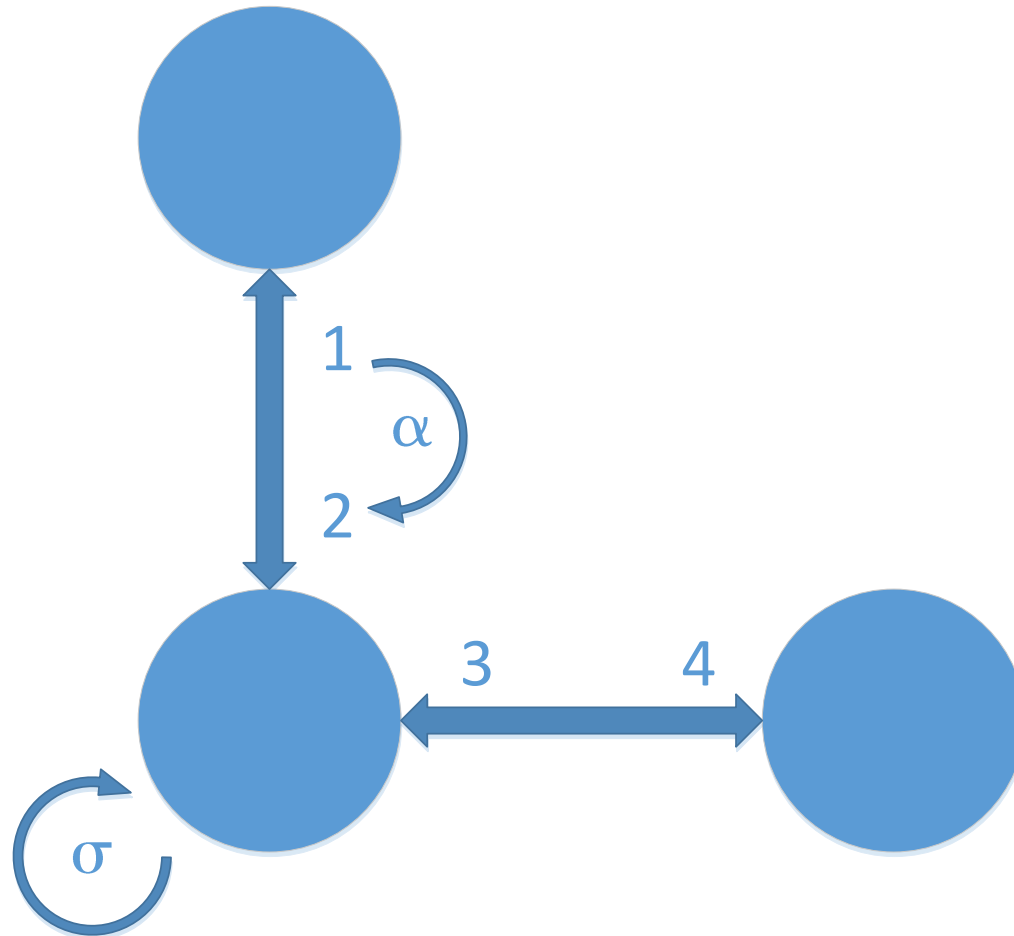


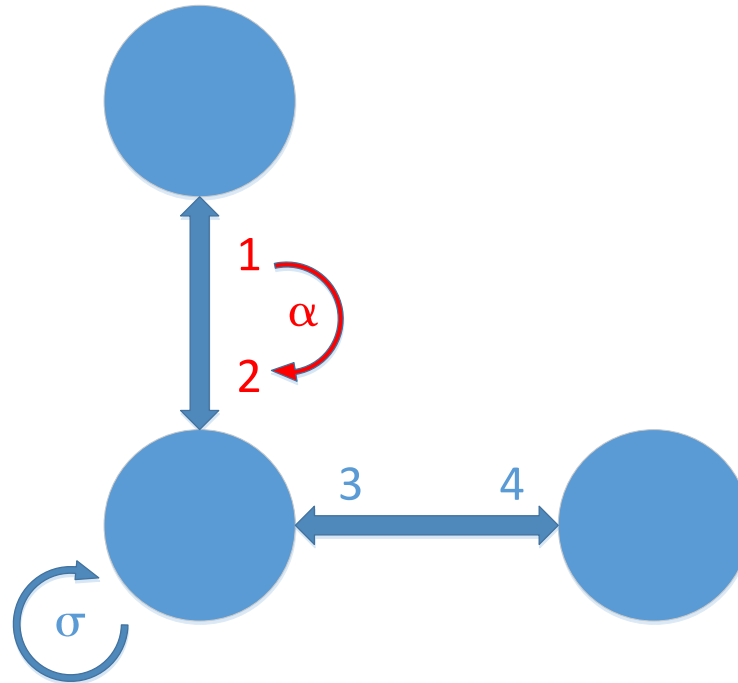
Combinatorial Pyramids – Development and Lessons Learned

David Pfahler

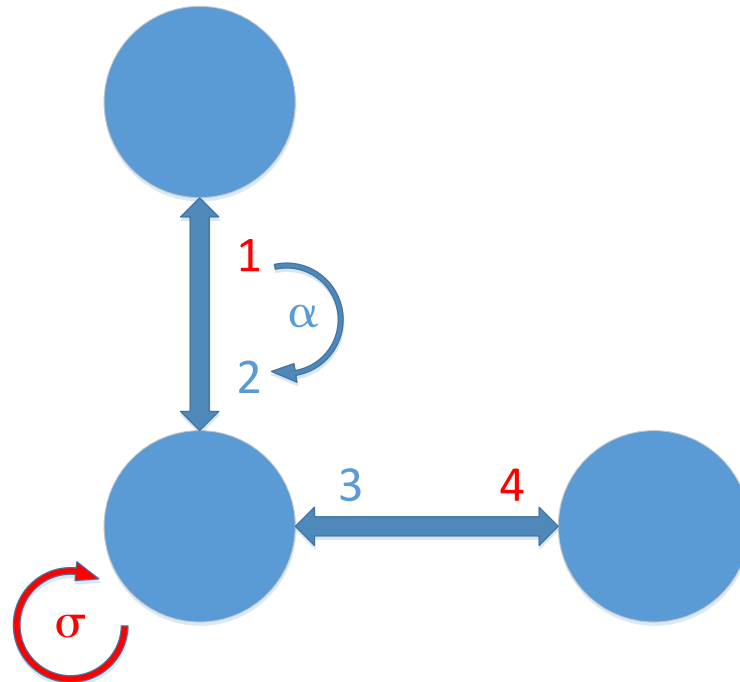
Structural Pattern Recognition



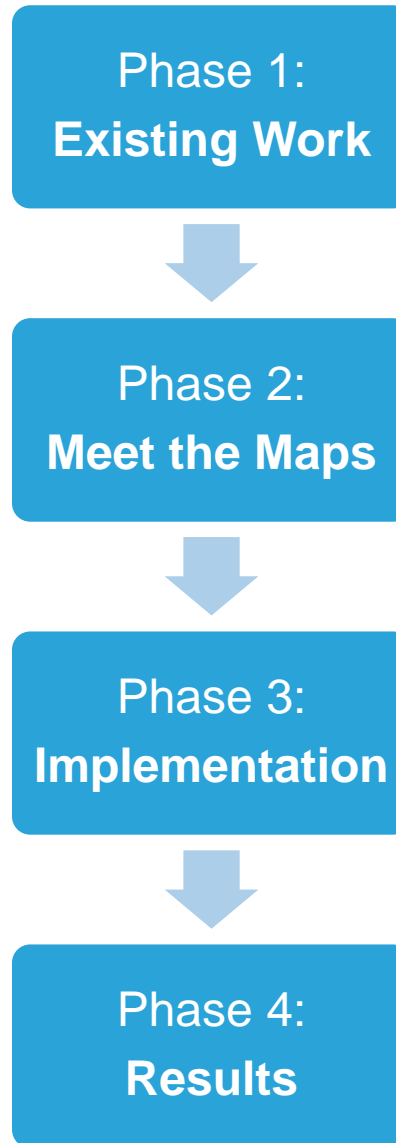
Recap Combinatorial Maps

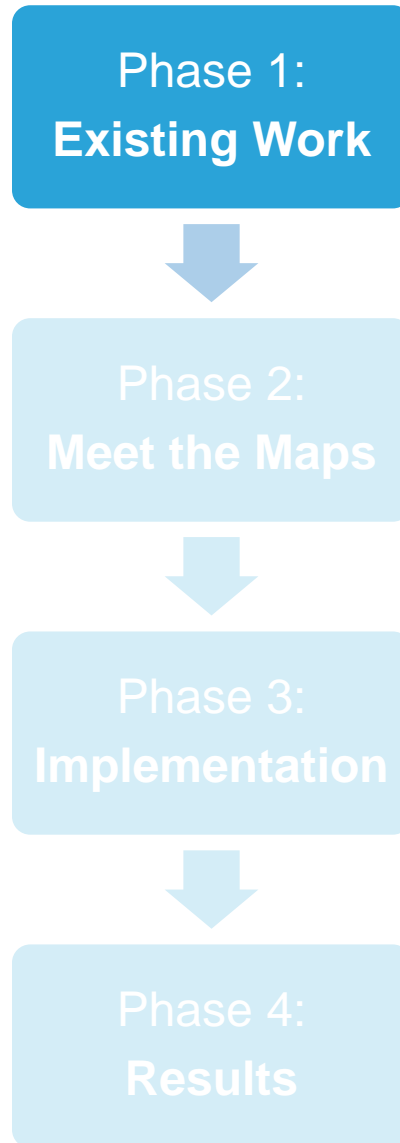


	1	2	3	4
α	2	1	4	3
σ	4	2	3	1



	1	2	3	4
α	2	1	4	3
σ	4	2	3	1

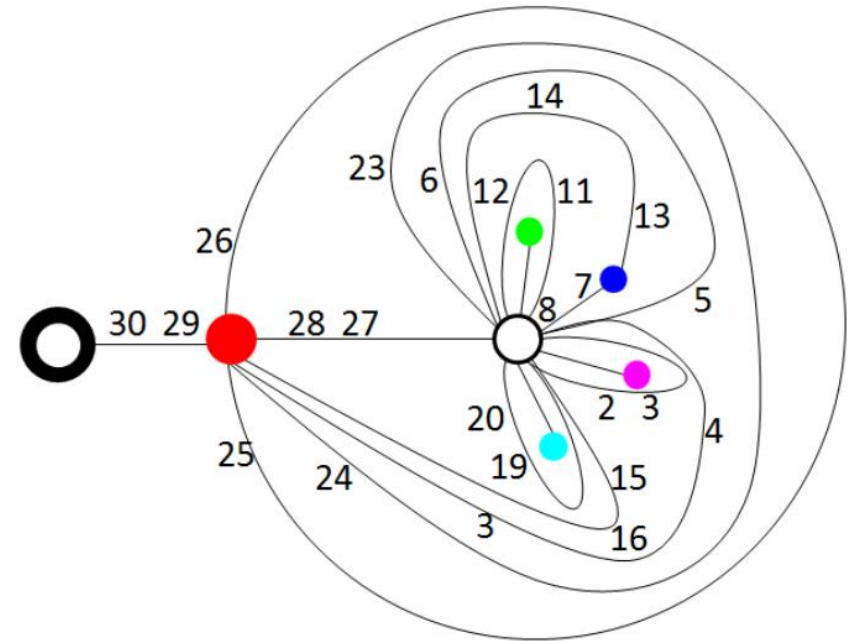




- [Tor] is a Technical Report about operations on a Combinatorial Pyramid (CP).

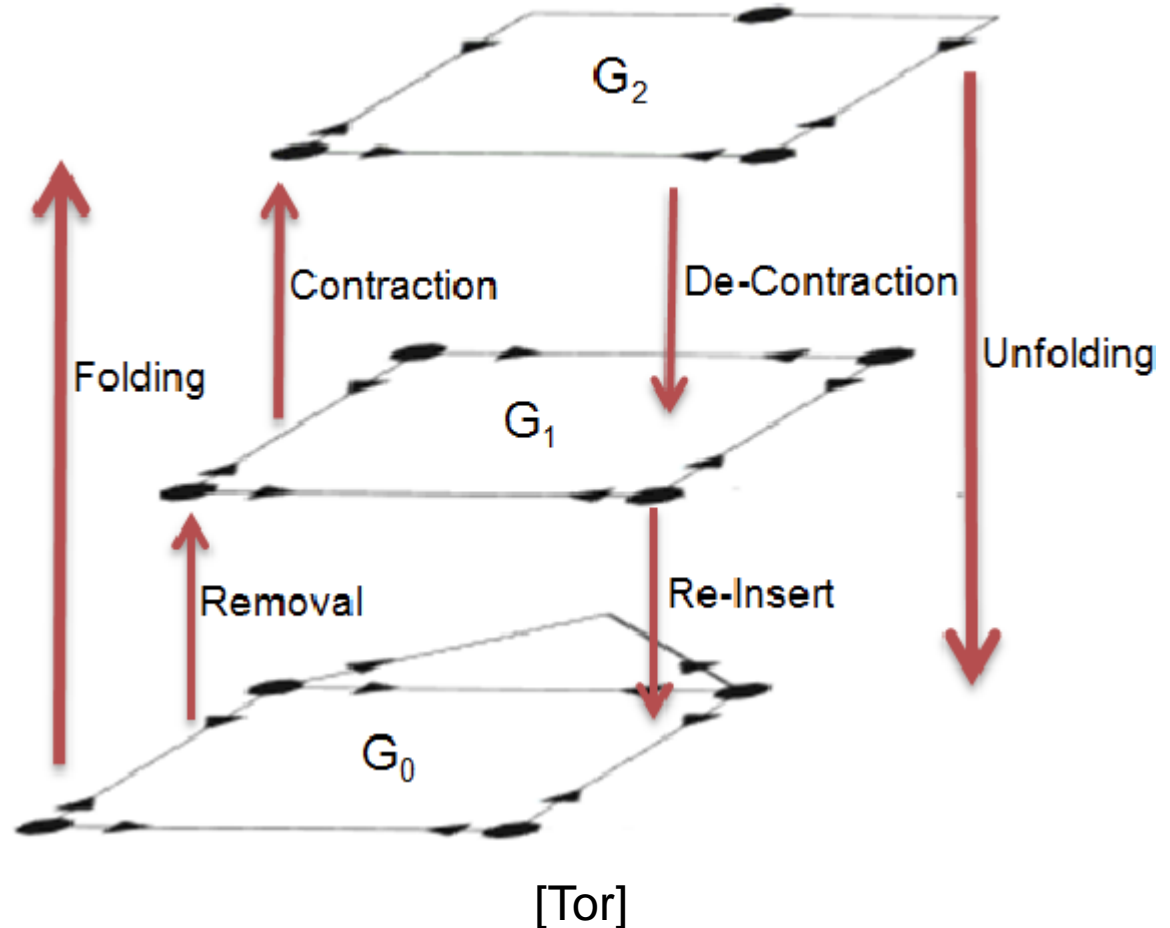


[Tor]

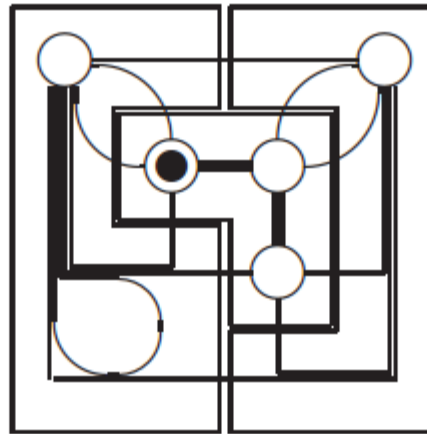
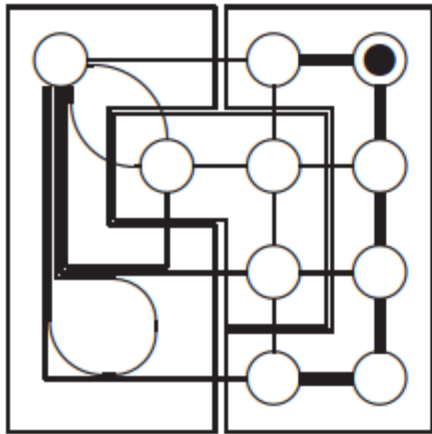


■ Recap: Operations on a Combinatorial Map (CM):

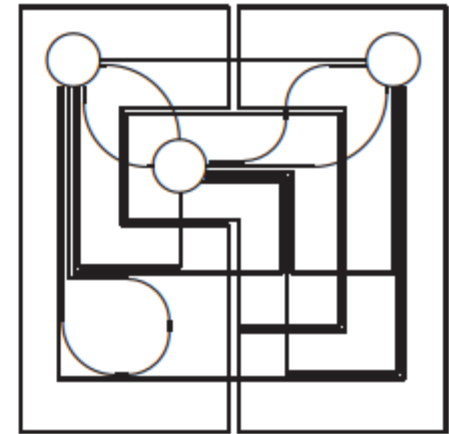
- Remove
- Contract



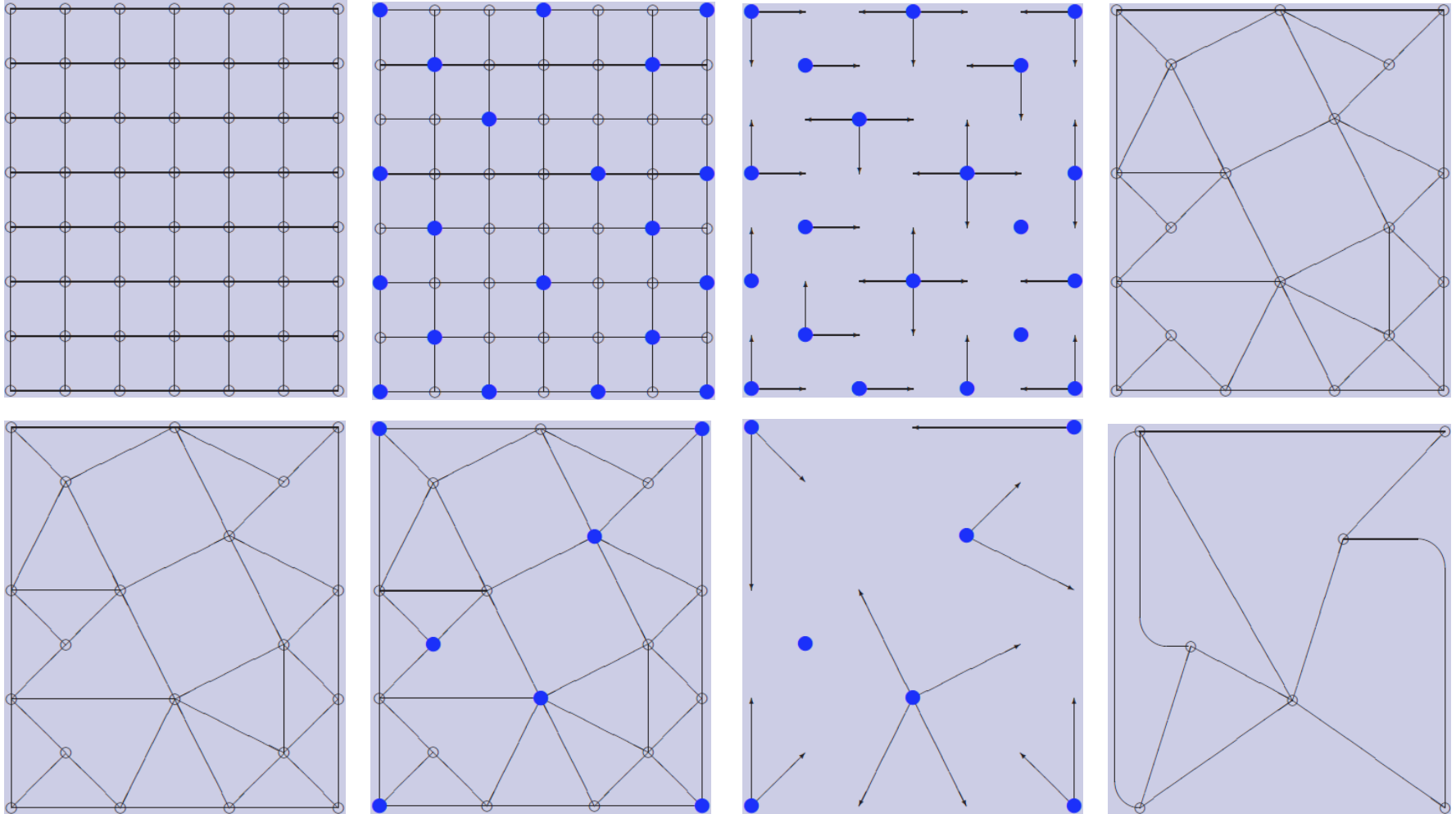
- [Bru01] is an introduction to the CP. With background information on the contraction kernel.



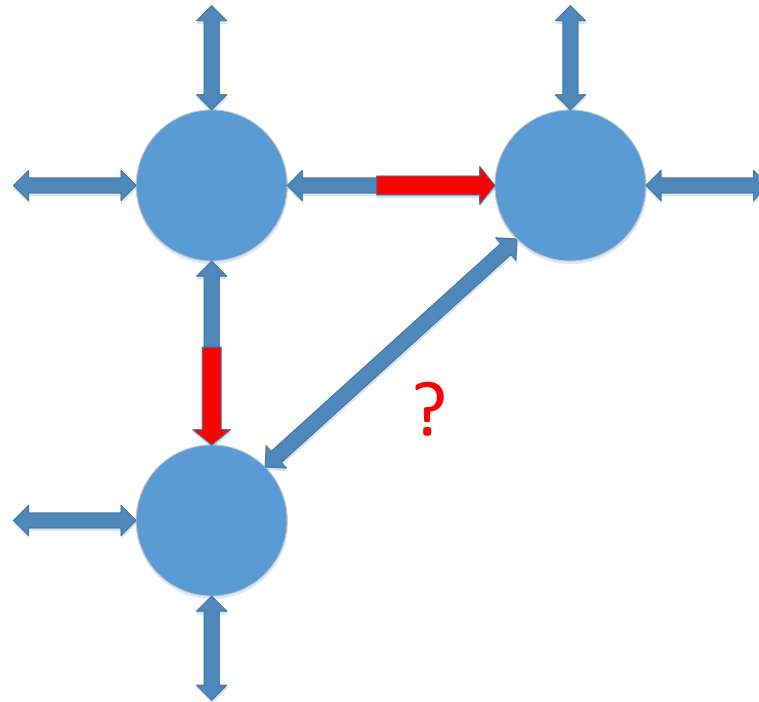
[Bru01]

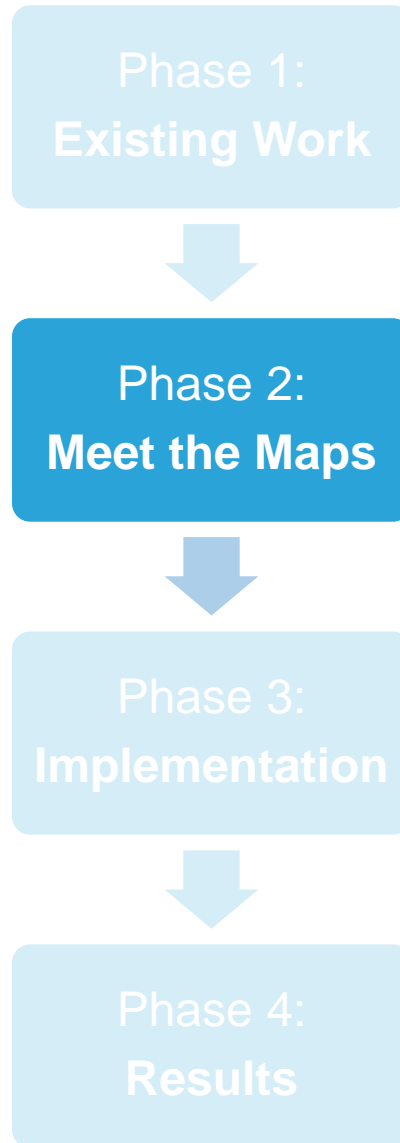


■ Recap: Contraction Kernel



- [Bru03] describes contraction in parallel with contraction kernels
- Recap: Problem why parallel contraction/removal is not trivial





- I Implemented a simple python class for CM
Not for computations but for representations

```
1. import CombinatorialMap
2. map=CombinatorialMap()
3. map.setSize(5, 5)
4. map.printNodes()
```

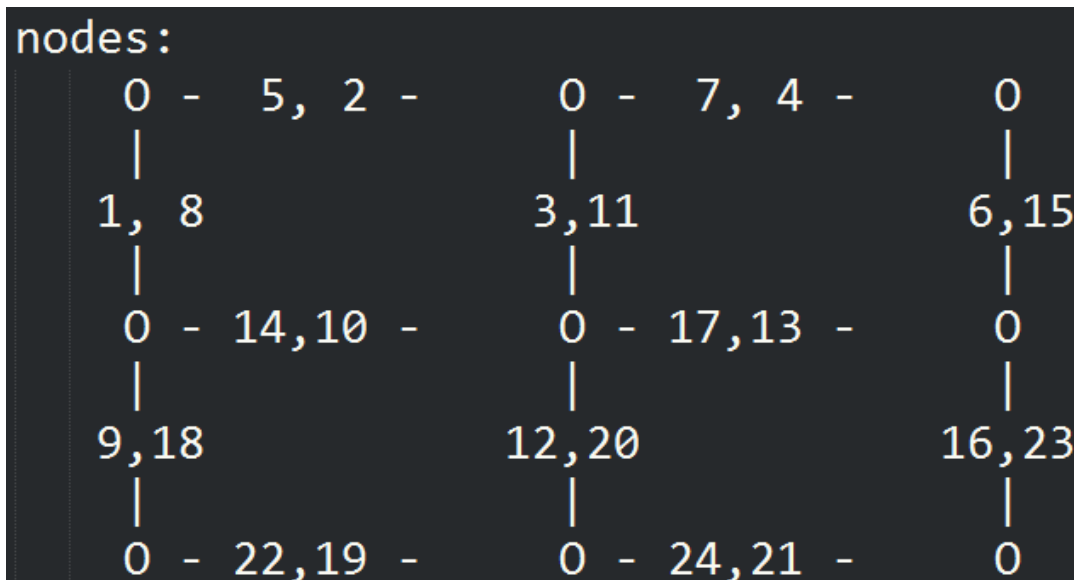
Phase 2: Meet the Maps

nodes:

0 - 5, 2 -	0 - 8, 4 -	0 - 11, 7 -	0 - 13, 10 -	0
1, 14	3, 17	6, 21	9, 25	12, 29
0 - 20, 16 -	0 - 24, 19 -	0 - 28, 23 -	0 - 31, 27 -	0
15, 32	18, 35	22, 39	26, 43	30, 47
0 - 38, 34 -	0 - 42, 37 -	0 - 46, 41 -	0 - 49, 45 -	0
33, 50	36, 53	40, 57	44, 61	48, 65
0 - 56, 52 -	0 - 60, 55 -	0 - 64, 59 -	0 - 67, 63 -	0
51, 68	54, 70	58, 73	62, 76	66, 79
0 - 72, 69 -	0 - 75, 71 -	0 - 78, 74 -	0 - 80, 77 -	0

- Also created functions to receive:
 - All darts with specific direction
 - All involutions of specific darts
 - The orbit of a vertex
 - All next darts
 - ...

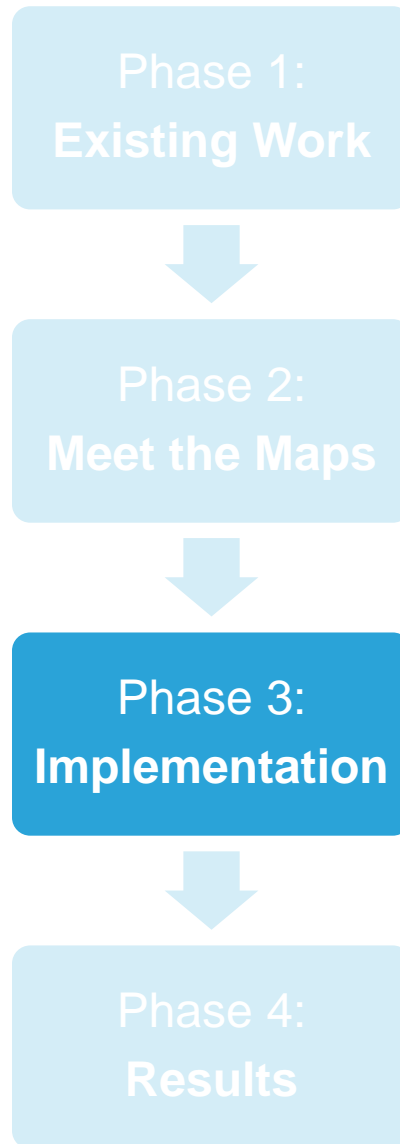
- These are used to understand and create a CM from an image

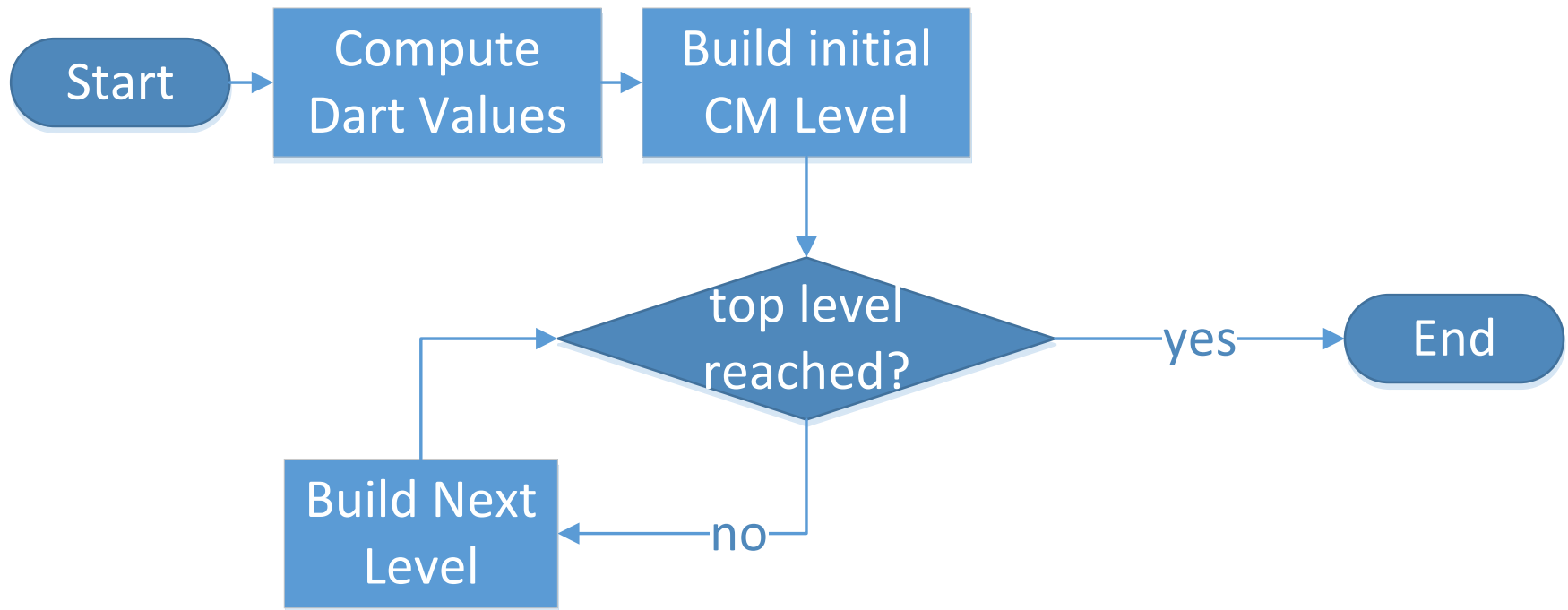


$d \mid \sigma(d) \mid \sigma(d)-d$

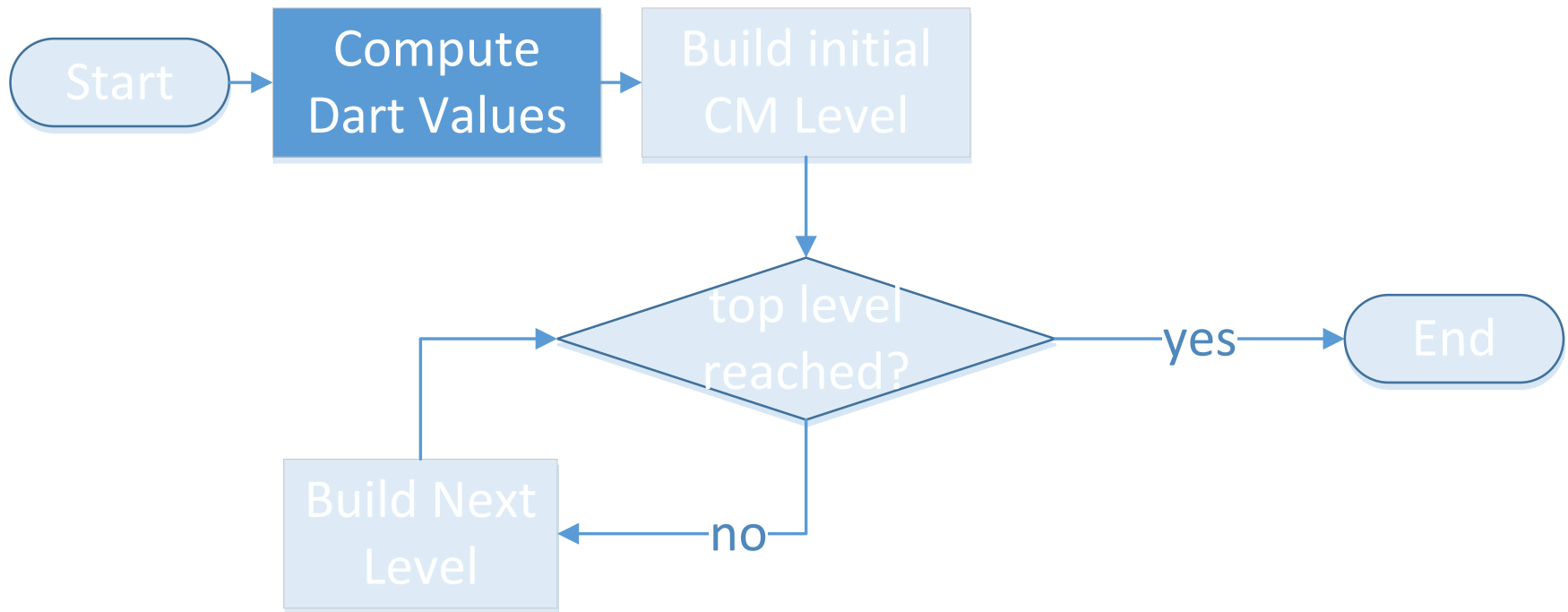
1	2	1
2	1	-1
3	5	2
4	3	-1
5	4	-1

20	21	1
21	22	1
22	20	-2
23	24	1
24	23	-1

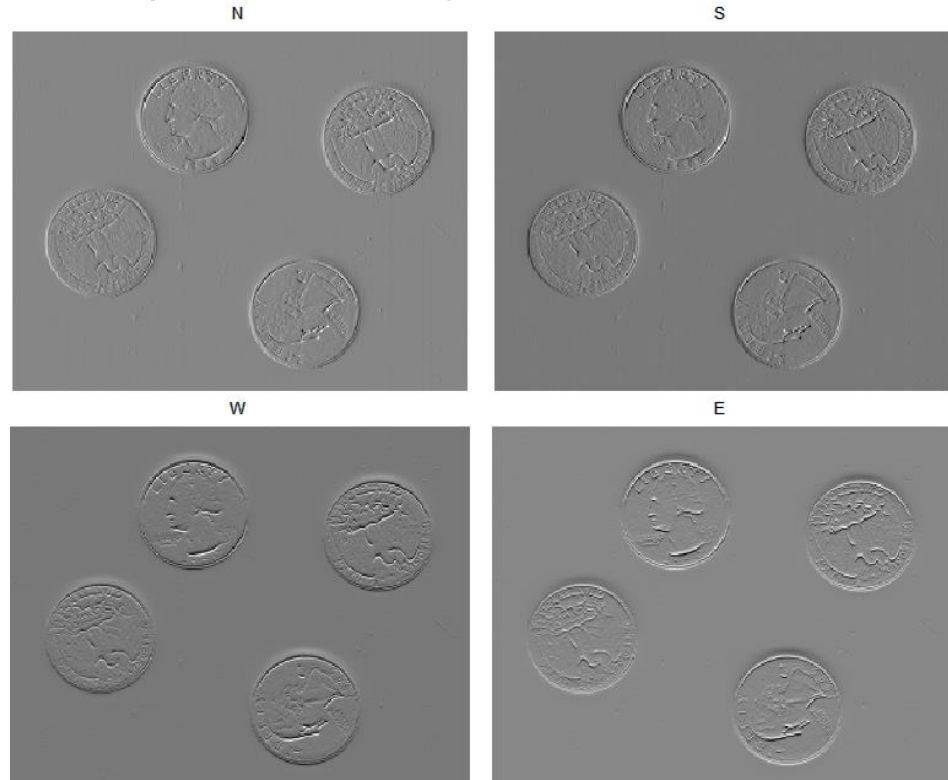


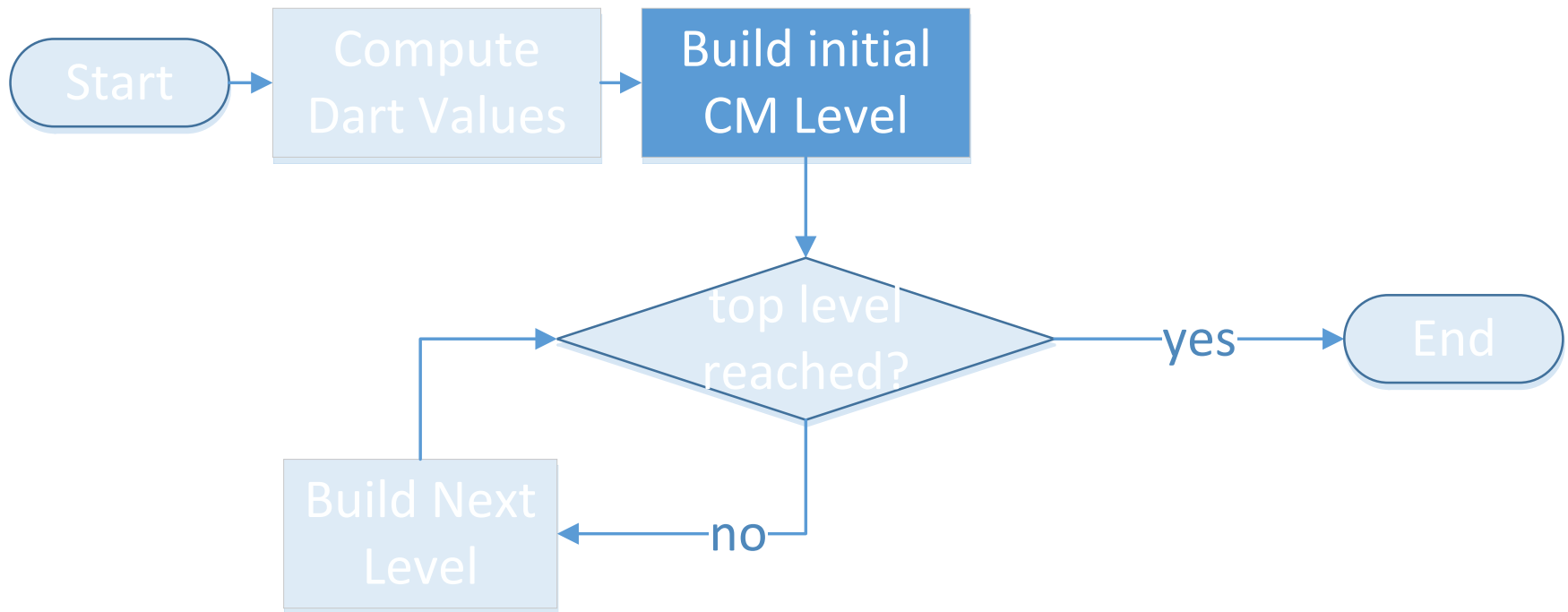


Phase 3: Implementation



- A dart in an image is a transition from one pixel to another.
- Compute from every pixel the change of the pixel value to every neighbor (N,E,S,W)





- Computation of the dart indices
 - Example South Indices:

0 - 38,34 -	0 - 42,37 -	0 - 46,41 -	0 - 49,45 -	0
33,50	36,53	40,57	44,61	48,65
0 - 56,52 -	0 - 60,55 -	0 - 64,59 -	0 - 67,63 -	0

- Added by 4 inside the image. On the border it is only added by 3 (because the west dart is missing)
- Also exceptions in the first row and the last row (missing north dart)
- Additional exceptions for the corners

■ Computation of the next index:

■ one row:

```
repmat([2; 2; -1; -3], width-2, 1)
```

■ middle of the image:

```
repmat([next_darts_one_row; 1; 1; -2; 2; -1; -1],  
height-2, 1)
```

■ Special cases for first and last row

$d \mid \sigma(d) \mid \sigma(d)-d$

1	2	1
2	1	-1
3	5	2
4	3	-1
5	4	-1

20	21	1
21	22	1
22	20	-2
23	24	1
24	23	-1

■ Involution α

- `cm.involution(N) = S`

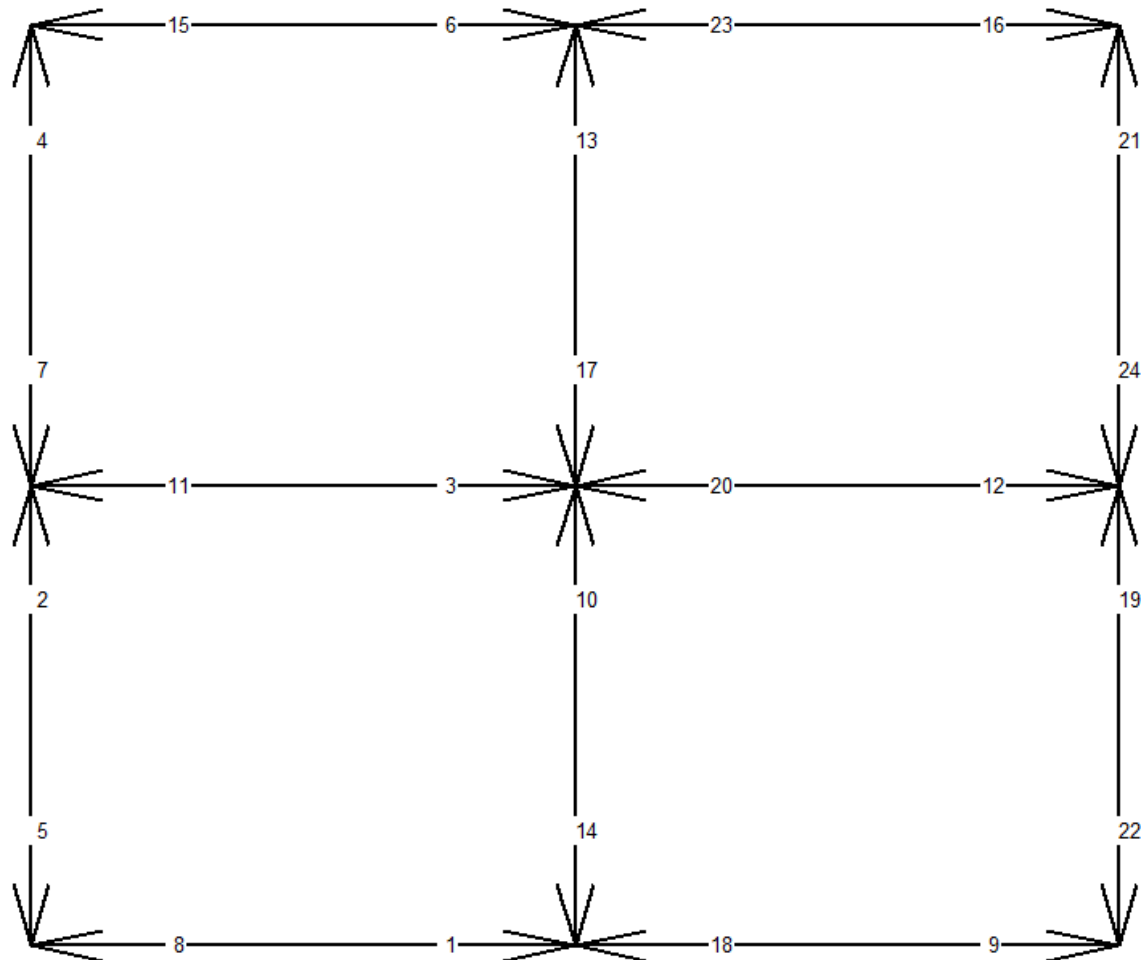
- `cm.involution(E) = W`

- ...

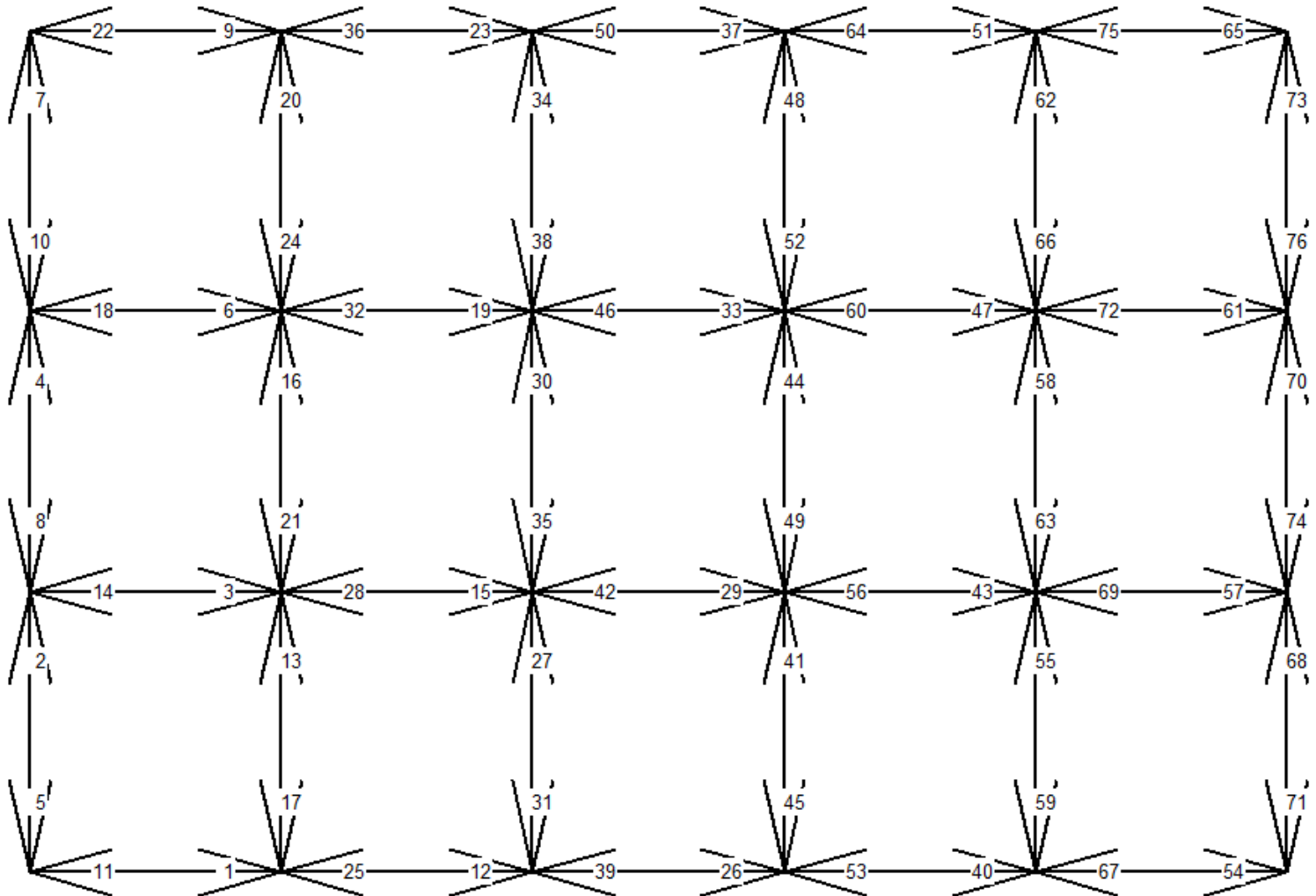
■ Previous Dart ρ

- `cm.prev(cm.next) = 1:num_darts`

■ Some Examples (3x3):



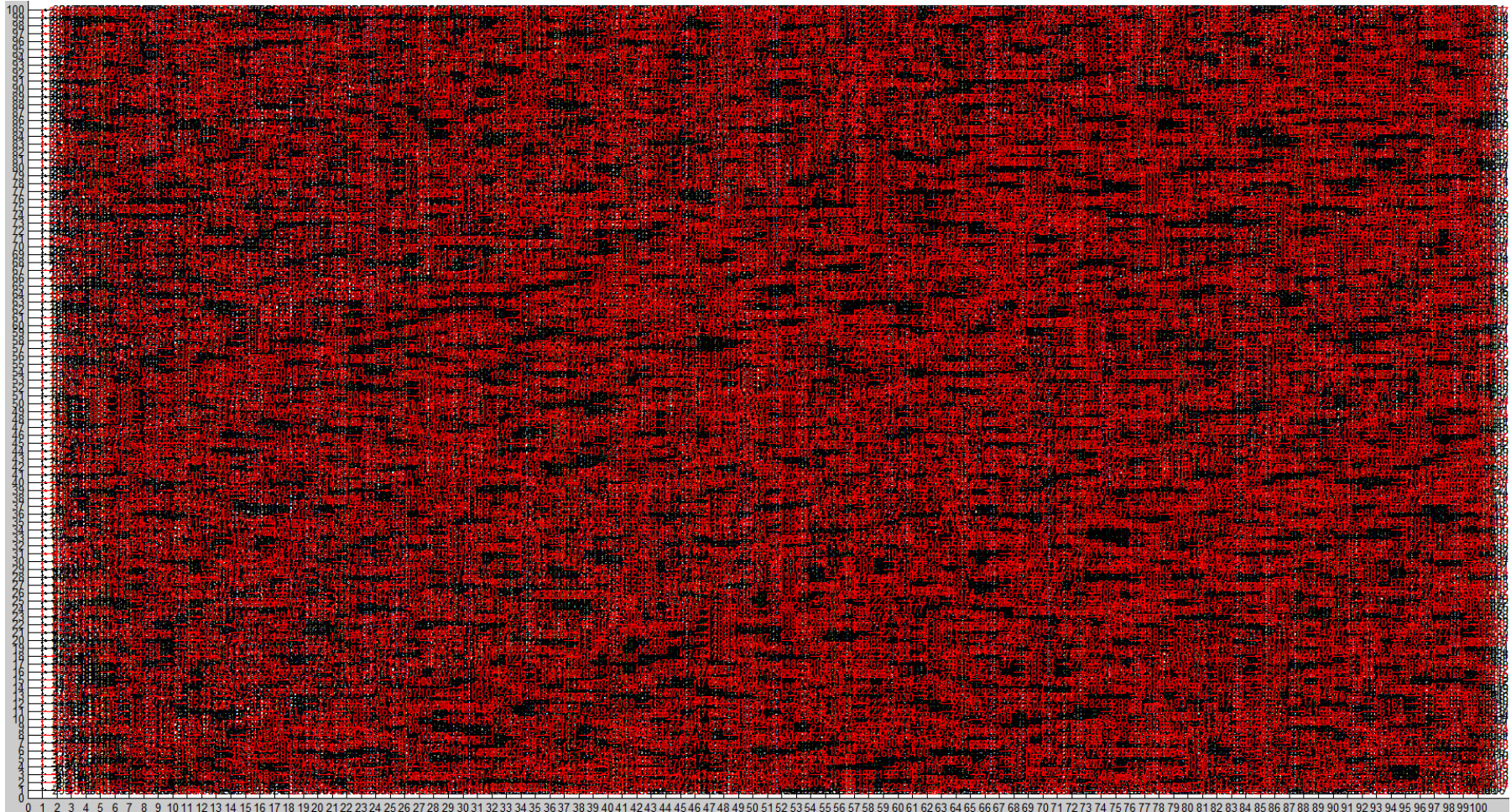
■ Some Examples (4x6):



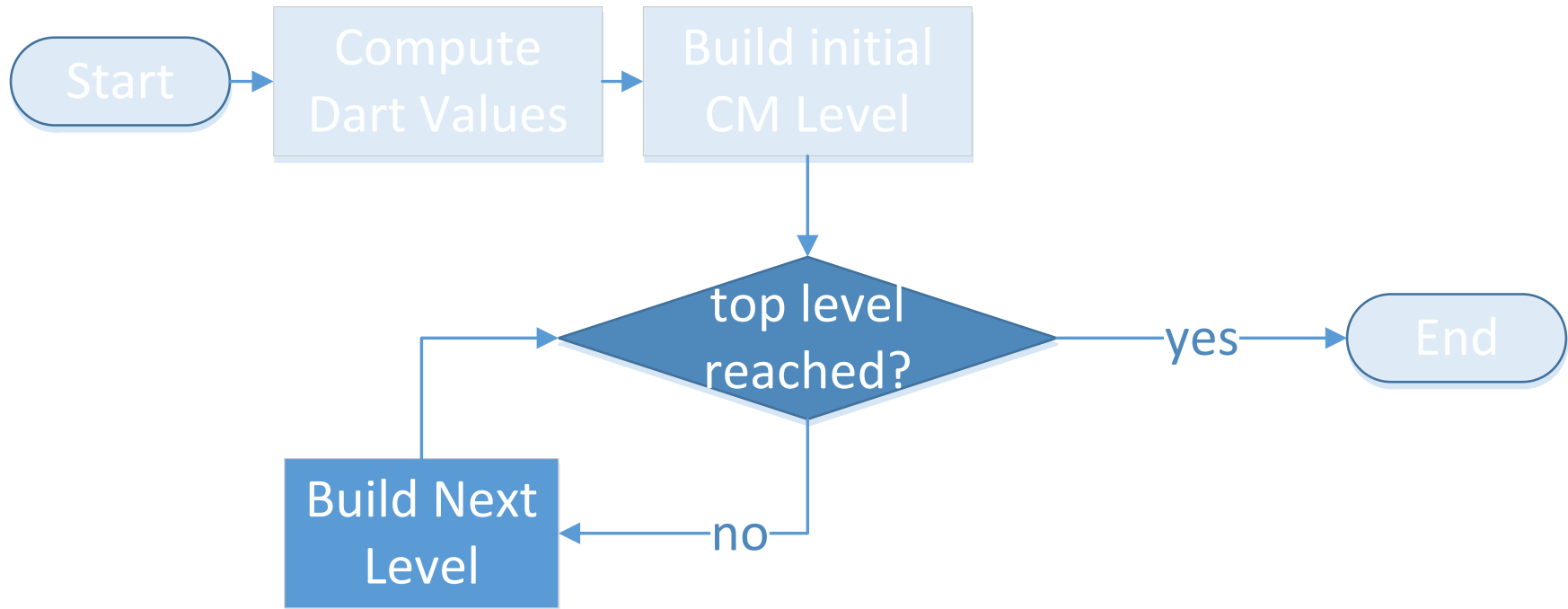
■ Some Examples (20x20):

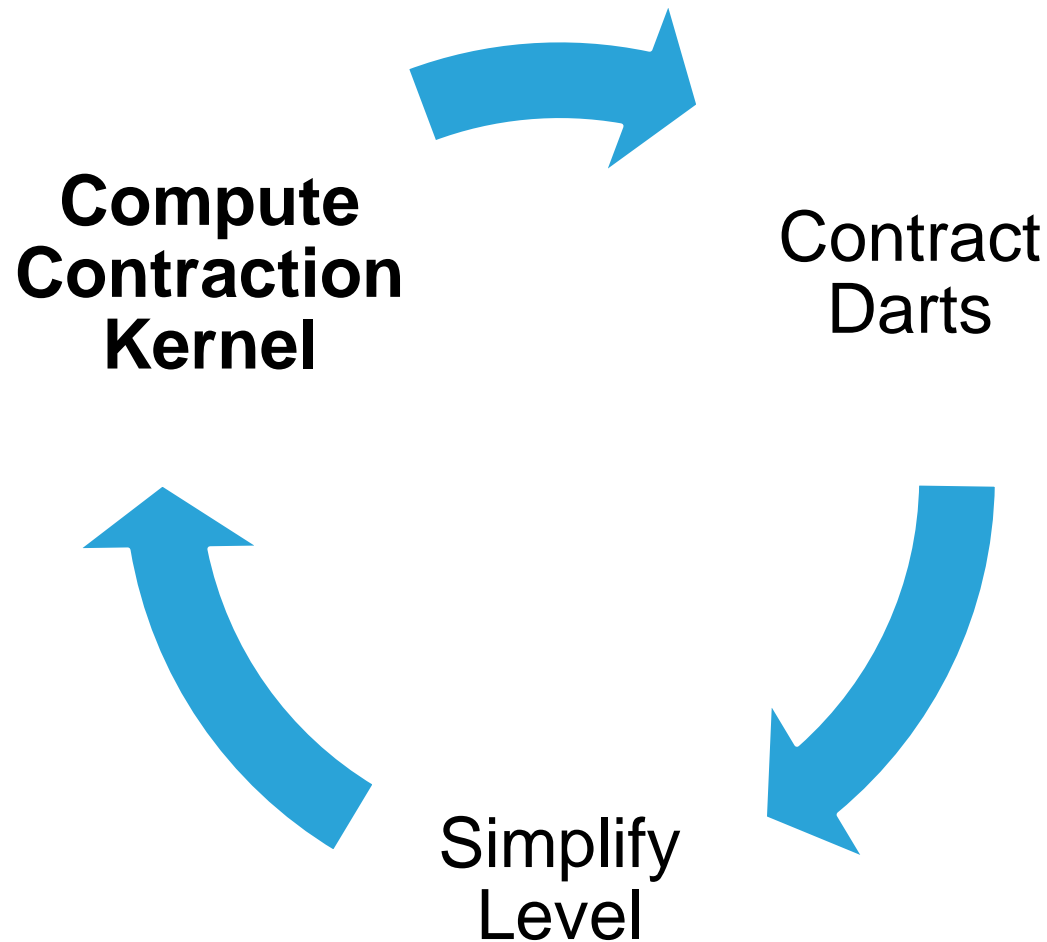
55	134	57	132	212	135	290	213	368	291	366	446	369	444	524	447	522	602	525	600	680	603	758	681	756	836	759	834	914	837	912	992	915	1070	993	1148	1071	1226	1149	1304	1227	1382	1305	1460	1383	1519	1461	1517		
58	130	54	136	208	137	214	286	205	292	364	287	362	442	365	440	520	443	598	521	596	674	599	675	754	677	752	832	755	838	910	833	908	988	911	1066	989	1144	1067	1222	1145	1300	1223	1378	1301	1456	1379	1516	1457	1520
52	128	52	128	206	137	206	284	205	284	362	287	362	440	365	440	520	443	598	521	596	674	599	675	754	677	752	832	755	838	910	833	908	988	911	1066	989	1144	1067	1222	1145	1300	1223	1378	1301	1456	1379	1516	1457	1520
56	126	51	133	204	127	202	282	205	280	360	283	358	438	361	436	516	439	594	517	592	672	595	670	750	673	748	828	751	826	906	829	904	984	907	1062	985	1140	1063	1218	1141	1296	1219	1374	1297	1452	1375	1513	1453	1518
49	126	51	133	204	127	202	282	205	280	360	283	358	438	361	436	516	439	594	517	592	672	595	670	750	673	748	828	751	826	906	829	904	984	907	1062	985	1140	1063	1218	1141	1296	1219	1374	1297	1452	1375	1513	1453	1518
53	122	48	129	200	123	207	278	201	276	356	279	354	434	357	432	512	435	590	513	588	668	591	666	746	669	744	824	747	822	902	825	900	980	903	1058	981	1136	1059	1214	1133	1292	1215	1370	1295	1448	1371	1510	1449	1515
46	120	46	120	198	123	198	276	201	276	354	279	354	434	357	432	512	435	590	513	588	668	591	666	746	669	744	824	747	822	902	825	900	980	903	1058	981	1136	1059	1214	1133	1292	1215	1370	1295	1448	1371	1510	1449	1515
50	118	45	125	196	119	203	274	197	272	352	275	350	430	353	428	508	431	586	509	584	664	587	662	742	665	740	820	743	818	898	821	896	976	899	1054	977	1132	1055	1210	1133	1288	1211	1366	1289	1444	1367	1507	1445	1505
43	118	45	125	196	119	203	274	197	272	352	275	350	430	353	428	508	431	586	509	584	664	587	662	742	665	740	820	743	818	898	821	896	976	899	1054	977	1132	1055	1210	1133	1288	1211	1366	1289	1444	1367	1507	1445	1505
47	114	42	121	192	115	199	270	193	268	348	271	346	426	349	424	504	427	582	505	580	660	583	655	738	661	736	816	739	814	894	817	892	972	895	1050	973	1128	1051	1206	1129	1284	1207	1362	1285	1440	1363	1504	1441	1509
40	114	42	121	192	115	199	270	193	268	348	271	346	426	349	424	504	427	582	505	580	660	583	655	738	661	736	816	739	814	894	817	892	972	895	1050	973	1128	1051	1206	1129	1284	1207	1362	1285	1440	1363	1504	1441	1509
44	110	39	117	188	111	195	266	189	264	344	267	342	422	345	420	500	423	578	501	576	656	579	654	734	657	732	812	735	810	890	813	888	968	891	1046	969	1124	1047	1202	1125	1280	1203	1358	1281	1436	1359	1501	1437	1499
37	110	39	117	188	111	195	266	189	264	344	267	342	422	345	420	500	423	578	501	576	656	579	654	734	657	732	812	735	810	890	813	888	968	891	1046	969	1124	1047	1202	1125	1280	1203	1358	1281	1436	1359	1501	1437	1499
41	106	36	113	184	107	191	262	185	259	340	263	338	418	341	416	496	419	574	497	572	652	575	650	730	653	728	808	731	806	886	809	884	964	887	1042	965	1120	1043	1198	1121	1276	1199	1354	1277	1432	1355	1498	1433	1503
34	106	36	113	184	107	191	262	185	259	340	263	338	418	341	416	496	419	574	497	572	652	575	650	730	653	728	808	731	806	886	809	884	964	887	1042	965	1120	1043	1198	1121	1276	1199	1354	1277	1432	1355	1498	1433	1503
38	102	33	109	180	103	187	258	181	256	336	259	334	414	337	412	492	415	570	493	568	648	571	646	726	649	724	804	727	802	882	805	880	960	883	1038	961	1116	1039	1194	1117	1272	1199	1350	1273	1428	1351	1495	1429	1493
31	102	33	109	180	103	187	258	181	256	336	259	334	414	337	412	492	415	570	493	568	648	571	646	726	649	724	804	727	802	882	805	880	960	883	1038	961	1116	1039	1194	1117	1272	1199	1350	1273	1428	1351	1495	1429	1493
35	98	30	105	176	99	183	254	177	252	332	255	330	410	333	408	488	411	566	489	564	644	567	642	722	645	720	800	723	798	878	801	885	956	879	1034	957	1112	1035	1190	1113	1268	1191	1346	1269	1424	1347	1492	1425	1497
28	98	30	105	176	99	183	254	177	252	332	255	330	410	333	408	488	411	566	489	564	644	567	642	722	645	720	800	723	798	878	801	885	956	879	1034	957	1112	1035	1190	1113	1268	1191	1346	1269	1424	1347	1492	1425	1497
32	94	27	101	172	95	179	250	173	248	328	251	326	406	329	404	484	407	562	485	560	640	563	638	718	641	716	796	719	794	874	797	872	952	875	1030	953	1108	1031	1186	1109	1264	1187	1342	1265	1420	1343	1489	1421	1487
25	94	27	101	172	95	179	250	173	248	328	251	326	406	329	404	484	407	562	485	560	640	563	638	718	641	716	796	719	794	874	797	872	952	875	1030	953	1108	1031	1186	1109	1264	1187	1342	1265	1420	1343	1489	1421	1487
29	90	24	97	168	91	175	246	169	244	324	247	322	402	325	400	480	403	558	481	556	636	559	634	714	637	712	792	715	790	870	793	868	948	871	1026	949	1104	1027	1182	1105	1260	1183	1338	1261	1416	1339	1486	1417	1484
22	90	24	97	168	91	175	246	169	244	324	247	322	402	325	400	480	403	558	481	556	636	559	634	714	637	712	792	715	790	870	793	868	948	871	1026	949	1104	1027	1182	1105	1260	1183	1338	1261	1416	1339	1486	1417	1484
26	86	21	93	164	87	171	242	165	240	320	243	318	398	321	396	476	399	554	477	552	632	555	630	710	633	708	788	711	786	866	789	864	944	867	1022	945	1100	1023	1178	1101	1256	1179	1334	1257	1412	1335	1483	1413	1481
19	86	21	93	164	87	171	242	165	240	320	243	318	398	321	396	476	399	554	477	552	632	555	630	710	633	708	788	711	786	866	789	864	944	867	1022	945	1100	1023	1178	1101	1256	1179	1334	1257	1412	1335	1483	1413	1481
23	82	18	89	160	83	167	238	161	236	316	239	314	394	317	392	472	395	550	473	548	628	551	626	706	629	704	784	707	782	862	785	856	940	863	1018	941	1096	1019	1174	1097	1252	1175	1330	1253	1408	1331	1480	1409	1478
16	82	18	89	160	83	167	238	161	236	316	239	314	394	317	392	472	395	550	473	548	628	551	626	706	629	704	784	707	782	862	785	856	940	863	1018	941	1096	1019	1174	1097	1252	1175	1330	1253	1408	1331	1480	1409	1478
20	78	15	85	156	79	153	234	157	232	312	235	310	390	313	388	468	391	546	469	544	624	547	622	702	625	700	780	703	778	858	781	856	936	859	1014	937	1092	1015	1170	1093	1248	1171	1326	1249	1404	1327	1477	1405	1475
13	78	15	85	156	79	153	234	157	232	312	235	310	390	313	388	468	391	546	469	544	624	547	622	702	625	700	780	703	778	858	781	856	936	859	1014	937	1092	1015	1170	1093	1248	1171	1326	1249	1404	1327	1477	1405	1475
17	74	12	81	152	75	159	230	153	228	308	231	306	386	309	384	464	387	542	465	540	620	543	618	698	62																								

■ Some Examples (100x100):

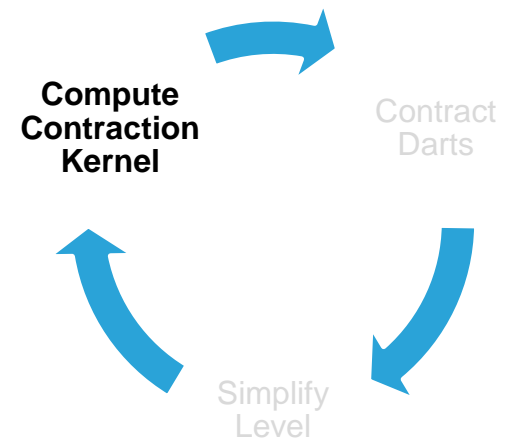


Phase 3: Implementation

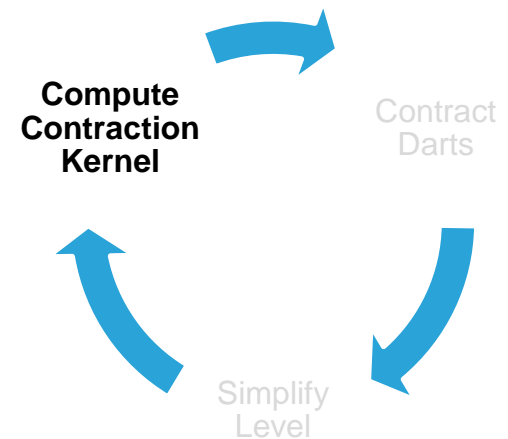
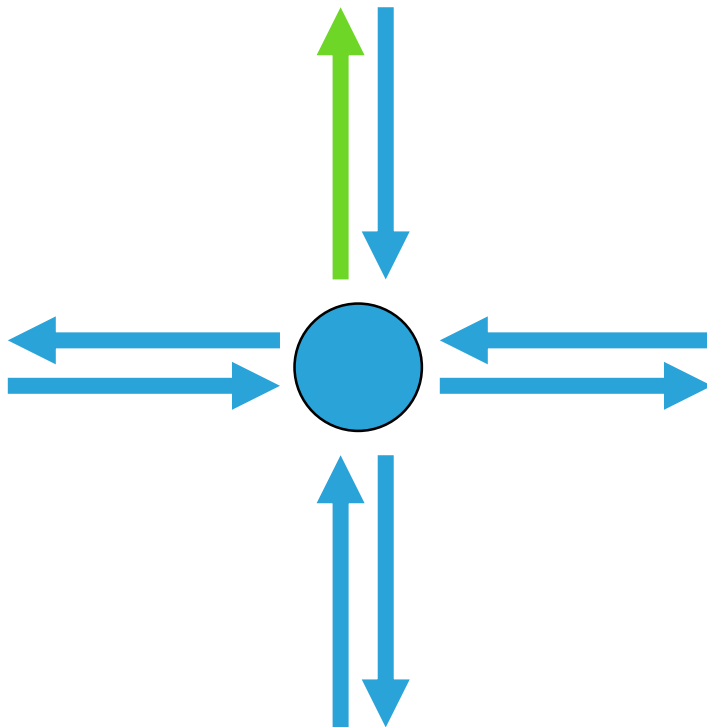


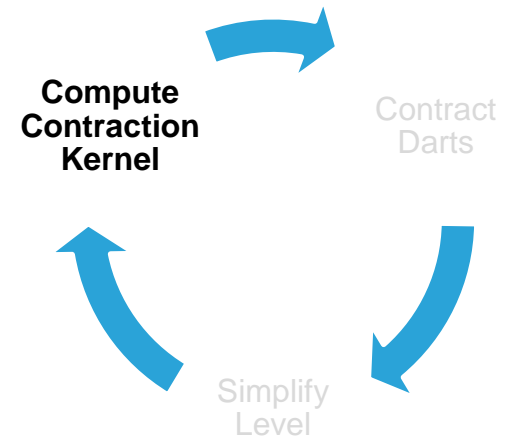
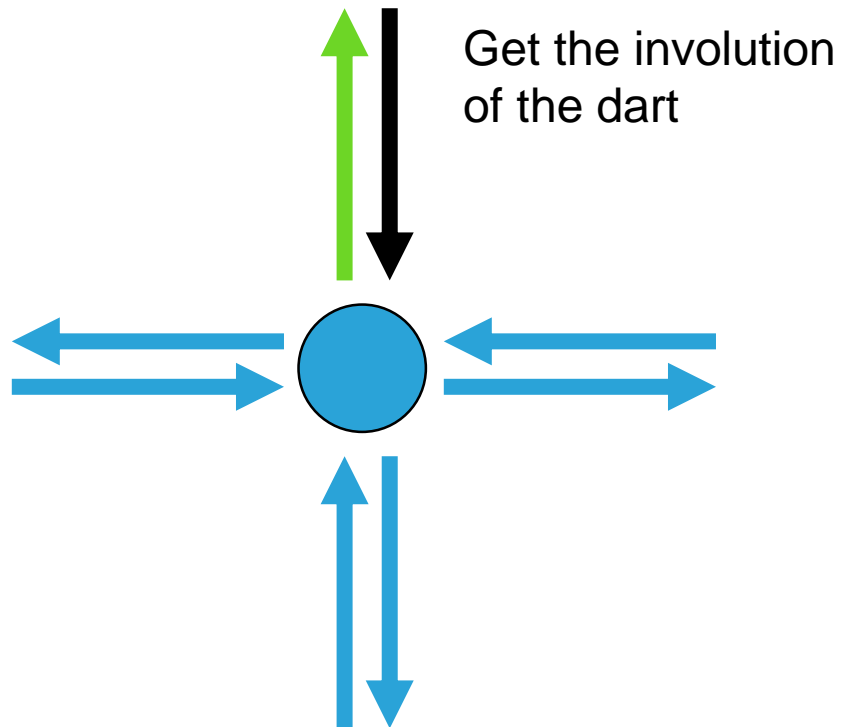


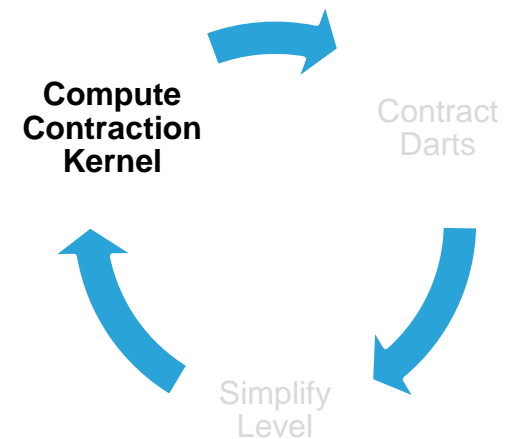
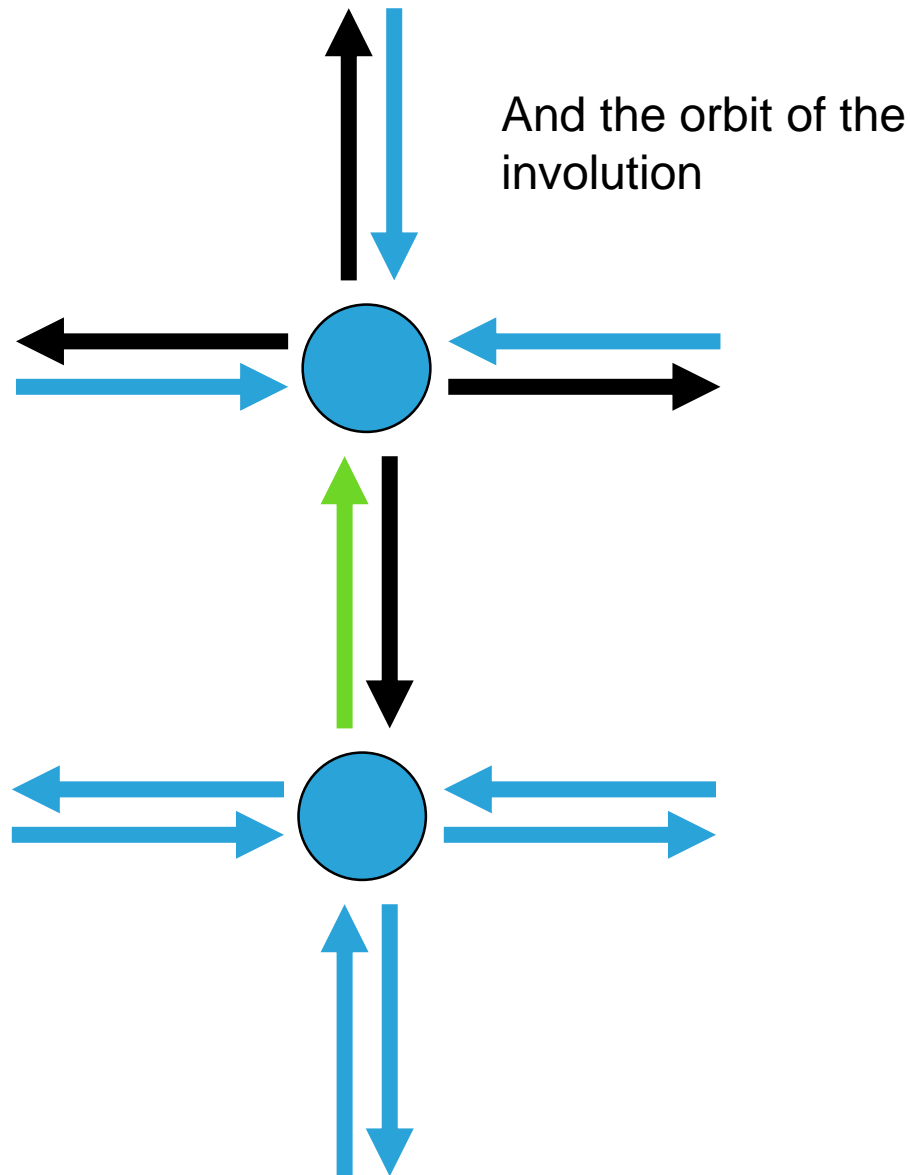
- Sort the darts by its values
- Add dart to contraction kernel if:
 - Not self loop
 - Not pending edge



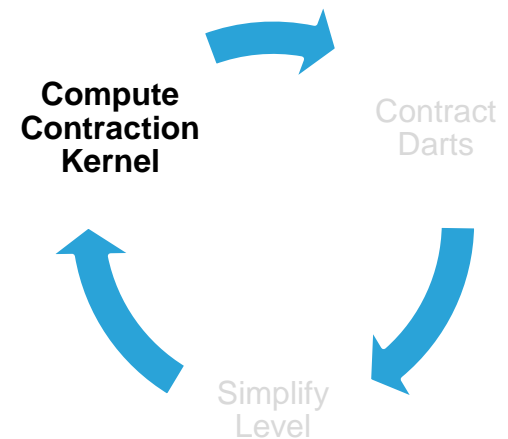
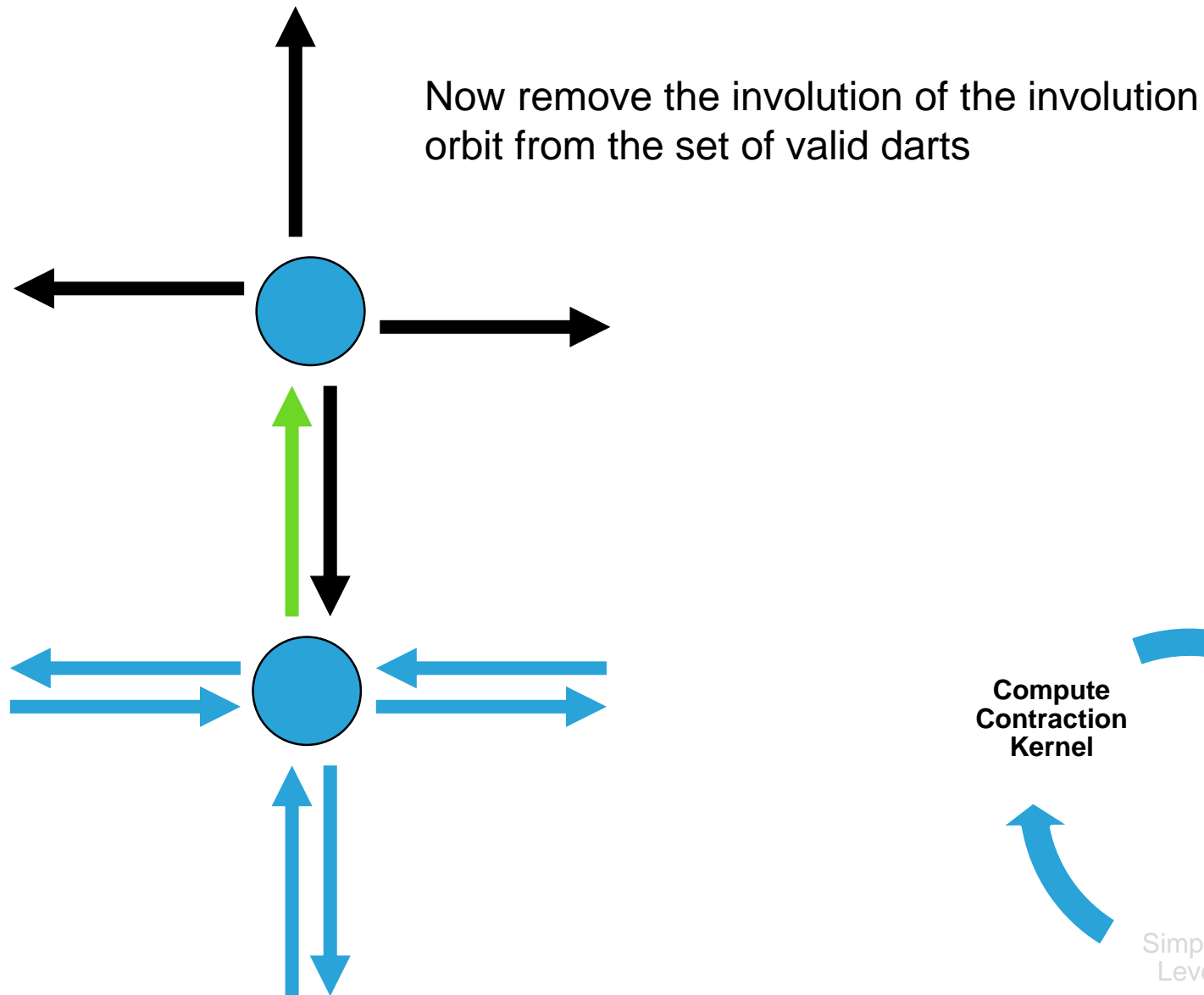
Greedy: Assign next best Dart For Contraction Kernel



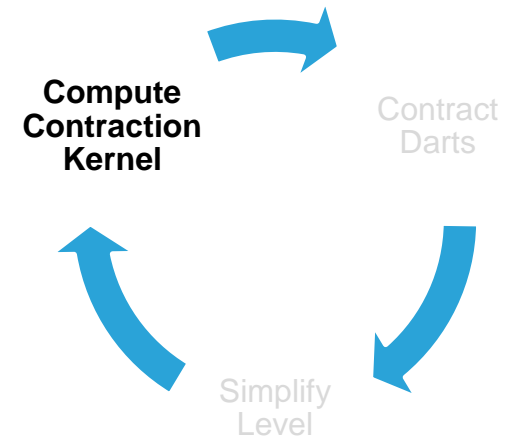
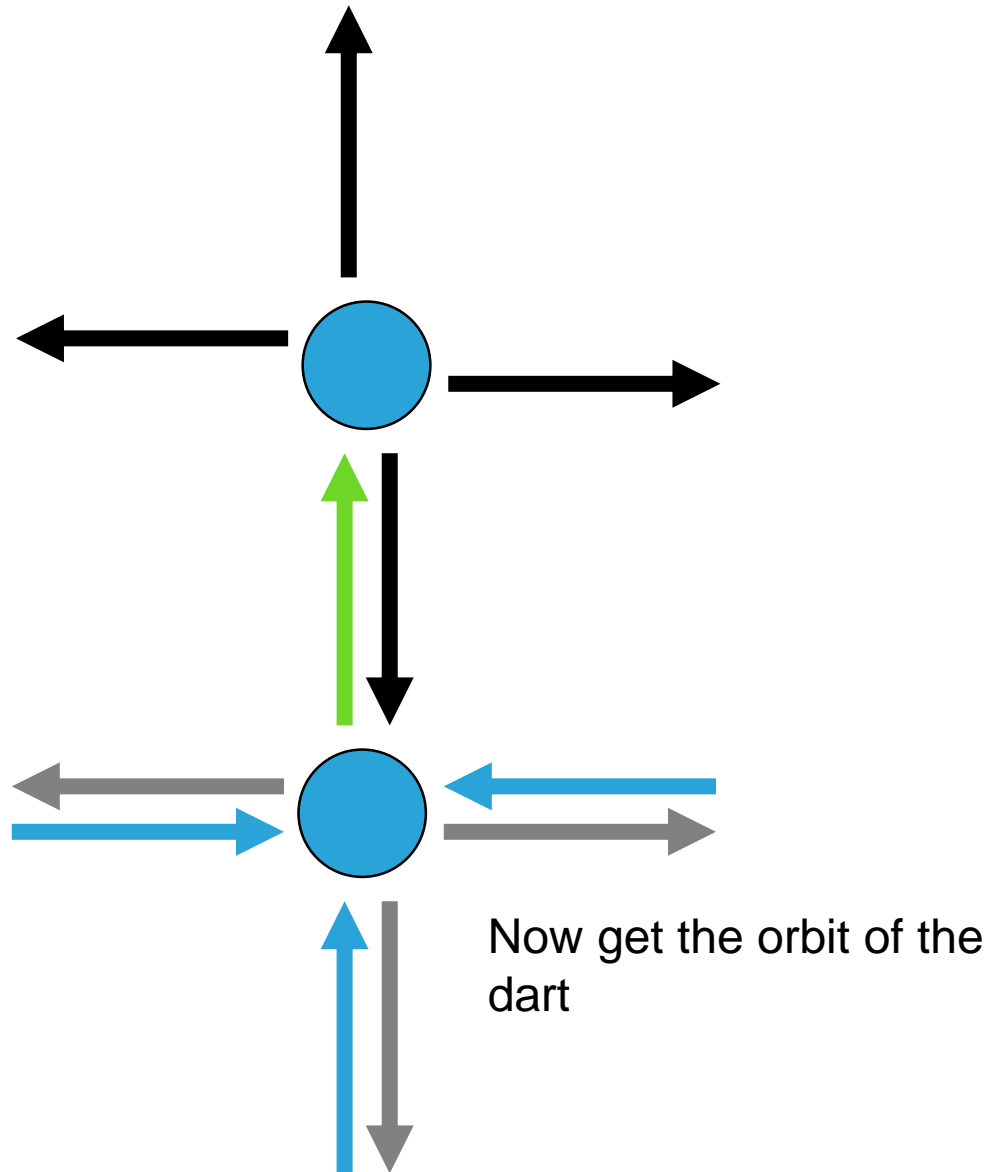




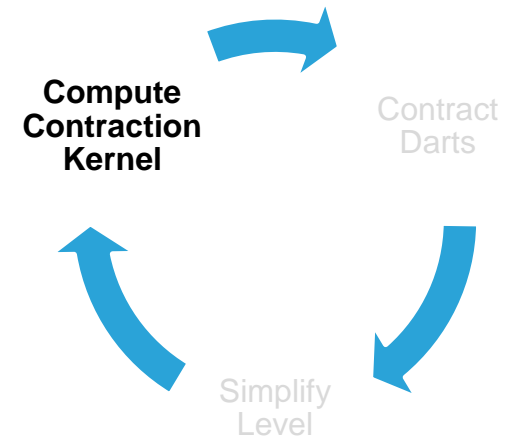
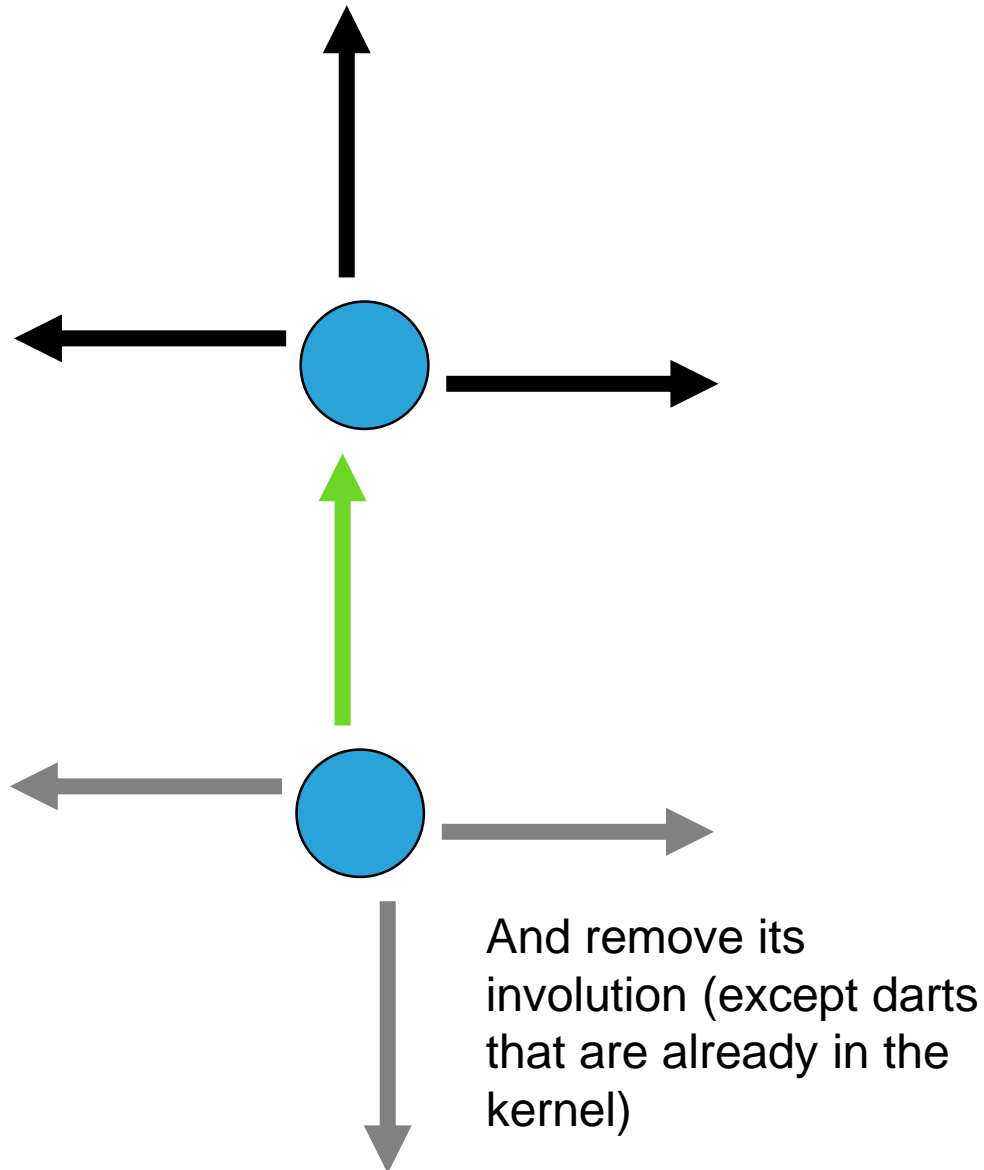
Compute the Contraction Kernel



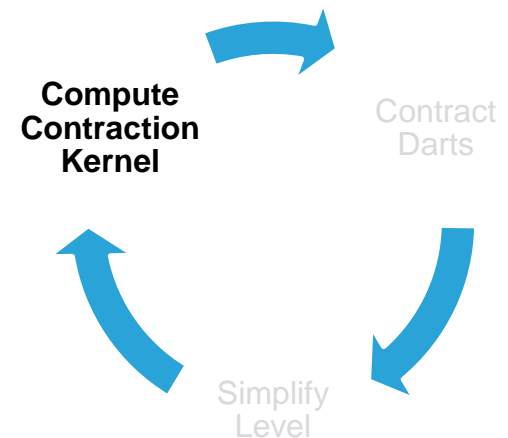
Compute the Contraction Kernel



Compute the Contraction Kernel

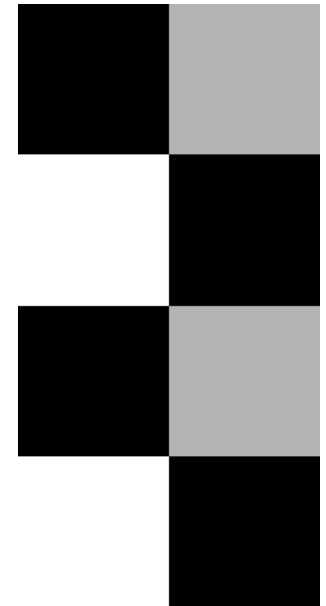
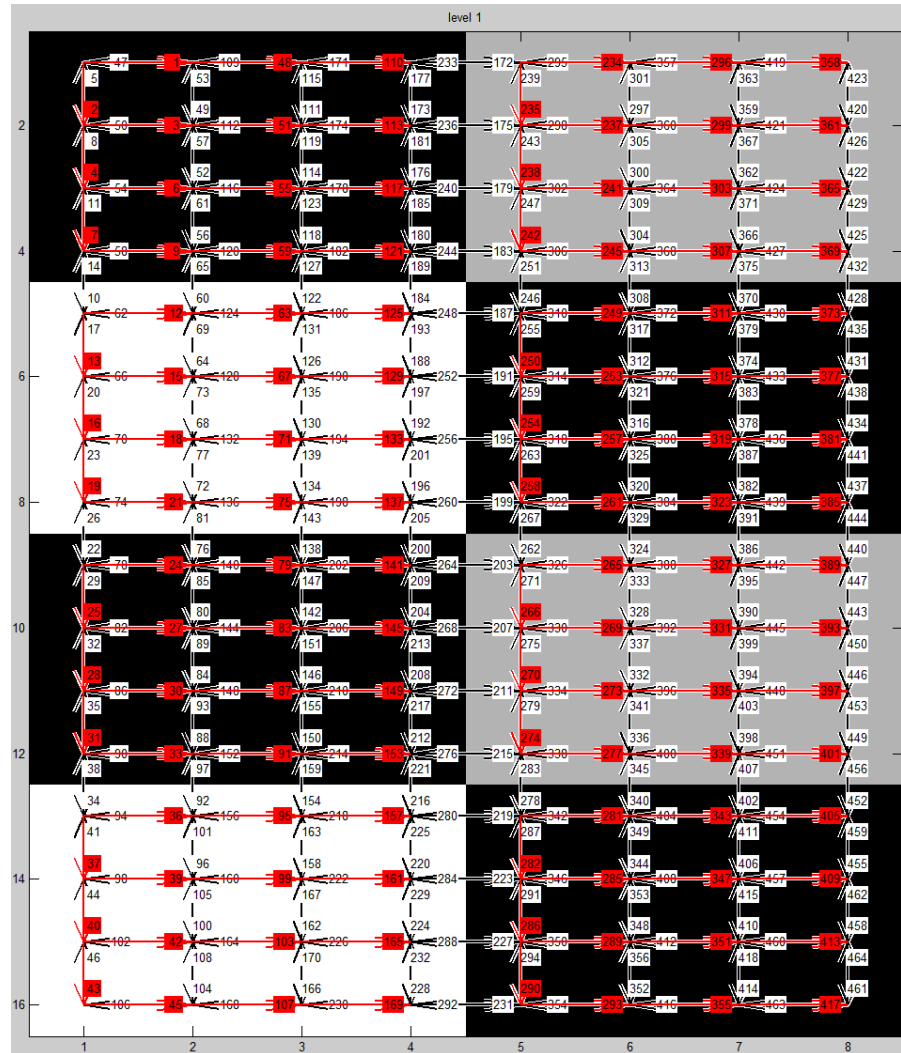


- Exceptions:
 - Self loops
 - Pending edges

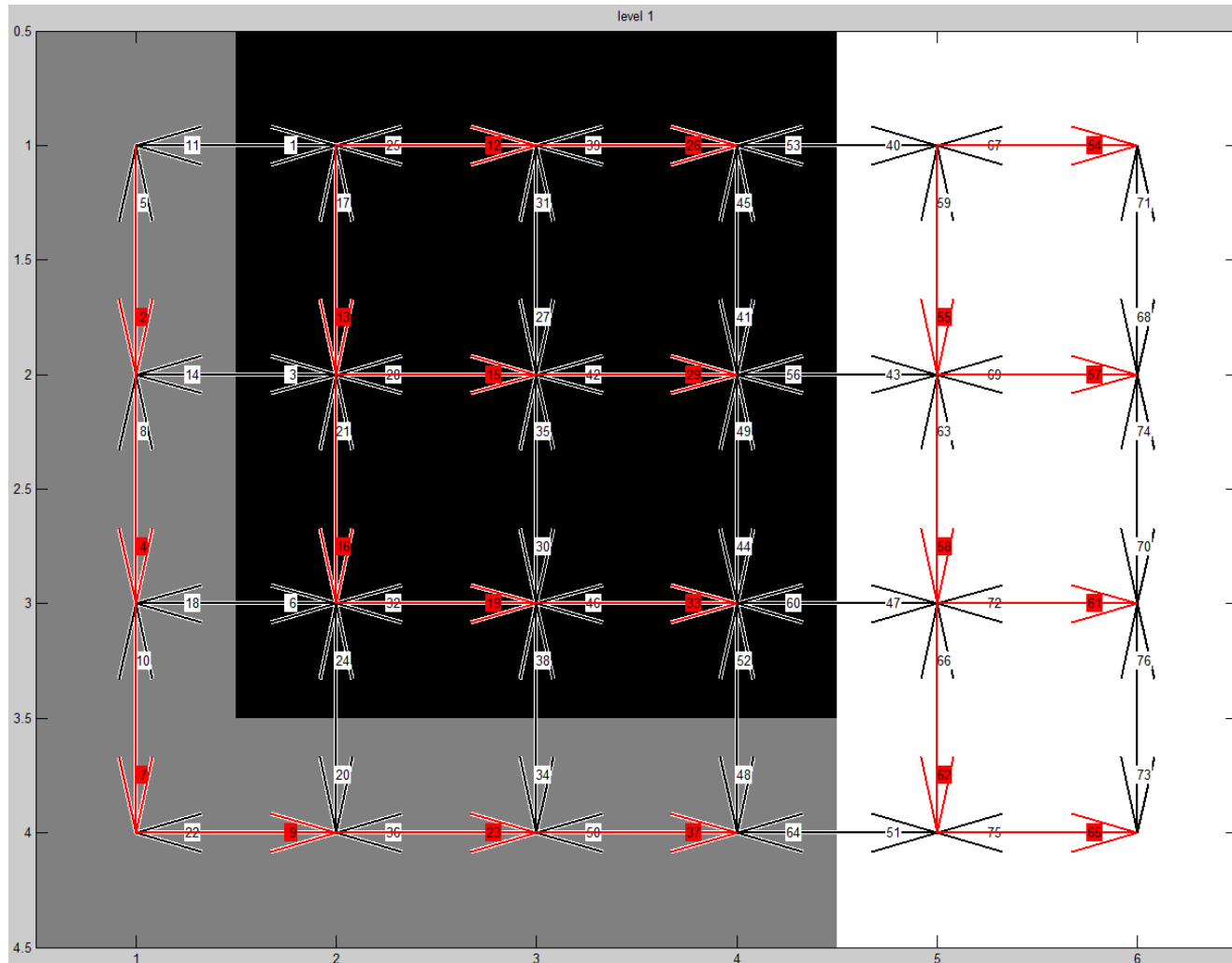


Compute the Contraction Kernel

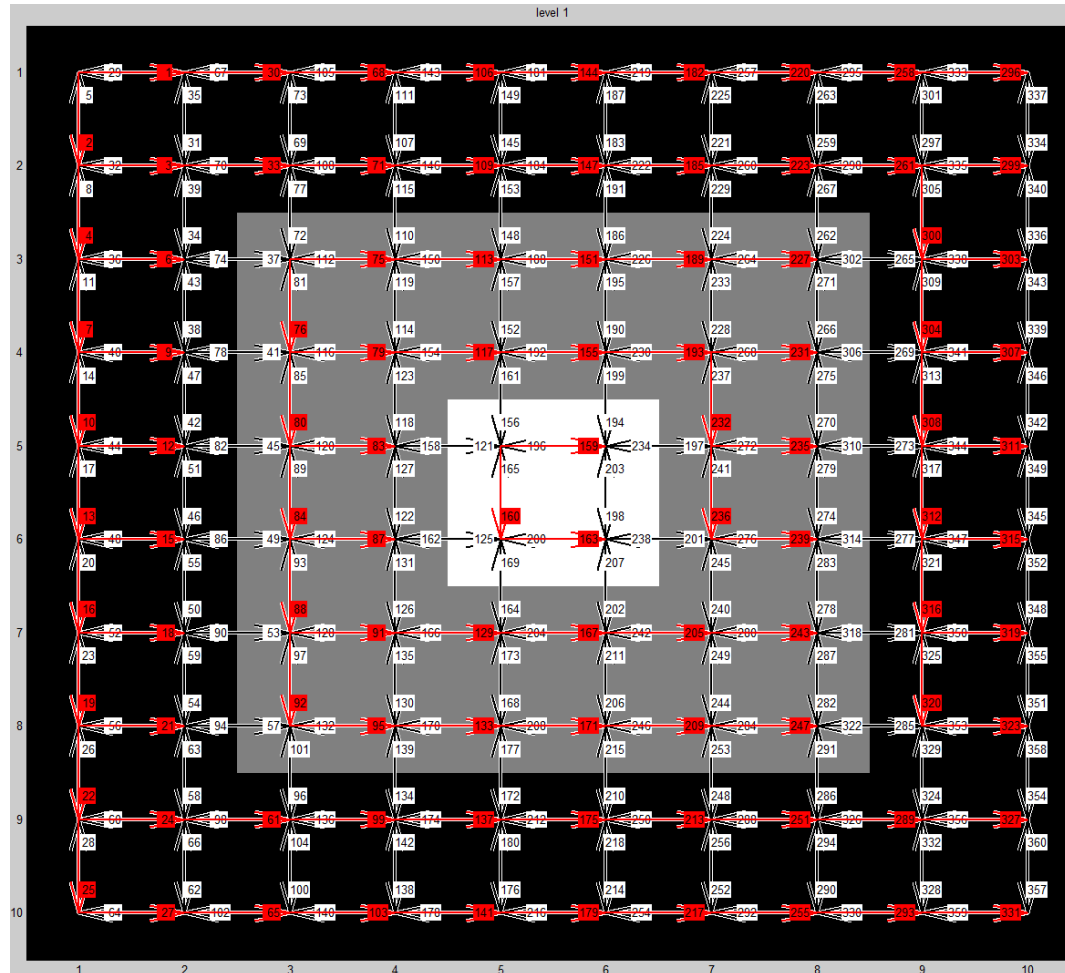
■ Examples: Checkerboard

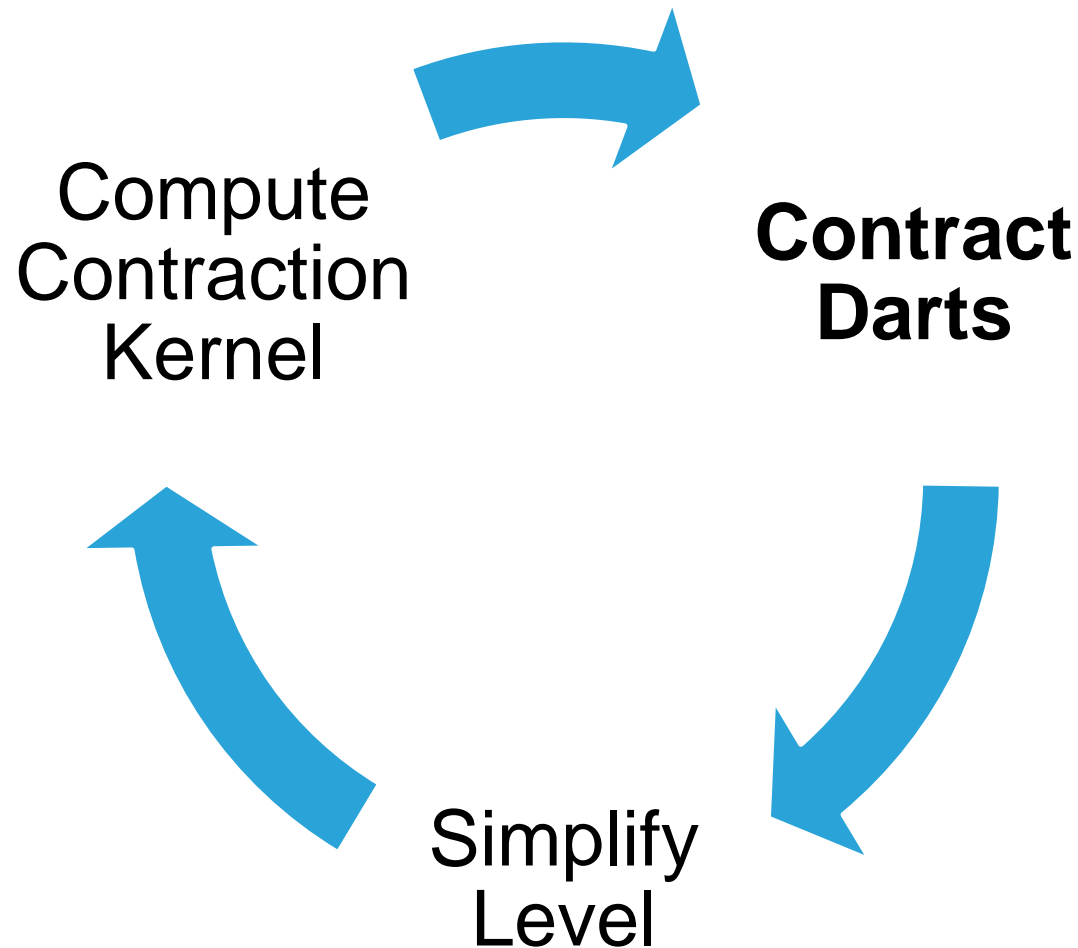


■ Examples: Lecture

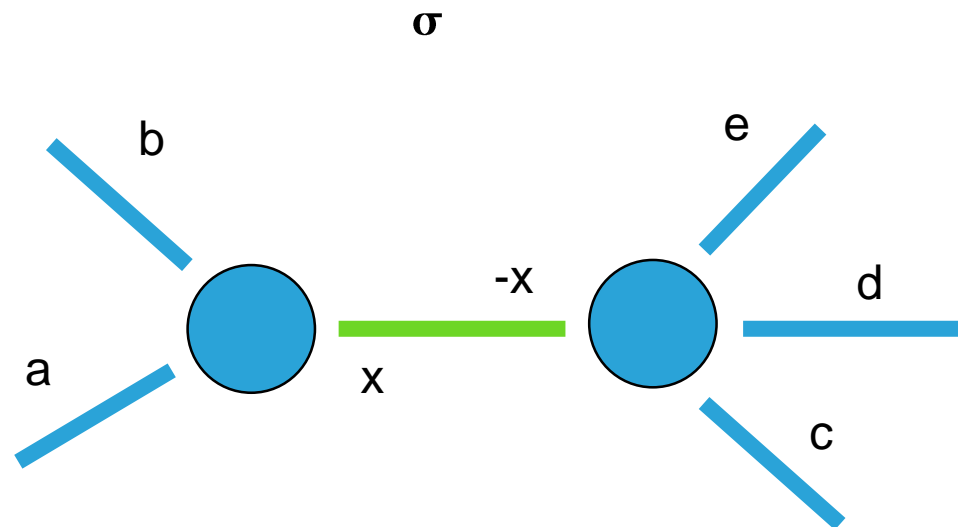


■ Examples: Enclosure

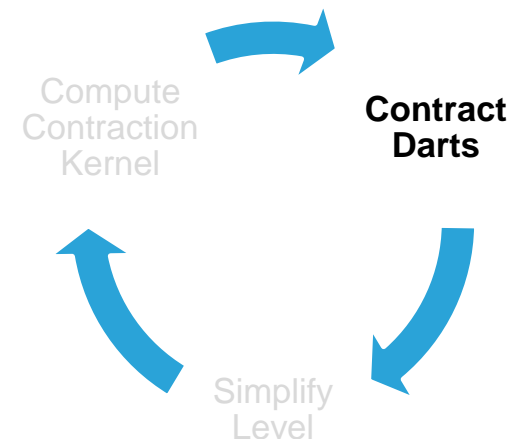




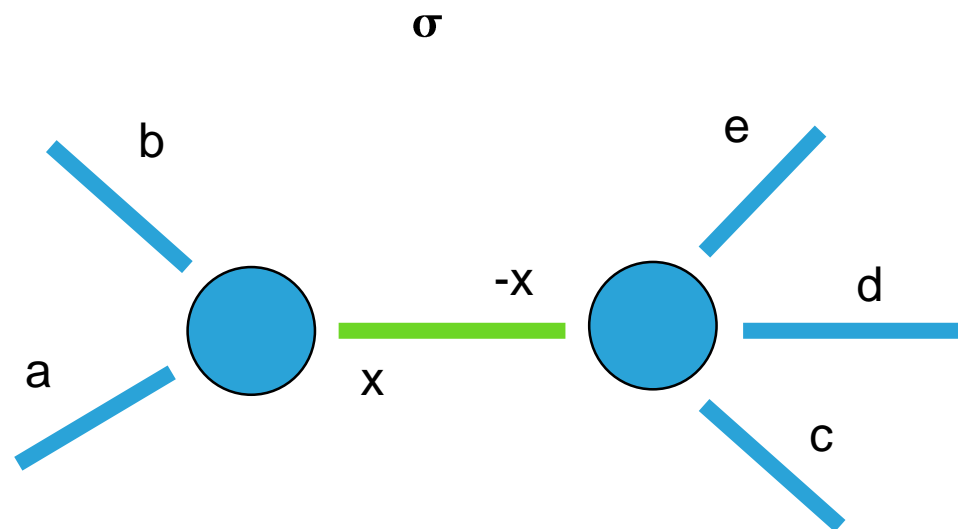
■ Recap: Contract Darts



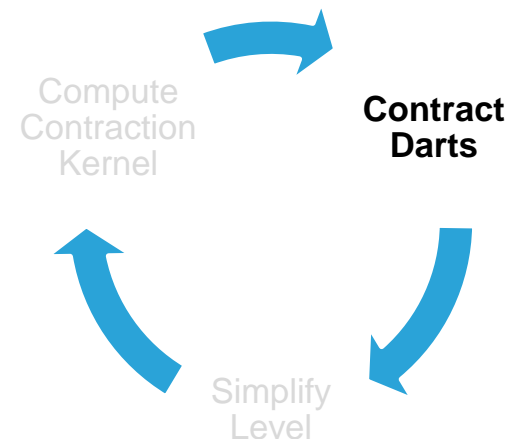
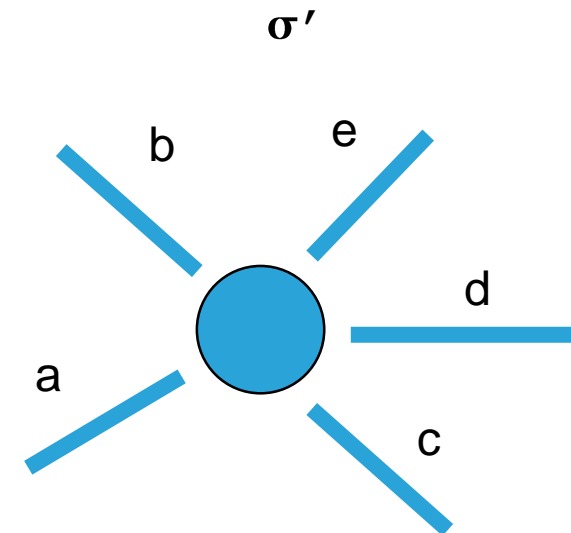
darts	a	b	c	d	e	x	-x
σ	x	a	d	e	-x	b	c
σ'	c	a	d	e	b	b	c



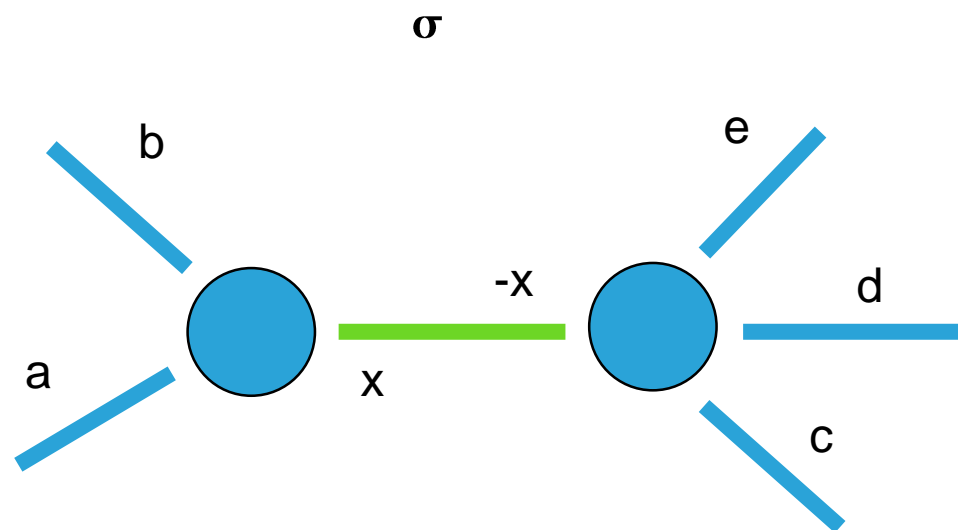
■ Recap: Contract Darts



darts	a	b	c	d	e	x	-x
σ	x	a	d	e	-x	b	c
σ'	c	a	d	e	b	b	c



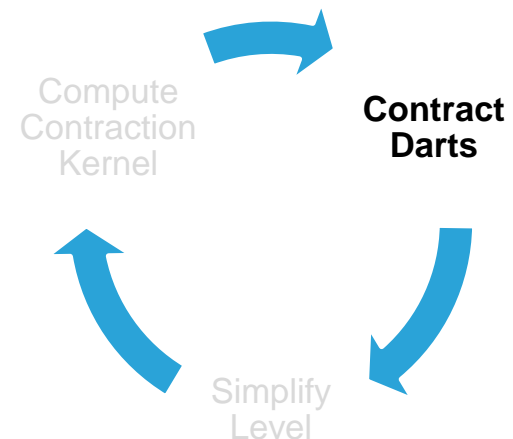
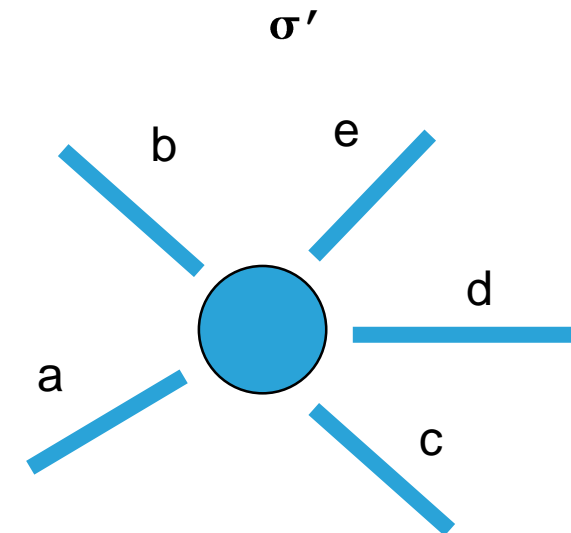
■ Recap: Contract Darts

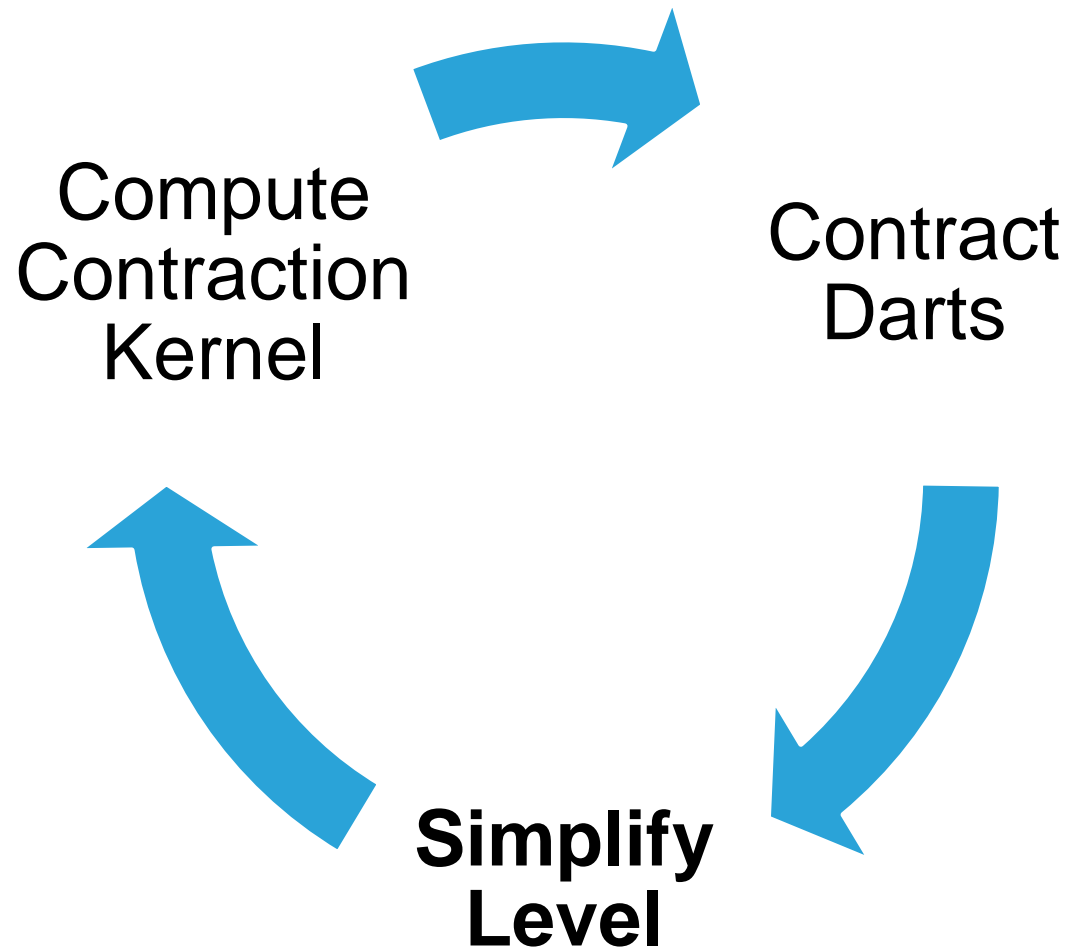


darts	a	b	c	d	e	x	-x
σ	x	a	d	e	-x	b	c
σ'	c	a	d	e	b	b	c

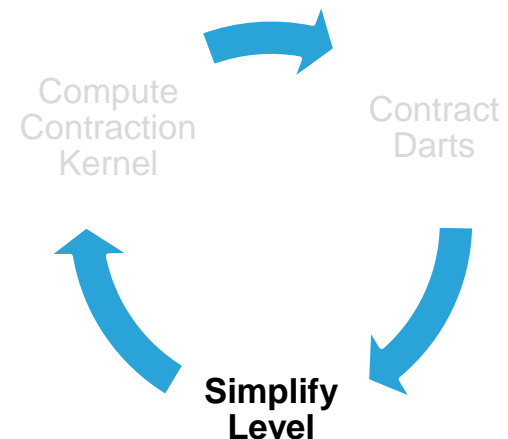
$$\sigma'(\sigma^{-1}(x)) := \sigma(-x)$$

$$\sigma'(\sigma^{-1}(-x)) := \sigma(x)$$

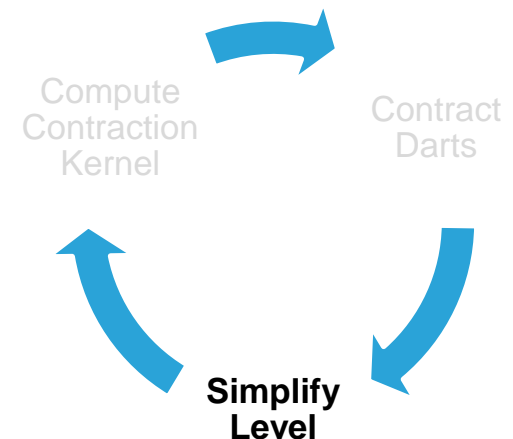
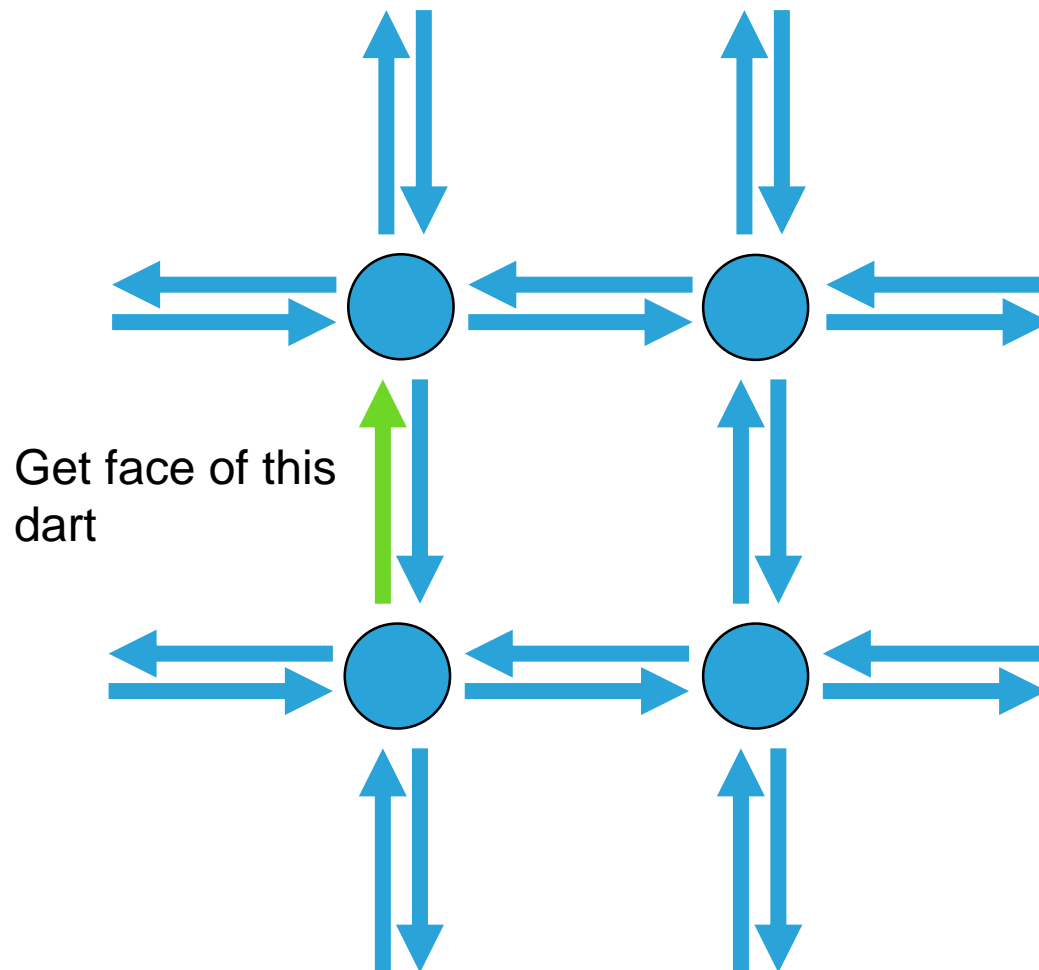




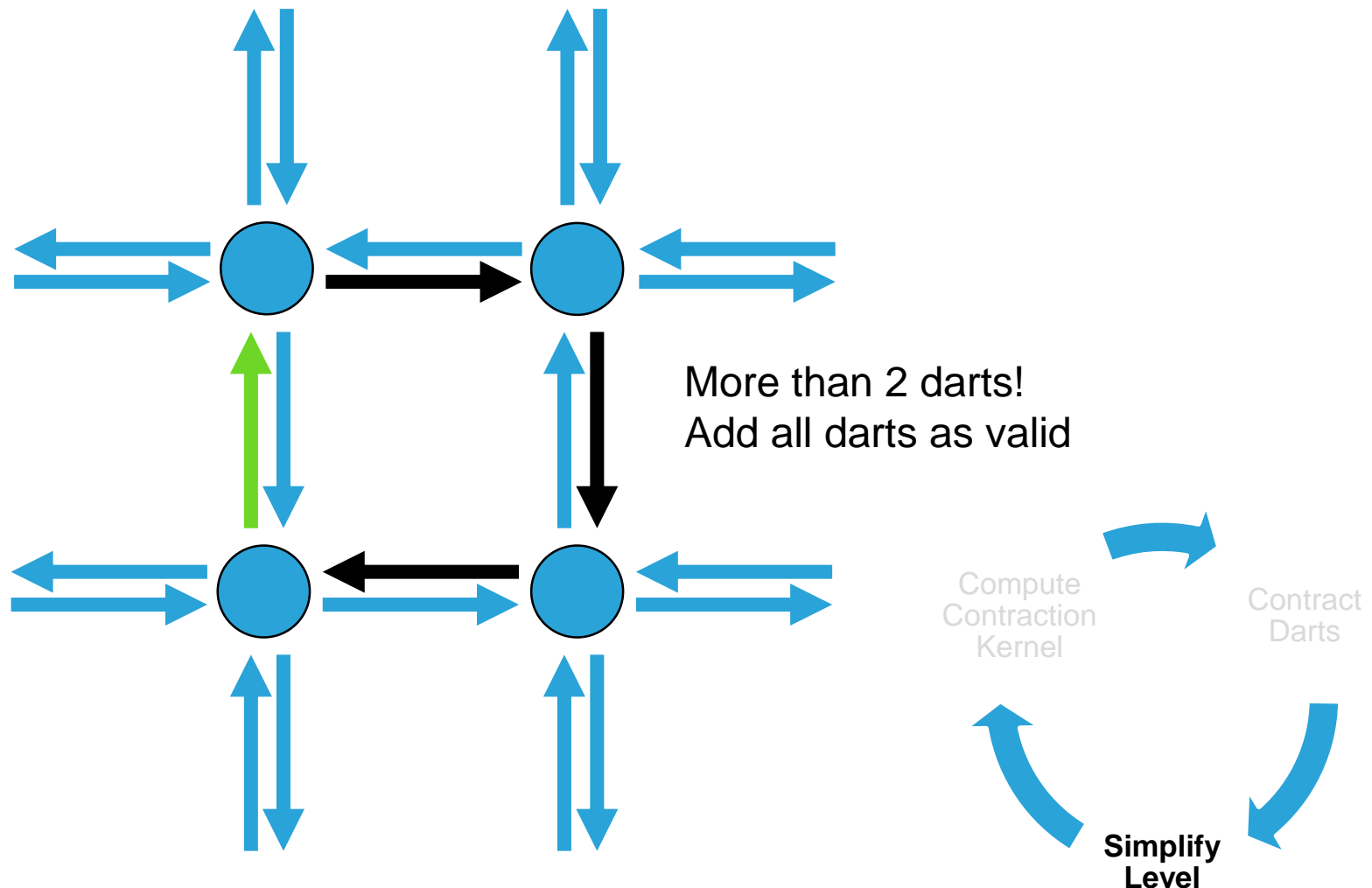
- When contracting darts double edges and self-direct-loops are created. By removing these darts the pyramid is easier to read and faster to compute.



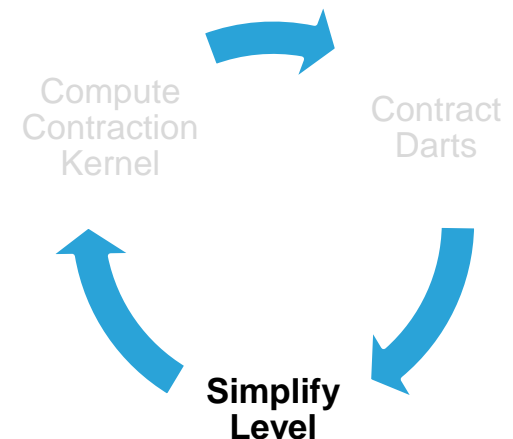
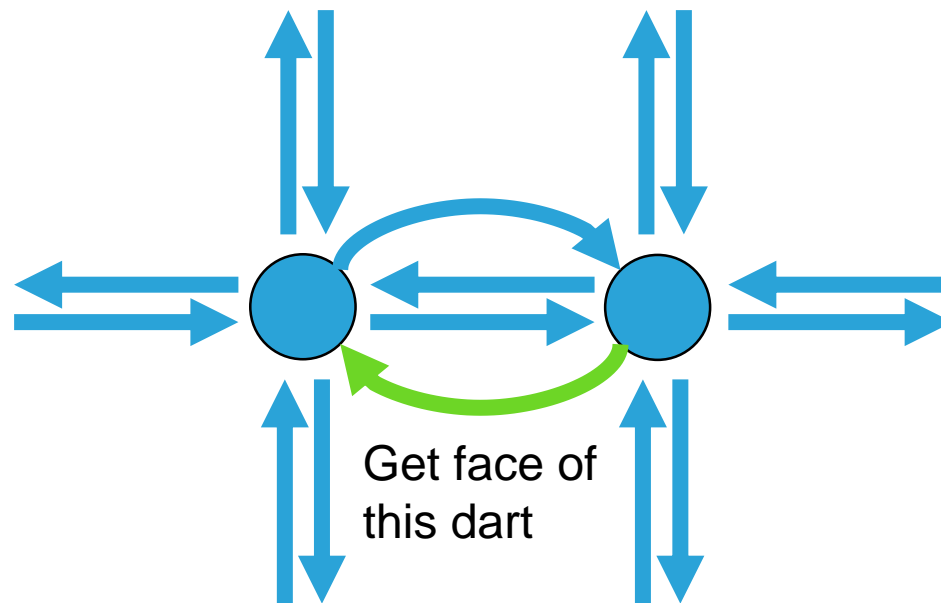
- Check if the **face of a dart** has less than 2 darts:



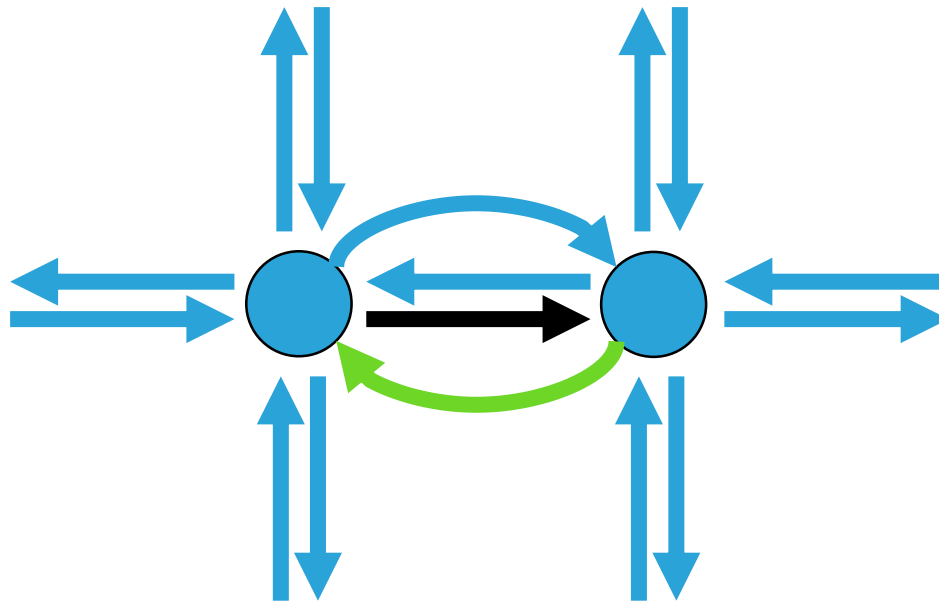
- Check if the **face of a dart** has less than 2 darts:



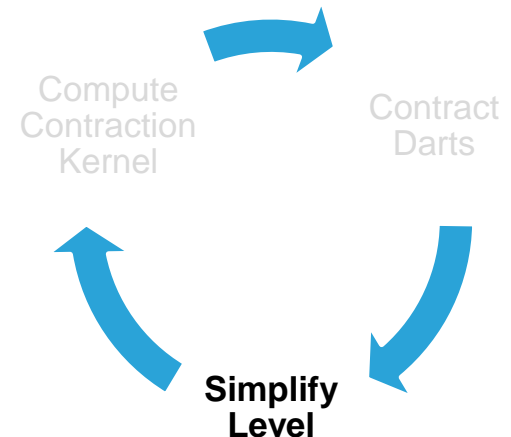
- Check if the **face of a dart** has less than 2 darts:



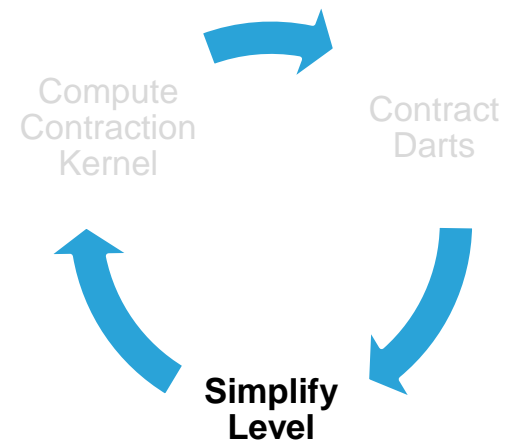
- Check if the **face of a dart** has less than 2 darts:



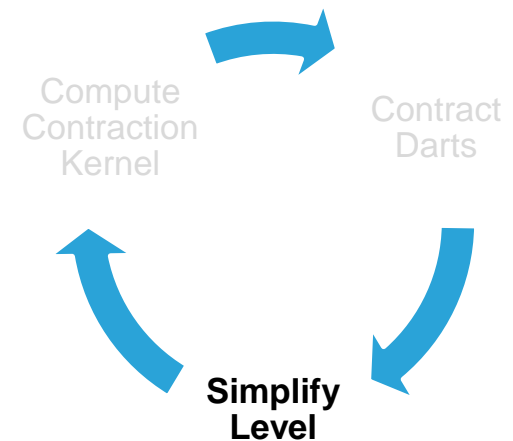
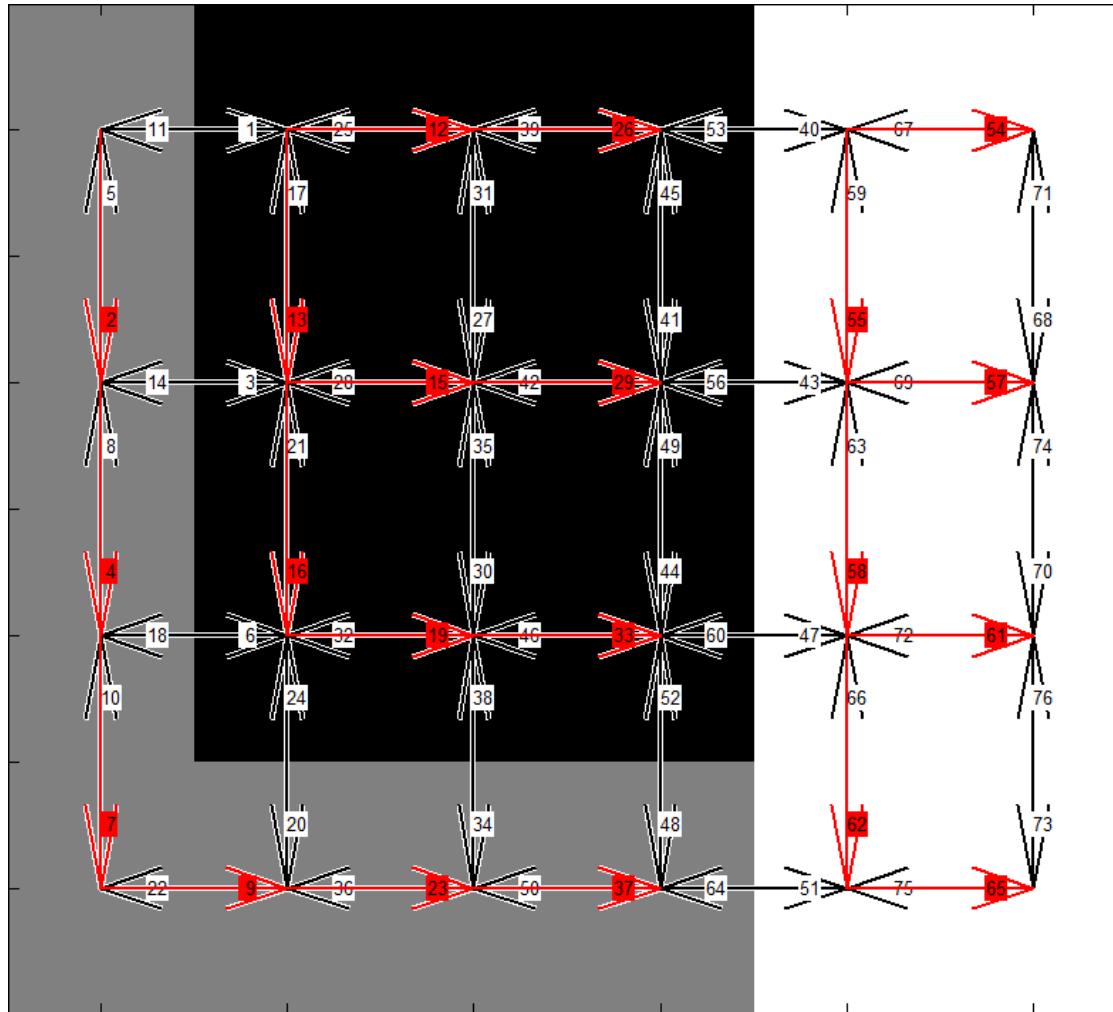
Number of darts in face is 2
→ can be removed!



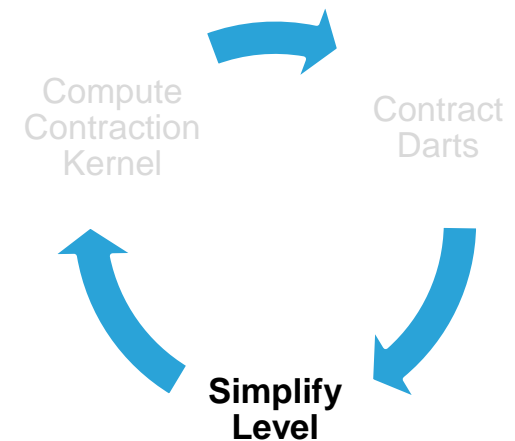
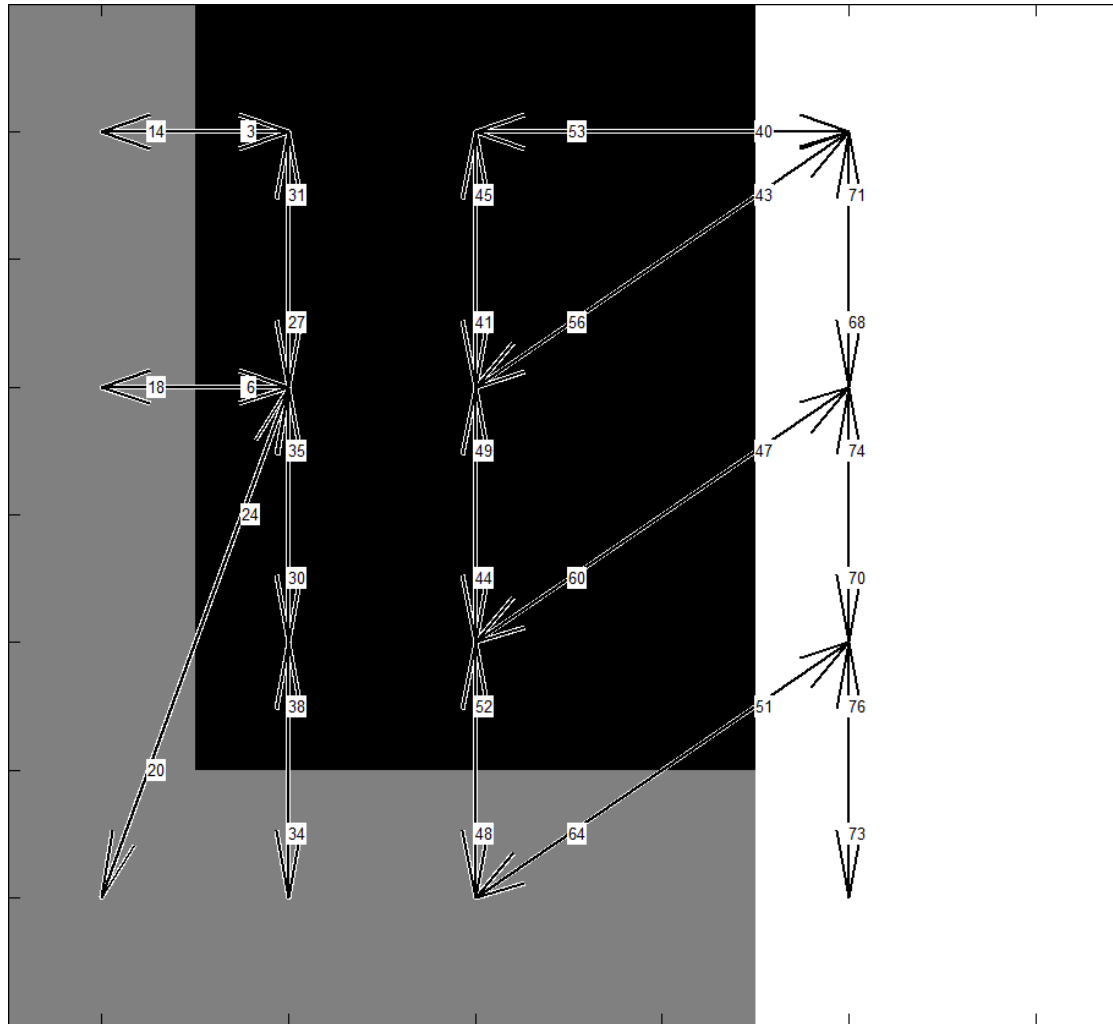
- Special cases for removal:
 - Self-Direct-Loops (2 cases)



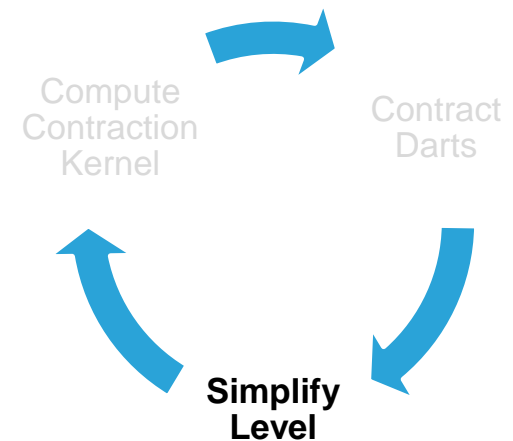
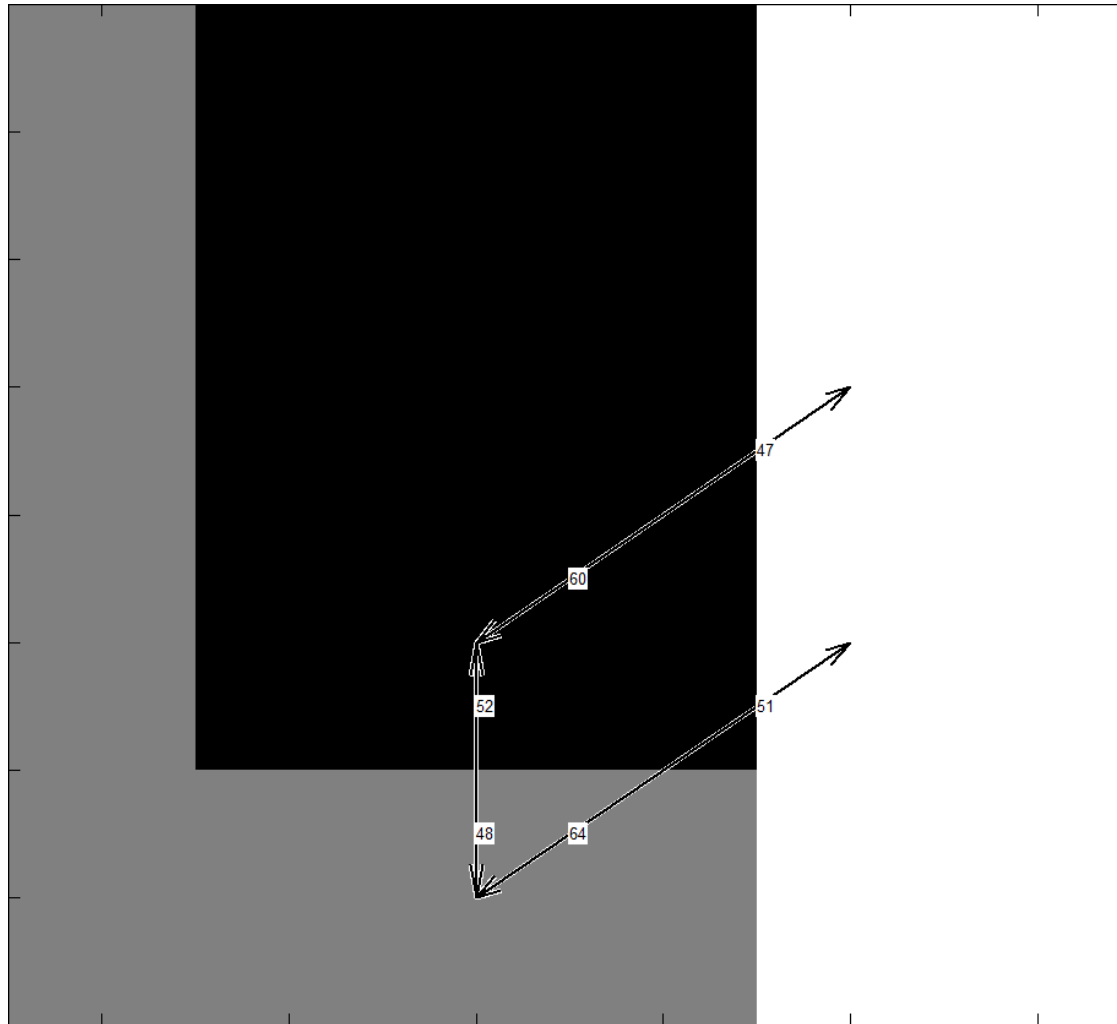
■ Example: contraction kernel (before contraction)

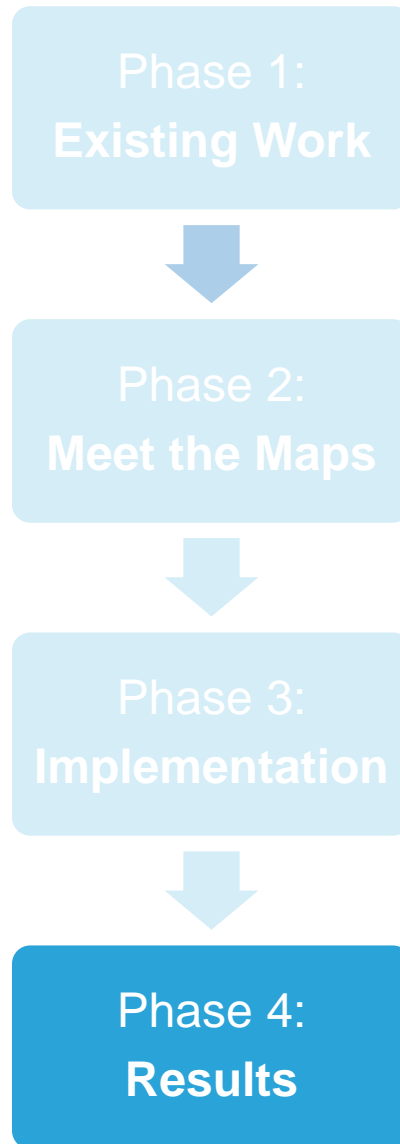


■ Example: after contraction



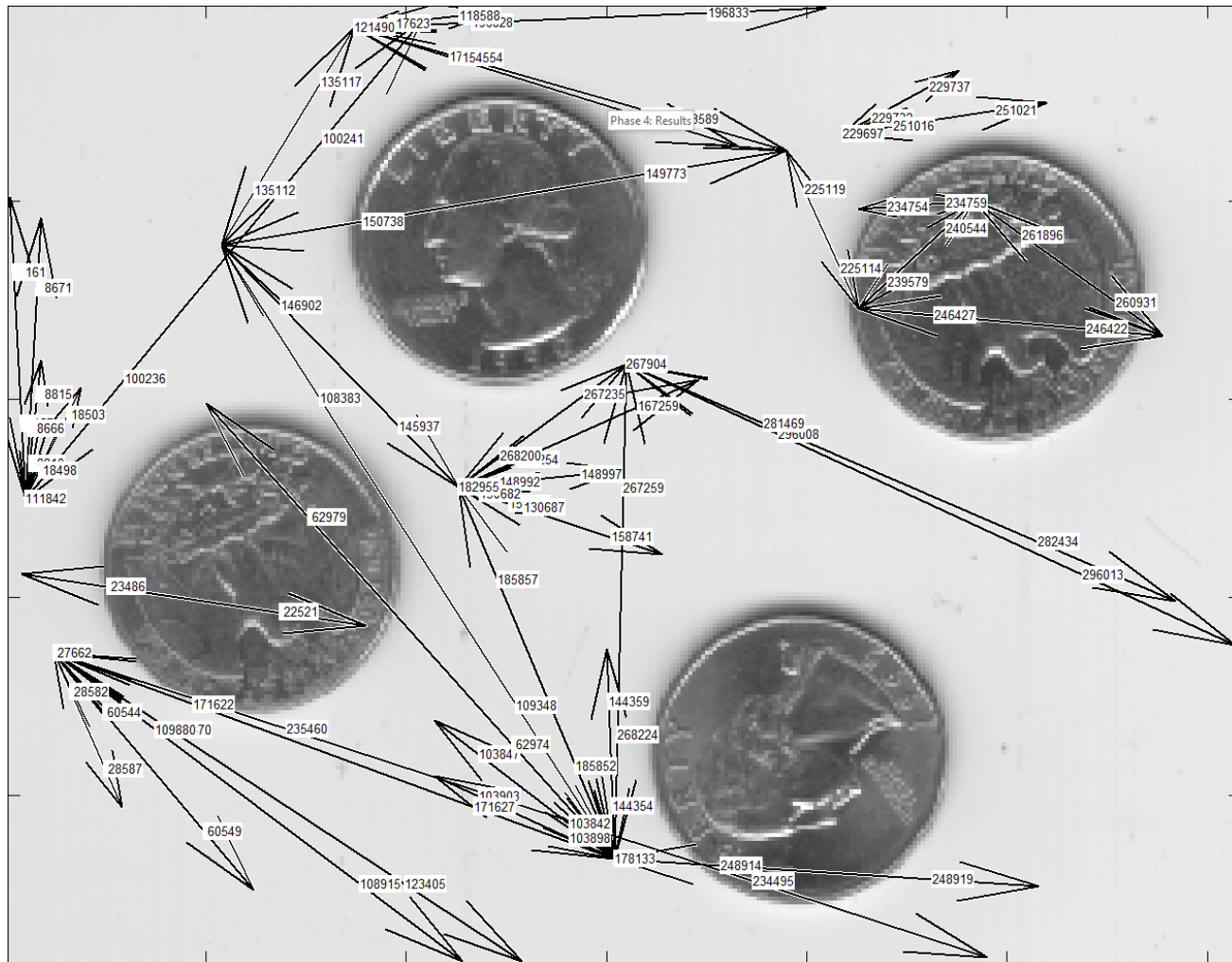
■ Example: after simplification





■ DEMO

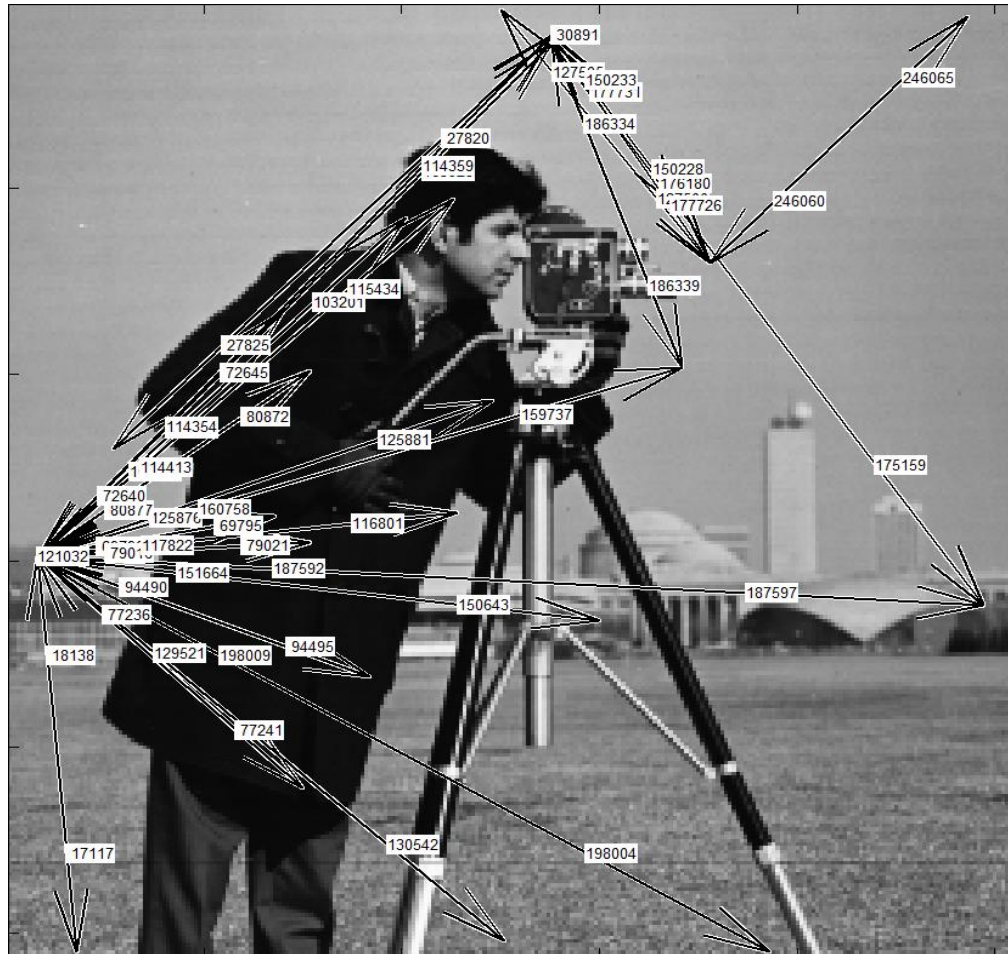
■ Some costly examples:



■ Some costly examples:

1. Computed dart values in $t = 0.0081839$
2. Build the first level in $t = 0.034072$
3. Pyramid level 1:
4. Computing contraction darts in $t = 3.1576$
5. Contracting darts in $t = 1.5676$
6. Simplify darts in $t = 99.6362$
7. Pyramid level 2:
8. Computing contraction darts in $t = 0.52598$
9. Contracting darts in $t = 0.22664$
10. Simplify darts in $t = 7.4213$
11. Pyramid level 3:
12. Computing contraction darts in $t = 0.14502$
13. Contracting darts in $t = 0.052272$
14. Simplify darts in $t = 0.90517$
15. Pyramid level 4:
16. Computing contraction darts in $t = 0.04065$
17. Contracting darts in $t = 0.013697$
18. Simplify darts in $t = 0.13381$
19. Pyramid level 5:
20. Computing contraction darts in $t = 0.020479$
21. Contracting darts in $t = 0.0055467$
22. Simplify darts in $t = 0.029576$
23. Pyramid level 6:
24. Computing contraction darts in $t = 0.0089889$
25. Contracting darts in $t = 0.0038467$
26. Simplify darts in $t = 0.010096$
27. Pyramid level 7:
28. Computing contraction darts in $t = 0.03025$
29. Contracting darts in $t = 0.0028141$
30. Simplify darts in $t = 0.0046334$
31. Pyramid level 8:
32. Computing contraction darts in $t = 0.021293$
33. Contracting darts in $t = 0.00013621$
34. Simplify darts in $t = 0.0022108$

■ Some costly examples:



■ Some costly examples:

1. Computed dart values in $t = 0.0040965$
2. Build the first level in $t = 0.028408$
3. Pyramid level 1:
4. Computing contraction darts in $t = 2.3352$
5. Contracting darts in $t = 1.0566$
6. Simplify darts in $t = 73.4445$
7. Pyramid level 2:
8. Computing contraction darts in $t = 0.66482$
9. Contracting darts in $t = 0.29912$
10. Simplify darts in $t = 12.5519$
11. Pyramid level 3:
12. Computing contraction darts in $t = 0.17884$
13. Contracting darts in $t = 0.077297$
14. Simplify darts in $t = 1.668$
15. Pyramid level 4:
16. Computing contraction darts in $t = 0.054448$
17. Contracting darts in $t = 0.021359$
18. Simplify darts in $t = 0.24825$
19. Pyramid level 5:
20. Computing contraction darts in $t = 0.016426$
21. Contracting darts in $t = 0.0073146$
22. Simplify darts in $t = 0.04384$
23. Pyramid level 6:
24. Computing contraction darts in $t = 0.010979$
25. Contracting darts in $t = 0.0039908$
26. Simplify darts in $t = 0.013091$
27. Pyramid level 7:
28. Computing contraction darts in $t = 0.008718$
29. Contracting darts in $t = 0.003313$
30. Simplify darts in $t = 0.004469$
31. Pyramid level 8:
32. Computing contraction darts in $t = 0.015805$
33. Contracting darts in $t = 0.0001234$
34. Simplify darts in $t = 0.0020848$

- [Tor] Torres, Fuensanta, and Walter G. Kropatsch. "Canonical Encoding of the Combinatorial Pyramid."
- [Bru01] Brun, Luc, and Walter Kropatsch. "Introduction to combinatorial pyramids." *Digital and image geometry*. Springer Berlin Heidelberg, 2001.
- [Bru03] Brun, Luc, and Walter Kropatsch. "Contraction kernels and combinatorial maps." *Pattern Recognition Letters* 24.8 (2003): 1051-1057.