

Dror Bar-Natan: Talks: Sandbjerg-0810: The Penultimate Alexander Invariant:
We Mean Business

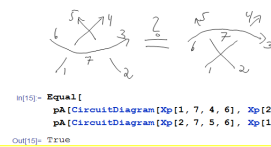
```

1 (* WP: Wedge Product *)
2 WSort[expr_] := Expand[expr /. w_W -> Signature[w]*Sort[w]];
3 WP[0, _] = WP[_ , 0] = 0;
4 WP[a_, b_] := WSort[Distribute[a ** b] /.
5 (c1_ * w1_W) ** (c2_ * w2_W) -> c1 c2 Join[w1, w2]];
6
7 (* IM: Interior Multiplication *)
8 IM[{}, expr_] := expr;
9 IM[i_, w_W] := If[FreeQ[w, i], 0,
10 -(-1)^Position[w, i][[1,1]]*DeleteCases[w, i]];
11 IM[{is___, i_}, w_W] := IM[{is}, IM[i, w]];
12 IM[is_List, expr_] := expr /. w_W -> IM[is, w]
13
14 (* pA on Crossings *)
15 pA[Xp[i_,j_,k_,l_]] := AHD[(t[i]==t[k])(t[j]==t[l]), {i,l}, W[j,k],
16 W[l,i] + (t[i]-1)W[l,j] - t[l]W[l,k] + W[i,j] + t[l]W[j,k]];
17 pA[Xm[i_,j_,k_,l_]] := AHD[(t[i]==t[k])(t[j]==t[l]), {i,j}, W[k,l],
18 t[j]W[i,j] - t[j]W[i,l] + W[j,k] + (t[i]-1)W[j,l] + W[k,l]]
19
20 (* Variable Equivalences *)
21 ReductionRules[Times[]] = {};
22 ReductionRules[Equal[a_, b_]] := (# -> a) & /@ {b};
23 ReductionRules[eqs_Times] := Join @@ (ReductionRules /@ List@@eqs)
24
25 (* AHD: Alexander Half Densities *)
26 AHD[eqs_, is_, -os_, p_] := AHD[eqs, is, os, Expand[-p]];
27 AHD /: Reduce[AHD[eqs_, is_, os_, p_]] :=
28 AHD[eqs, Sort[is], WSort[os], WSort[p /. ReductionRules[eqs]]];
29 AHD /: AHD[eqs1_, is1_, os1_, p1_] AHD[eqs2_, is2_, os2_, p2_] := Module[
30 {glued = Intersection[Union[is1, is2], List@@Union[os1, os2]]},
31 Reduce[AHD[
32 eqs1*eqs2 /. eq1_Equal*eq2_Equal /.
33 Intersection[List@@eq1, List@@eq2] != {} -> Union[eq1, eq2],
34 Complement[Union[is1, is2], glued],
35 IM[glued, WP[os1, os2]],
36 IM[glued, WP[p1, p2]]
37 ]]]
38
39 (* pA on Circuit Diagrams *)
40 pA[cd_CircuitDiagram, eqs___] := pA[cd, {}, AHD[Times[eqs], {}], W[], W[]];
41 pA[cd_CircuitDiagram, done_, ahd_AHD] := Module[
42 {pos = First[Ordering[Length[Complement[List @@ #, done]] & /@ cd]]},
43 pA[Delete[cd, pos], Union[done, List @@ cd[[pos]], ahd*pA[cd[[pos]]]]
44 ];
45 pA[CircuitDiagram[], _, ahd_AHD] := ahd

```

Comments online **2**. $W[i_1, i_2, \dots]$ represents $i_1 \wedge i_2 \wedge \dots$. To sort it we Sort its arguments and multiply by the Signature of the permutation used. **3**. The wedge product of 0 with anything is 0. **4-5**. The wedge product of two things involves applying the Distributive law, Joining all pairs of W's, and WSorting the result. **8**. Inner multiplying by an empty list of indices does nothing. **9-10**. Inner multiplying a single index yields 0 if that index is not present, otherwise it's a sign and the index is deleted. **11-12**. Afterwards it's simple recursion. **15-18**. For the crossings Xp and Xm it is straightforward to determine the incoming strands, the outgoing ones, and the variable equivalences. The associated half-densities are just as in the formulas. **21-23**. The technicalities of imposing variable equivalences are annoying. **26**. That's all we need from the definition of a tensor product. **27-28**. Straightforward simplifications. **29**. The (circuit algebra) product of two Alexander Half Densities: **30**. The glued strands are the intersection of the ins and the outs. **32-33**. Merging the variable equivalences is tricky but natural. **34-35**. Removing the glued strands from the ins and outs. **36 The Key Point**. The wedge product of the half-densities, inner with the glued strands. **40-45**. A quick implementation of a "thin scanning" algorithm for multiple products. The key line is **42**, where we select the next crossing we multiply in to be the crossing with the fewest "loose strands".

Overcrossings Commute



Hence "w-knots"

A very large output was generated. Here is a sample of it:

```

{9.86, {AHD[(t[1]==t[2]==t[3]==t[4]==t[5]==t[6]==t[7]==t[8]==t[9]==t[10])
(t[11]==t[12]==t[13]==t[14]==t[15]==t[16]==t[17]==t[18]==t[19]==t[20])
(t[21]==t[22]==t[23]==t[24]==t[25]==t[26]==t[27]==t[28]==t[29]==t[30]),
{1, 6, 11, 16, 21, 26}, <<1>>,
-t[1]^2 t[11]^2 t[21]^2 W[1, 5, 6, 11, 15, 21] + <<2574>>, <<1>>}}

```

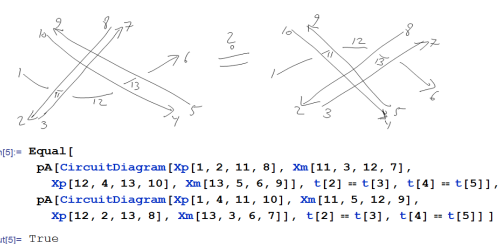
In[11]= Equal @@ (Last /@ res4)

Out[11]= True

(The program also prints "False" when appropriate, and computes Alexander polynomials)

More at <http://www.math.toronto.edu/~drorbn/Sandbjerg-0810/pA.nb>

Commutators Commute



Question: Does this specify the Alexander polynomial?

References

[Ar] J. Archibald, *The Weight System of the Multivariable Alexander Polynomial*, arXiv:0710.4885.

[MH] H. Murakami, *A Weight System Derived from the Multivariable Conway Potential Function*, Jour. of the London Math. Soc. **59** (1999) 698-714, arXiv:math/9903108.

[MJ] J. Murakami, *A State Model for the Multi-Variable Alexander Polynomial*, Pac. Jour. of Math. **157-1** (1993) 109-135.

[NS] S. Naik and T. Stanford, *A Move on Diagrams that Generates S-Equivalence of Knots*, Jour. of Knot Theory and its Ramifications **12-5** (2003) 717-724, arXiv:math/9911005.

[Va] A. Vaintrob, *Melvin-Morton Conjecture and Primitive Feynman Diagrams*, Inter. J. Math. **8** (1997) 537-553, arXiv:q-alg/9605028.