, 35]][q]
```

```
Out[3]= True
```

```
In[4]:= A2Invariant[Knot[10, 22]][q]
```

```
Out[4]=
      -12   -8   -6   -4   2   4   6   8   10   12   14   18
      -1 + q   + q   + q   - q   + -- - q - 2 q + q - q + q + q + q
                        2
                        q
```

```
In[5]:= A2Invariant[Knot[10, 35]][q]
```

```
Out[5]= -14 -12 -10 -8 2 2 2 6 8 10 14 16 18
         q  + q  - q  + q  - -- + -- + q  - q  + q  - 2 q  + q  - q  + q  +
                4  2
                q  q
         20
         > q
```

The A2 invariant attains 2163 values on the 2226 knots and links known to KnotTheory':

```
In[6]:= all = Join[AllKnots[], AllLinks[]];
In[7]:= Length /@ {Union[A2Invariant[#][q]& /@ all], all}
Out[7]= {2163, 2226}
```

7.8. The HOMFLY-PT Polynomial. The *HOMFLY-PT polynomial* $H(L)(a, z)$ (see [HOMFLY] and [PT]) of a knot or link L is defined by the skein relation

$$aH\left(\begin{array}{c} \nearrow \\ \searrow \end{array}\right) - a^{-1}H\left(\begin{array}{c} \nwarrow \\ \swarrow \end{array}\right) = zH\left(\begin{array}{c} \\ \end{array}\right) \left(\begin{array}{c} \\ \end{array}\right)$$

and by the initial condition $H(\bigcirc) = 1$.

KnotTheory' knows about the HOMFLY-PT polynomial:
(for In[1] see Section 2.)

```
In[2]:= ?HOMFLYPT
HOMFLYPT[K][a, z] computes the HOMFLY-PT (Hoste, Ocneanu, Millett,
Freyd, Lickorish, Yetter, Przytycki and Traczyk) polynomial of a
knot/link K, in the variables a and z.

In[3]:= HOMFLYPT::about
The HOMFLYPT program was written by Scott Morrison.
```

Thus, for example, here's the HOMFLY-PT polynomial of the knot 8_1 :

```
In[4]:= K = Knot[8, 1];
In[5]:= HOMFLYPT[Knot[8, 1]][a, z]
Out[5]= -2 4 6 2 2 2 4 2
         a - a + a - z - a z - a z
         3 2 q
         q q
```

It is well known that HOMFLY-PT polynomial specializes to the Jones polynomial at $a = q^{-1}$ and $z = q^{1/2} - q^{-1/2}$ and to the Conway polynomial at $a = 1$. Indeed,

```
In[6]:= {Expand[HOMFLYPT[K][1/q, Sqrt[q]-1/Sqrt[q]]], Jones[K][q]}
Out[6]= -6 -5 -4 2 2 2 2
         {2 + q - q + q - -- + -- - - - q + q ,
                3 2 q
                q q
         -6 -5 -4 2 2 2 2
         > 2 + q - q + q - -- + -- - - - q + q }
                3 2 q
                q q
```

```
In[7]:= {HOMFLYPT[K][1, z], Conway[K][z]}
```

```
Out[7]=      2      2
      {1 - 3 z , 1 - 3 z }
```

In our parametrization of the A_2 link invariant, it satisfies

$$A_2(L)(q) = (-1)^c(q^2 + 1 + q^{-2})H(L)(q^{-3}, q - q^{-1}),$$

where L is some knot or link and where c is the number of components of L . Let us verify this fact for the Whitehead link, $L5a1$:

```
In[8]:= L = Link[5, Alternating, 1];
```

```
In[9]:= Simplify[{
      (-1)^(Length[Skeleton[L]]-1)(q^2+1+1/q^2)HOMFLYPT[L][1/q^3, q-1/q],
      A2Invariant[L][q]
    }]
```

```
Out[9]=      -12      -8      -6      2      -2      2      4      6
      {2 - q      + q      + q      + -- + q      + q      + q      + q      ,
              4
              q
      >      -12      -8      -6      2      -2      2      4      6
              2 - q      + q      + q      + -- + q      + q      + q      + q      }
              4
              q
```

7.9. The Kauffman Polynomial. The *Kauffman polynomial* $F(K)(a, z)$ (see [Ka2]) of a knot or link K is $a^{-w(K)}L(K)$ where $w(L)$ is the writhe of K (see Section 7.5.1) and where $L(K)$ is the regular isotopy invariant defined by the skein relations

$$L(s_+) = aL(s), \quad L(s_-) = a^{-1}L(s)$$

(here s is a strand and s_{\pm} is the same strand with a \pm kink added) and

$$L\left(\begin{array}{c} \diagup \\ \diagdown \end{array}\right) + L\left(\begin{array}{c} \diagdown \\ \diagup \end{array}\right) = z\left(L\left(\begin{array}{c} \diagup \\ \diagup \end{array}\right)\right) + L\left(\begin{array}{c} \diagdown \\ \diagdown \end{array}\right)$$

and by the initial condition $L(\bigcirc) = 1$.

`KnotTheory` knows about the Kauffman polynomial:

(for `In[1]` see Section 2.)

```
In[2]:= ?Kauffman
Kauffman[K][a, z] computes the Kauffman polynomial of a knot or link
K, in the variables a and z.

In[3]:= Kauffman::about
The Kauffman program was written by Scott Morrison.
```

Thus, for example, here's the Kauffman polynomial of the knot 5_2 :

```
In[4]:= Kauffman[Knot[5, 2]][a, z]
```

```
Out[4]=  2  4  6  5  7  2 2  4 2  6 2  3 3  5 3
        -a + a + a - 2 a z - 2 a z + a z - a z - 2 a z + a z + 2 a z +
        7 3  4 4  6 4
        > a z + a z + a z
```

It is well known that the Jones polynomial is related to the Kauffman polynomial via

$$J(L)(q) = (-1)^c L(K)(-q^{-3/4}, q^{1/4} + q^{-1/4}),$$

where K is some knot or link and where c is the number of components of K . Let us verify this fact for the torus knot $T(8, 3)$:

```
In[5]:= K = TorusKnot[8, 3];
```

```
In[6]:= Simplify[{
  (-1)^(Length[Skeleton[K]]-1)Kauffman[K] [-q^(-3/4), q^(1/4)+q^(-1/4)],
  Jones[K][q]
}]
```

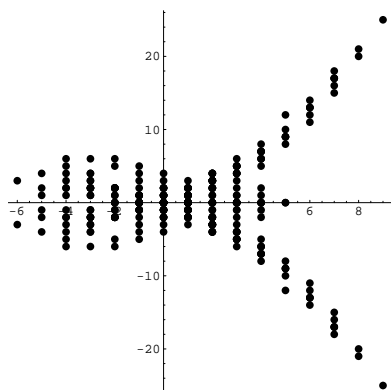
```
Out[6]=  7  9  16  7  9  16
        {q + q - q , q + q - q }
```

7.10. Finite Type (Vassiliev) Invariants. (for In[1] see Section 2.)

```
In[2]:= ?Vassiliev
Vassiliev[2][K] computes the (standardly normalized) type 2
Vassiliev invariant of the knot K, i.e., the coefficient of
z^2 in Conway[K][z]. Vassiliev[3][K] computes the (standardly
normalized) type 3 Vassiliev invariant of the knot K, i.e.,
3J''(1)-(1/36)J'''(1) where J is the Jones polynomial of K.
```

Thus, for example, let us reproduce Willerton's "fish" [Wi1, Wi2], the result of plotting the values of $V_2(K)$ against the values of $\pm V_3(K)$, where $V_2(K)$ is the (standardly normalized) type 2 invariant of K , $V_3(K)$ is the (standardly normalized) type 3 invariant of K , and where K runs over a set of knots with equal crossing numbers (10, in the example below):

```
In[3]:= ListPlot[
  Join @@ Table[
    K = Knot[10, k] ; v2 = Vassiliev[2][K]; v3 = Vassiliev[3][K];
    {{v2, v3}, {v2, -v3}},
    {k, 165}
  ],
  PlotStyle -> PointSize[0.02], PlotRange -> All, AspectRatio -> 1
]
```



Out[3]= -Graphics-

7.11. Khovanov Homology. The Khovanov Homology $KH(L)$ of a knot or a link L , also known as Khovanov's categorification of the Jones polynomial of L , was defined by Khovanov in [Kh1] (also check my paper [BN1], where the notation is much closer to the notation used here). It is a graded homology theory; each homology group $KH^r(L)$ is in itself a direct sum $\bigoplus_j KH_j^r(L)$ of homogeneous components. Over a field one can form the two-variable "Poincaré polynomial" $Kh(L)$ (which deserves the name "the Khovanov polynomial of L "),

$$Kh(L)(q, t) := \sum_{r,j} t^r q^j \dim KH_j^r(L).$$

(for In[1] see Section 2.)

```
In[2]:= ?Kh
Kh[L][q, t] returns the Poincare polynomial of the Khovanov Homology
of a knot/link L (over a field of characteristic 0) in terms of the
variables q and t. Kh[L, Program -> prog] uses the program prog
to perform the computation. The currently available programs are
"FastKh", written in Mathematica by Dror Bar-Natan in the winter of
2005 and "JavaKh" (default), written in java (java 1.5 required!)
by Jeremy Green in the summer of 2005. The java program is several
thousand times faster than the Mathematica program, though java may
not be available on some systems. "JavaKh" also takes the option
"Modulus -> p" which changes the characteristic of the ground field
to p. If p==0 JavaKh works over the rational numbers; if p==Null
JavaKh works over Z (see ?ZMod for the output format).

In[3]:= Options[Kh]
{ExpansionOrder -> Automatic, Program -> "JavaKh", Modulus -> 0,
JavaOptions -> ""}
```

Thus for example, here's the Khovanov polynomial of the knot 5_1 :

```
In[4]:= kh = Kh[Knot[5, 1]][q, t]
```

$$\text{Out}[4]= \frac{-5}{q} + \frac{-3}{q} + \frac{1}{q^5} + \frac{1}{q^4} + \frac{1}{q^3} + \frac{1}{q^2}$$

The Euler characteristic of the Khovanov Homology $KH(L)$ is (up to normalization) the Jones polynomial $J(L)$ of L . Precisely,

$$\text{Kh}(L)(q, -1) = \hat{J}(L)(q) := (q + q^{-1})J(L)(q^2).$$

Let us verify this in the case of 5_1 :

```
In[5]:= {kh /. t -> -1, Expand[(q+1/q)Jones[Knot[5, 1]][q^2]]}
```

$$\text{Out}[5]= \{-q^{-15} + q^{-7} + q^{-5} + q^{-3}, -q^{-15} + q^{-7} + q^{-5} + q^{-3}\}$$

Khovanov's homology is a strictly stronger invariant than the Jones polynomial. Indeed, $J(5_1) = J(10_{132})$ though $\text{Kh}(5_1) \neq \text{Kh}(10_{132})$:

```
In[6]:= {
  Jones[Knot[5, 1]] === Jones[Knot[10, 132]],
  Kh[Knot[5, 1]] === Kh[Knot[10, 132]]
}
```

```
Out[6]= {True, False}
```

The algorithm presently used by `KnotTheory` is an efficient algorithm modeled on the Kauffman bracket algorithm of Section 7.5.1, as explained in [BN3] (which follows [BN2]). Currently, two implementations of this algorithm are available:

- **FastKh**: My original implementation, written in Mathematica in the winter of 2005. This implementation can be explicitly invoked using the syntax `Kh[L, Program -> "FastKh"] [q, t]` or by changing the default behaviour of `Kh` by evaluating `SetOptions[Kh, Program -> "FastKh"]`.
- **JavaKh**: In the summer of 2005 Jeremy Green re-implemented the algorithm in java (**java 1.5 required!**) with much further care to the details, leading to an improvement factor of several thousands for large knots/links. This implementation is the default. It can also be explicitly invoked from within Mathematica using the syntax `Kh[L, Program -> "JavaKh"] [q, t]`.

`JavaKh` takes an additional option, `Modulus`, which sets the characteristic of the ground field for the homology computations to 0 or to a prime p . Thus for example, the following four `In` lines imply that the Khovanov homology of the torus knot $T(6,5)$ has both 3 torsion and 5 torsion, but no 7 torsion:

```
In[7]:= T65 = TorusKnot[6, 5]; kh = Kh[T65][q, t];
```

```
In[8]:= Kh[T65, Modulus -> 3][q, t] - kh
```

$$\text{Out}[8]= \frac{43}{q} + \frac{13}{t} + \frac{43}{q} + \frac{14}{t}$$

```
In[9]:= Kh[T65, Modulus -> 5][q, t] - kh
```

$$\text{Out}[9]= \frac{35}{q} + \frac{10}{t} + \frac{35}{q} + \frac{11}{t} + \frac{39}{q} + \frac{11}{t} + \frac{39}{q} + \frac{12}{t}$$

```
In[10]:= Kh[T65, Modulus -> 7][q, t] - kh
```

```
Out[10]= 0
```

The following further example is a bit tougher. It takes my computer nearly an hour and some 256Mb of memory to find that the Khovanov homology of the 48-crossing torus knot $T(8,7)$ has 3, 5 and 7 torsion but no 11 torsion:

```
In[11]:= ?JavaOptions
JavaOptions is an option to Kh. Kh[L, Program -> "JavaKh",
JavaOptions -> jopts] calls java with options jopts. Thus for
example, JavaOptions -> "-Xmx256m" sets the maximum java heap size
to 256MB - useful for large computations.
```

```
In[12]:= SetOptions[Kh, JavaOptions -> "-Xmx256m"];
```

```
In[13]:= T87 = TorusKnot[8, 7]; kh = Kh[T87][q, t];
```

```
In[14]:= Factor[Kh[T87, Modulus -> 3][q, t] - kh]
```

```
Out[14]= 79 25
         q  t  (1 + t)
```

```
In[15]:= Factor[Kh[T87, Modulus -> 5][q, t] - kh]
```

```
Out[15]= 61 11          12 10  14 12  18 13
         q  t  (1 + t) (1 + q  t  + q  t  + q  t  )
```

```
In[16]:= Factor[Kh[T87, Modulus -> 7][q, t] - kh]
```

```
Out[16]= 61 14          8 6  12 7  10 8  14 9
         q  t  (1 + t) (1 + q  t  + q  t  + q  t  + q  t  )
```

```
In[17]:= Factor[Kh[T87, Modulus -> 11][q, t] - kh]
```

```
Out[17]= 0
```

JavaKh also works over the integers:

```
In[18]:= ?ZMod
ZMod[m] denotes the cyclic group Z/mZ. Thus if m=0 it is the
infinite cyclic group Z and if m>0 it is the finite cyclic group
with m elements. ZMod[m1, m2, ...] denotes the direct sum of
ZMod[m1], ZMod[m2], ... .
```

For example, the 22nd homology group over \mathbb{Z} of the torus knot $T(8,7)$ at degree 73 is the 280 element torsion group $\mathbb{Z}_2 \oplus \mathbb{Z}_4 \oplus \mathbb{Z}_5 \oplus \mathbb{Z}_7$:

```
In[19]:= Coefficient[Kh[T87, Modulus -> Null][q, t], t^22 * q^73]
```

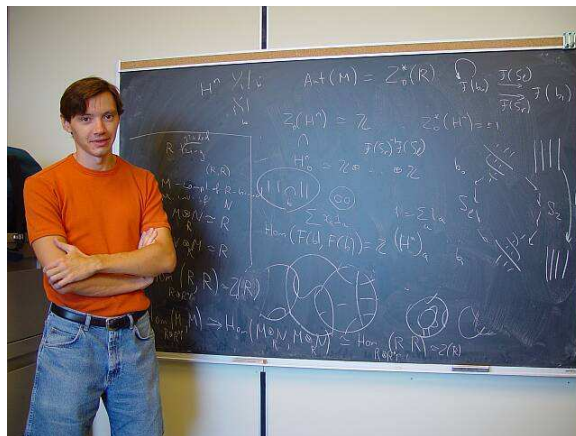


Figure 9. August 2002, Toronto: Mikhail Khovanov explaining his more recent paper [Kh2].

```
Out[19]= ZMod[2, 4, 5, 7]
```

Finally, JavaKh may also be run outside of Mathematica, as the following example demonstrates:

```
drorbn@coxeter:~/KnotTheory: cd JavaKh
drorbn@coxeter:~/KnotTheory/JavaKh: java JavaKh
PD[X[3, 1, 4, 6], X[1, 5, 2, 4], X[5, 3, 6, 2]]
"+ q^1t^0 + q^3t^0 + q^5t^2 + q^9t^3 "
```

(Type `java JavaKh -help` for some further help).

8. EXTRAS

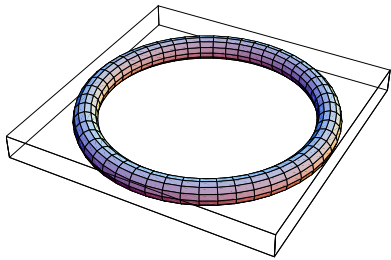
8.1. Drawing with TubePlot. (for In[1] see Section 2.)

```
In[2]:= ?TubePlot
TubePlot[gamma, {t, t0, t1}, r, opts] plots the space curve gamma
with the variable t running from t0 to t1, as a tube of radius
r. The available options are TubeSubdivision, TubeFraming and
TubePlotPrelude. All other options are passed on to Graphics3D.
TubePlot[TorusKnot[m, n], opts] produces a tube plot of the (m,n)
torus knot.

In[3]:= Options[TubePlot]
{TubeSubdivision -> {50, 12}, TubeFraming -> Normal, TubePlotPrelude
-> {}}
```

Thus here's a thin unknot:

```
In[4]:= Show[TubePlot[{Cos[t], Sin[t], 0}, {t, 0, 2Pi}, 0.1]]
```



Out[4]= -Graphics-

```
In[5]:= ?TubeSubdivision
```

TubeSubdivision is an option for TubePlot. TubePlot[\dots , TubeSubdivision \rightarrow {1, m} draws the tube subdivided to 1 pieces lengthwise and m pieces around. The default is TubeSubdivision \rightarrow {50, 12}.

```
In[6]:= ?TubeFraming
```

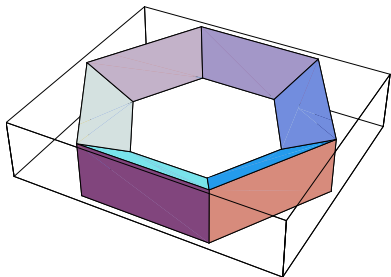
TubeFraming is an option for TubePlot. TubePlot[γ , {t, \dots }, \dots , TubeFraming \rightarrow n] sets the framing of the tube (visible when TubeSubdivision \rightarrow {1, m} with small m) to be the vector n, which in itself may be a function of t. Thus TubeFraming \rightarrow {0,0,1} is "blackboard framing". TubeFraming \rightarrow Normal (default) uses the normal vector of the curve γ .

```
In[7]:= ?TubePlotPrelude
```

TubePlotPrelude is an option for TubePlot. Its value is passed to Graphics3D before the main part of the plot, allowing to set various graphics options. For example, TubePlotPrelude \rightarrow EdgeForm[{}], will suppress the drawing of edges between the polygons making up the tube. The default is TubePlotPrelude \rightarrow {}.

Here's the same unknot, made thicker and not as smooth:

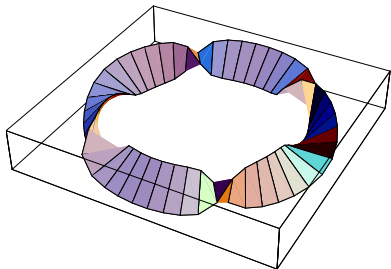
```
In[8]:= Show[TubePlot[
  {Cos[t], Sin[t], 0}, {t, 0, 2Pi}, 0.3, TubeSubdivision -> {6, 3}
]]
```



Out[8]= -Graphics-

Let's play with the framing now:

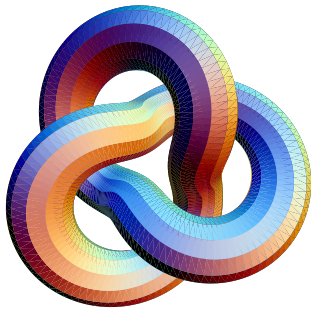
```
In[9]:= Show[TubePlot[
  {Cos[t], Sin[t], 0}, {t, 0, 2Pi}, 0.2,
  TubeSubdivision -> {50, 2},
  TubeFraming -> {Cos[2t]Cos[t], Cos[2t]Sin[t], Sin[3t]}
]]
```



Out[9]= -Graphics-

Here's an example that uses a prelude and passes options on to Graphics3D:

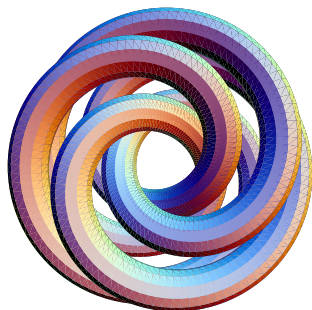
```
In[10]:= Show[TubePlot[
  {Cos[2t], Sin[2t], 0} +
  0.5{Cos[3t]Cos[2t], Cos[3t]Sin[2t], -Sin[3t]},
  {t, 0, 2Pi}, 1/3,
  TubeSubdivision -> {280, 12}, TubeFraming -> {0,0,1},
  TubePlotPrelude -> EdgeForm[{}],
  Boxed -> False, ViewPoint -> {0,0,1}
]]
```



Out[10]= -Graphics-

The last example serves as the basis for the definition of TubePlot[TorusKnot[m, n]]. Here's a final example:

```
In[11]:= Show[TubePlot[TorusKnot[3, 5]]]
```



Out[11]= -Graphics-

8.1.1. *Standalone TubePlot*. There may be some independent interest in the routine `TubePlot`, and hence it is available also as an independent package. See the online version of this manual.

9. LIGHTLY DOCUMENTED FEATURES

(for In[1] see Section 2.)

```
In[2]:= ?NumberOfKnots
NumberOfKnots[type] return the number of knots of a given type.
```

```
In[3]:= NumberOfKnots[16, NonAlternating]
```

```
Out[3]= 1008906
```

```
In[4]:= ?MorseLink
MorseLink[K] returns a presentation of the oriented link K,
composed, in successive order, of the following 'events': Cup[m,n]
is a directed creation, starting at strand position n, towards
position m, where m and n differ by 1. X[n,a = {Over/Under}, b
= {Up/Down}, c={Up/Down}] is a crossing with lower-left edge at
strand n, a determines whether the strand running bottom-left to
top-right is over/under the crossing, b and c give the directions
of the bottom-left and bottom-right strands respectively through the
crossing. Cap[m,n] is a directed cap, from strand m to strand n.

In[5]:= MorseLink::about
MorseLink was added to KnotTheory' by Siddarth Sankaran at the
University of Toronto in the summer of 2005.
```

```
In[6]:= MorseLink[Knot[3, 1]]
```

```
Out[6]= MorseLink[1~Cup~2, 4~Cup~3, X[2, Under, Up, Up], X[2, Under, Up, Up],
> X[2, Under, Up, Up], 2~Cap~1, 1~Cap~2]
```

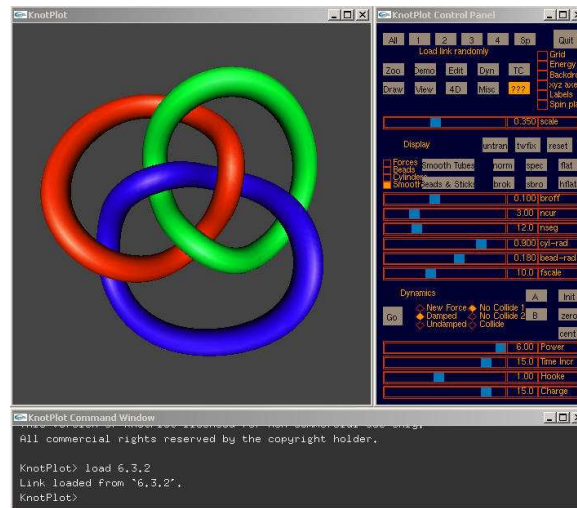
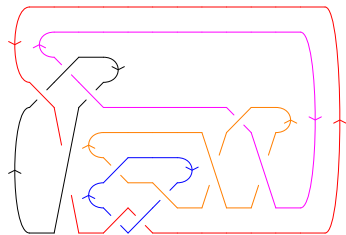


Figure 10. A screenshot of Rob Scharein's KnotPlot

```
In[7]:= ?DrawMorseLink
```

`DrawMorseLink[L]` returns a drawing of the knot or link `L` as a "Morse Link". For diagrams with a large number of crossings, it may be helpful to use one or both of the options as in `DrawMorseLink[L, Gap -> g, ArrowSize -> as]`, with $0 < as, g < 1$, where `g` controls the amount of white space at each crossing, and `as` controls the size of the orientation arrows.

```
In[8]:= Show[DrawMorseLink[Link[11, Alternating, 548]]]
```



```
Out[8]= -Graphics-
```

10. FURTHER KNOT THEORY SOFTWARE

10.1. **KnotPlot**. Many of the images appearing in The Knot Atlas were created using Rob Scharein's program KnotPlot. See Figure 10.

10.2. **Knotscape**. Many of the images appearing in The Knot Atlas were created using Jim Hoste's and Morwen Thistlethwaite's program Knotscape. See Figure 11.

11. ABOUT THIS MANUAL...

Revision History:

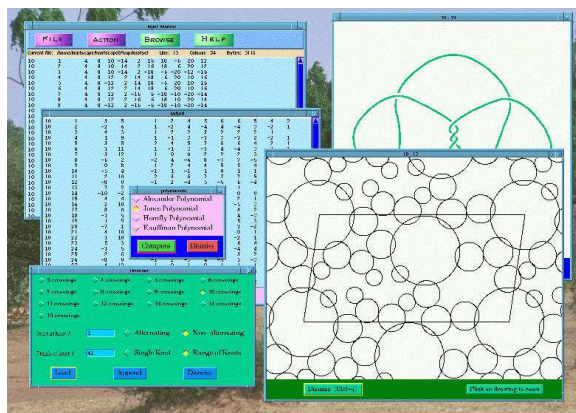


Figure 11. A screenshot of Jim Hoste's and Morwen Thistlethwaite's Knotscape

August 17, 2005: Alexander ideals are now computable. See Section 7.3.

August 11, 2005: Section 9 lightly documents `NumberOfKnots`, `MorseLink` and `DrawMorseLink`.

August 8, 2005: `JavaKh` works also over \mathbb{Z} . See Section 7.11.

July 26, 2005: `JavaKh` is yet faster and also works modulo p . See Section 7.11.

July 24, 2005: An even faster `JavaKh` is linked. See Section 7.11.

July 5, 2005: A fast java implementation of `Kh` is now linked. See Section 7.11.

June 29, 2005: The Hoste-Thistlethwaite knot enumeration is now available up to 16 crossings (Sections 2 and 3).

June 27, 2005: `DTCode` and `GaussCode` to `PD` conversion added.

June 2, 2005: Minor improvement to `ColouredJones` (Section 7.6), `DTCode` to `GaussCode` conversion (Section 4.3).

June 1, 2005: Serious revision tracking begins. New: `DTCode`, Section 4.3.

April 6, 2003: First posted version.

March 18, 2003: Writing began.

About this Document . . .

The printable versions of this document were generated from the source files using a longish makefile that preprocesses everything using `Mathematica` and `sed` and then run `LATEX` (and `dvipdfm/dvips`, as appropriate).

REFERENCES

- [An1] A. Andreev, *On convex polyhedra in Lobacevskii spaces* (in Russian), *Math. Sbornik USSR*, Nov. Ser. **81** (1970) 445–478.
- [An2] A. Andreev, *On convex polyhedra of finite volume in Lobacevskii spaces* (in Russian), *Math. Sbornik USSR*, Nov. Ser. **83** (1970) 256–260.
- [BN1] D. Bar-Natan, *On Khovanov's categorification of the Jones polynomial*, *Algebraic and Geometric Topology* **2-16** (2002) 337–370, <http://www.ma.huji.ac.il/~drornb/papers/Categorification/>, arXiv:math.GT/0201043.
- [BN2] ———, *Khovanov's Homology for Tangles and Cobordisms*, University of Toronto preprint, October 2004, arXiv:math.GT/0410495.
- [BN3] ———, *I've Computed $\text{Kh}(T(9,5))$ and I'm Happy*, talk given at Knots in Washington XX, George Washington University, February 2005.
- [GL] S. Garoufalidis and T. Q. T. Le, *The Colored Jones Function is q -Holonomic*, Georgia Institute of Technology preprint, September 2003, arXiv:math.GT/0309214.

- [Gi] T. A. Gittings, *Minimum braids: a complete invariant of knots and links*, arXiv:math.GT/0401051.
- [HOMFLY] J. Hoste, A. Ocneanu, K. Millett, P. Freyd, W. B. R. Lickorish and D. Yetter, *A new polynomial invariant of knots and links*, Bull. Amer. Math. Soc. **12** (1985) 239–246.
- [Ka1] L. H. Kauffman, *On knots*, Princeton Univ. Press, Princeton, 1987.
- [Ka2] ———, *An invariant of regular isotopy*, Trans. Amer. Math. Soc. **312** (1990) 417–471.
- [Kh1] M. Khovanov, *A categorification of the Jones polynomial*, arXiv:math.QA/9908171.
- [Kh2] ———, *An invariant of tangle cobordisms*, arXiv:math.QA/0207264.
- [Kh3] ———, *$sl(3)$ link homology I*, arXiv:math.QA/0304375.
- [Ku] G. Kuperberg, *Spiders for rank 2 Lie algebras*, Comm. Math. Phys. **180** (1996) 109–151, arXiv:q-alg/9712003.
- [PT] J. Przytycki and P. Traczyk, *Conway Algebras and Skein Equivalence of Links*, Proc. Amer. Math. Soc. **100** (1987) 744–748.
- [Ro] D. Rolfsen, *Knots and Links*, Publish or Perish, Mathematics Lecture Series **7**, Wilmington 1976.
- [Sh] A. Shumakovitch, *Torsion of the Khovanov Homology*, arXiv:math.GT/0405474.
- [Wi1] S. Willerton, *Fishing with Vassiliev invariants*, web document, <http://www.ma.hw.ac.uk/~simon/fishing.html>.
- [Wi2] S. Willerton, *On the first two Vassiliev invariants*, IRMA Strasbourg preprint, March 1999, arXiv:math.GT/0104061.

INDEX

- A2Invariant, **30**
- Alexander, **21**, **23**
- Alexander polynomial, **23**
- AllKnots, **5**
- AllLinks, **5**
- Andreev, A., **13**
- Archibald, Jana, **2**, **21**

- Borromean rings, **4**
- BR, **10**
- Braid index, **18**
- Braid length, **17**
- Braid Representatives, **10**
- BraidIndex, **18**
- BraidLength, **17**
- BraidPlot, **14**
- Bridge index, **19**
- BridgeIndex, **19**

- categorification, **34**
- Chmutov, Sergei, **2**
- Circle Packing, **11**
- CJ'Summand, **28**
- CollapseBraid, **15**
- ColoredJones, **30**
- ColouredJones, **27**
- ConnectedSum, **17**
- Conway, **21**, **31**
- Conway polynomial, **31**
- Crossings, **5**, **16**

- De Wit, David, **2**
- Determinant, **22**
- Dowker, Clifford Hugh, **9**
- DrawMorseLink, **40**
- DrawPD, **11**
- DT Code, **2**, **9**
- DTCode, **9**

- Euler characteristic, **35**

- FastKh, **35**
- Flint, Ortho, **8**
- Freyd, Peter, **31**
- Furmaniak, Ralph, **2**, **8**

- Gap, **13**
- Garoufalidis, Stavros, **2**, **27**
- Gauss Code, **8**
- GaussCode, **8**, **9**
- Gittings, Thomas A., **2**, **10**

- Green, Jeremy, **2**, **35**

- HOMFLY-PT polynomial, **31**
- HOMFLYPT, **31**
- Hoste, Jim, **31**, **41**
- Hoste-Thistlethwaite table, **4**
- HTMLOpts, **14**

- Images, **14**

- JavaKh, **35**
- JavaOptions, **36**
- Jones, **23**, **23**, **30**, **31**, **33**
- Jones polynomial, **23**, **24**, **30**, **31**, **33**, **34**
- Jones, Vaughan, **18**

- Kauffman, **32**
- Kauffman bracket, **24**
- Kauffman polynomial, **32**
- Kauffman, Louis, **32**
- Kh, **34**
- Khovanov Homology, **34**
- Knot, **4**
- KnotDet, **22**, **23**
- Knotilus, **8**
- KnotilusURL, **8**
- KnotPlot, **41**
- Knotscape, **41**
- KnotSignature, **2**, **23**
- KnotTheoryDirectory, **3**
- KnotTheoryVersion, **3**
- KnotTheoryVersionString, **3**
- KnotTheoryWelcomeMessage, **3**

- Le, Thang, **2**, **27**
- Lickorish, Raymond, **31**
- Link, **4**
- Litherland, Rick, **2**
- Livingston, Charles, **2**, **18-20**
- Loop, **7**

- Miller Institute knot, **6**
- Millett, Kenneth, **31**
- MillettUnknot, **12**
- Mirror, **10**
- Mode, **14**
- Modulus, **35**
- Morrison, Scott, **2**, **31**, **32**
- MorseLink, **40**

- NakanishiIndex, **20**
- NegativeCrossings, **16**

NegativeQ, **16**
NumberOfKnots, **40**
 Ocneanu, Adrian, 31
 P, **7**
 Patureau-Mirand, Bertrand, 2
PD, **6**
 Planar Diagrams, 6
 Poincaré polynomial, 34
PositiveCrossings, **16**
PositiveQ, **16**
 Przytycki, Jozef, 2

qBinomial, **29**
qExpand, **29**
qPochhammer, **29**

 Rankin, Stuart, 2, 8
 Redelmeier, Emily, 2, 11
 Riese, Axel, 29
 Rolfsen table, 4

 Sankaran, Siddarth, 2, 6, 10, 40
 Scharein, Rob, 41
 Shumakovitch, Alexander, 2, 5
 $sl(3)$ invariant, **30**
 Stoimenow, Alexander, 2, 18
 Super bridge index, 20
SuperBridgeIndex, **20**
SymmetryType, **18**

 Tao, Z-X., 2
 Thistlethwaite table, 4
 Thistlethwaite, Morwen, 2, 9, 41
ThreeGenus, **19**
 Thurston, Dylan, 2
TorusKnot, **5**, 39
 trefoil, 24
TubeFraming, **38**
TubePlot, **37**
TubePlotPrelude, **38**
TubeSubdivision, **38**

 Unknotting number, 19
UnknottingNumber, **19**

Vassiliev, 22, **33**
 Vassiliev invariants, 5, 22

 Weisstein, Eric, 29
 Willerton's "fish", **33**
 Willerton, Simon, 33
 writhe, **24**, 32

X, **6**
Xm, **7**
Xp, **7**
 Yetter, David, 31

ZMod, **36**

DEPARTMENT OF MATHEMATICS, UNIVERSITY OF TORONTO, TORONTO ON M5S 3G3, CANADA

E-mail address: `drorbn@math.toronto.edu`

URL: `http://www.math.toronto.edu/~drorbn`