Structures from Combinatorial Geometry and their Encodings

Manfred Scheucher

Our Goal

axiomize structures from combinatorial geometry

Our Goal

axiomize structures from combinatorial geometry

suitable for computer-assisted investiation, in particular using SAT solvers (Boolean satisfiability)

A simple example

given a graph $G = (V, E)$

for $i, j \in \binom{V}{2}$, variables X_{ij} indicates whether $ij \in E$

A simple example

given a graph $G = (V, E)$

for $i, j \in \binom{V}{2}$, variables X_{ij} indicates whether $ij \in E$

Ramsey $R(3, 3)$: neither G nor its complement contains K_3

A simple example

given a graph $G = (V, E)$

for $i, j \in \binom{V}{2}$, variables X_{ij} indicates whether $ij \in E$

Ramsey $R(3, 3)$: neither G nor its complement contains K_3

$\forall a, b, c \in \binom{V}{3} : \neg X_{ab} \vee \neg X_{ac} \vee \neg X_{bc}$ and $X_{ab} \vee X_{ac} \vee X_{bc}$

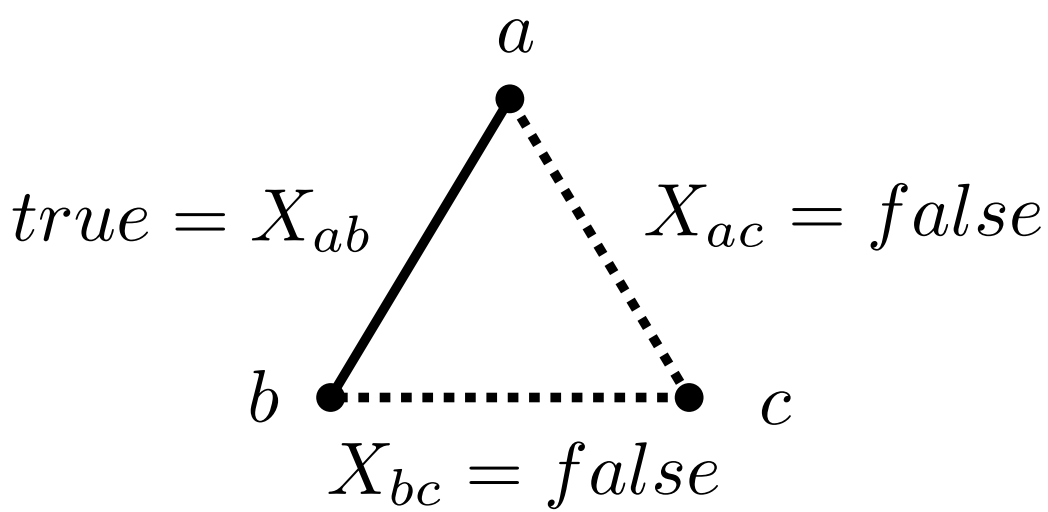
A simple example

given a graph $G = (V, E)$

for $i, j \in \binom{V}{2}$, variables X_{ij} indicates whether $ij \in E$

Ramsey $R(3,3)$: neither G nor its complement contains K_3

$\forall a, b, c \in \binom{V}{3} : \neg X_{ab} \vee \neg X_{ac} \vee \neg X_{bc}$ and $X_{ab} \vee X_{ac} \vee X_{bc}$



A simple example

given a graph $G = (V, E)$

for $i, j \in \binom{V}{2}$, variables X_{ij} indicates whether $ij \in E$

Ramsey $R(3, 3)$: neither G nor its complement contains K_3

$\forall a, b, c \in \binom{V}{3} : \neg X_{ab} \vee \neg X_{ac} \vee \neg X_{bc}$ and $X_{ab} \vee X_{ac} \vee X_{bc}$

in python:

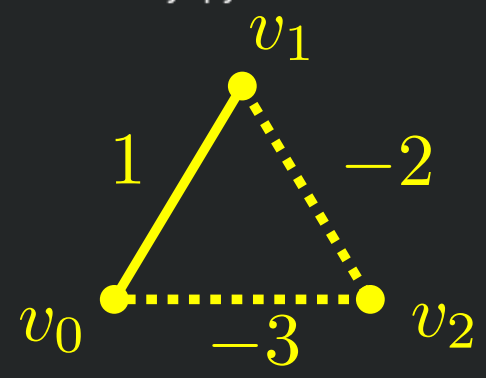
```
for a,b,c in combinations(V,3):
  cnf.append([-edge_var(a,b),-edge_var(a,c),-edge_var(b,c)])
  cnf.append([+edge_var(a,b),+edge_var(a,c),+edge_var(b,c)])
```

ramsey.py | test_chirotope.py | test_signotope.py | test_theorem3.py | signotopes.py | for a,b,c,d in combinations(range(n),4):

```python
from itertools import combinations
from pysat.formula import IDPool,CNF
from pysat.solvers import Solver
from sys import argv

n = int(argv[1])
vertices = range(n)

variables = IDPool()
edge_var = lambda a,b: variables.id((a,b))

print("edge vars:",{(a,b):edge_var(a,b) for (a,b) in combinations(vertices,2)})

cnf = CNF()
for a,b,c in combinations(vertices,3):
    cnf.append([+edge_var(a,b),+edge_var(a,c),+edge_var(b,c)])
    cnf.append([-edge_var(a,b),-edge_var(a,c),-edge_var(b,c)])

print("clauses:",cnf.clauses)

print("solutions:")
s = Solver(bootstrap_with=cnf.clauses)
for ct,sol in enumerate(s.enum_models()):
    edges = [(a,b) for (a,b) in combinations(vertices,2) if edge_var(a,b) in sol]
    print("#",ct,":",sol,"->",edges)

print("end.")
```

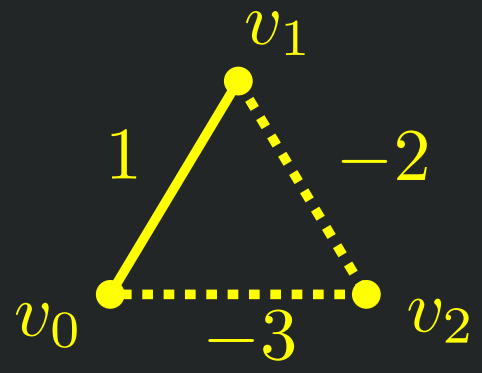
```
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 3
edge vars: {(0, 1): 1, (0, 2): 2, (1, 2): 3}
clauses: [[1, 2, 3], [-1, -2, -3]]
solutions:
# 0 : [1, -2, -3] -> [(0, 1)]
# 1 : [1, 2, -3] -> [(0, 1), (0, 2)]
# 2 : [-1, 2, 3] -> [(0, 2), (1, 2)]
# 3 : [-1, -2, 3] -> [(1, 2)]
# 4 : [-1, 2, -3] -> [(0, 2)]
# 5 : [1, -2, 3] -> [(0, 1), (1, 2)]
end.
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 5
edge vars: {(0, 1): 1, (0, 2): 2, (0, 3): 3, (0, 4): 4, (1, 2): 5, (1, 3): 6, (1, 4): 7, (2, 3): 8, (2, 4): 9, (3,
clauses: [[1, 2, 5], [-1, -2, -5], [1, 3, 6], [-1, -3, -6], [1, 4, 7], [-1, -4, -7], [2, 3, 8], [-2, -3, -8], [2,
, [-5, -6, -8], [5, 7, 9], [-5, -7, -9], [6, 7, 10], [-6, -7, -10], [8, 9, 10], [-8, -9, -10]]
solutions:
# 0 : [-1, -2, 3, 4, 5, -6, 7, 8, -9, -10] -> [(0, 3), (0, 4), (1, 2), (1, 4), (2, 3)]
# 1 : [-1, 2, -3, 4, -5, 6, 7, 8, -9, -10] -> [(0, 2), (0, 4), (1, 3), (1, 4), (2, 3)]
# 2 : [1, -2, 3, -4, -5, -6, 7, 8, 9, -10] -> [(0, 1), (0, 3), (1, 4), (2, 3), (2, 4)]
# 3 : [1, -2, -3, 4, -5, 6, -7, 8, 9, -10] -> [(0, 1), (0, 4), (1, 3), (2, 3), (2, 4)]
# 4 : [1, -2, -3, 4, 5, -6, -7, 8, -9, 10] -> [(0, 1), (0, 4), (1, 2), (2, 3), (3, 4)]
# 5 : [1, 2, -3, -4, -5, -6, 7, 8, -9, 10] -> [(0, 1), (0, 2), (1, 4), (2, 3), (3, 4)]
# 6 : [1, 2, -3, -4, -5, 6, -7, -8, 9, 10] -> [(0, 1), (0, 2), (1, 3), (2, 4), (3, 4)]
# 7 : [-1, 2, -3, 4, 5, 6, -7, -8, -9, 10] -> [(0, 2), (0, 4), (1, 2), (1, 3), (3, 4)]
# 8 : [-1, -2, 3, 4, 5, 6, -7, -8, 9, -10] -> [(0, 3), (0, 4), (1, 2), (1, 3), (2, 4)]
# 9 : [-1, 2, 3, -4, -5, 6, 7, -8, 9, -10] -> [(0, 2), (0, 3), (1, 3), (1, 4), (2, 4)]
# 10 : [-1, 2, 3, -4, 5, -6, 7, -8, -9, 10] -> [(0, 2), (0, 3), (1, 2), (1, 4), (3, 4)]
# 11 : [1, -2, 3, -4, 5, -6, -7, -8, 9, 10] -> [(0, 1), (0, 3), (1, 2), (2, 4), (3, 4)]
end.
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 6
edge vars: {(0, 1): 1, (0, 2): 2, (0, 3): 3, (0, 4): 4, (0, 5): 5, (1, 2): 6, (1, 3): 7, (1, 4): 8, (1, 5): 9, (2,
(4, 5): 15}
clauses: [[1, 2, 6], [-1, -2, -6], [1, 3, 7], [-1, -3, -7], [1, 4, 8], [-1, -4, -8], [1, 5, 9], [-1, -5, -9], [2,
12], [-2, -5, -12], [3, 4, 13], [-3, -4, -13], [3, 5, 14], [-3, -5, -14], [4, 5, 15], [-4, -5, -15], [6, 7, 10], [
 -9, -12], [7, 8, 13], [-7, -8, -13], [7, 9, 14], [-7, -9, -14], [8, 9, 15], [-8, -9, -15], [10, 11, 13], [-10, -1
1, -12, -15], [13, 14, 15], [-13, -14, -15]]
solutions:
end.
manfred@mscheuch:~/SoCG/structures/examples$
```

```
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 3
edge vars: {(0, 1): 1, (0, 2): 2, (1, 2): 3}
clauses: [[1, 2, 3], [-1, -2, -3]]
solutions:
# 0 : [1, -2, -3] -> [(0, 1)]
# 1 : [1, 2, -3] -> [(0, 1), (0, 2)]
# 2 : [-1, 2, 3] -> [(0, 2), (1, 2)]
# 3 : [-1, -2, 3] -> [(1, 2)]
# 4 : [-1, 2, -3] -> [(0, 2)]
# 5 : [1, -2, 3] -> [(0, 1), (1, 2)]
end.
```

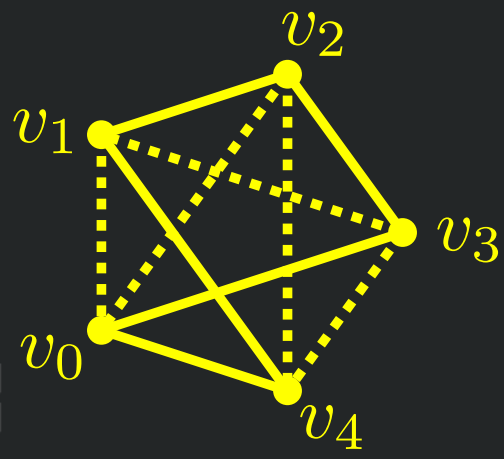



```
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 5
edge vars: {(0, 1): 1, (0, 2): 2, (0, 3): 3, (0, 4): 4, (1, 2): 5, (1, 3): 6, (1, 4): 7, (2, 3): 8, (2, 4): 9, (3,
clauses: [[1, 2, 5], [-1, -2, -5], [1, 3, 6], [-1, -3, -6], [1, 4, 7], [-1, -4, -7], [2, 3, 8], [-2, -3, -8], [2,
, [-5, -6, -8], [5, 7, 9], [-5, -7, -9], [6, 7, 10], [-6, -7, -10], [8, 9, 10], [-8, -9, -10]]
solutions:
# 0 : [-1, -2, 3, 4, 5, -6, 7, 8, -9, -10] -> [(0, 3), (0, 4), (1, 2), (1, 4), (2, 3)]
# 1 : [-1, 2, -3, 4, -5, 6, 7, 8, -9, -10] -> [(0, 2), (0, 4), (1, 3), (1, 4), (2, 3)]
# 2 : [1, -2, 3, -4, -5, -6, 7, 8, 9, -10] -> [(0, 1), (0, 3), (1, 4), (2, 3), (2, 4)]
# 3 : [1, -2, -3, 4, -5, 6, -7, 8, 9, -10] -> [(0, 1), (0, 4), (1, 3), (2, 3), (2, 4)]
# 4 : [1, -2, -3, 4, 5, -6, -7, 8, -9, 10] -> [(0, 1), (0, 4), (1, 2), (2, 3), (3, 4)]
# 5 : [1, 2, -3, -4, -5, -6, 7, 8, -9, 10] -> [(0, 1), (0, 2), (1, 4), (2, 3), (3, 4)]
# 6 : [1, 2, -3, -4, -5, 6, -7, -8, 9, 10] -> [(0, 1), (0, 2), (1, 3), (2, 4), (3, 4)]
# 7 : [-1, 2, -3, 4, 5, 6, -7, -8, -9, 10] -> [(0, 2), (0, 4), (1, 2), (1, 3), (3, 4)]
# 8 : [-1, -2, 3, 4, 5, 6, -7, -8, 9, -10] -> [(0, 3), (0, 4), (1, 2), (1, 3), (2, 4)]
# 9 : [-1, 2, 3, -4, -5, 6, 7, -8, 9, -10] -> [(0, 2), (0, 3), (1, 3), (1, 4), (2, 4)]
# 10 : [-1, 2, 3, -4, 5, -6, 7, -8, -9, 10] -> [(0, 2), (0, 3), (1, 2), (1, 4), (3, 4)]
# 11 : [1, -2, 3, -4, 5, -6, -7, -8, 9, 10] -> [(0, 1), (0, 3), (1, 2), (2, 4), (3, 4)]
end.
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 6
edge vars: {(0, 1): 1, (0, 2): 2, (0, 3): 3, (0, 4): 4, (0, 5): 5, (1, 2): 6, (1, 3): 7, (1, 4): 8, (1, 5): 9, (2,
(4, 5): 15}
clauses: [[1, 2, 6], [-1, -2, -6], [1, 3, 7], [-1, -3, -7], [1, 4, 8], [-1, -4, -8], [1, 5, 9], [-1, -5, -9], [2,
12], [-2, -5, -12], [3, 4, 13], [-3, -4, -13], [3, 5, 14], [-3, -5, -14], [4, 5, 15], [-4, -5, -15], [6, 7, 10],
 -9, -12], [7, 8, 13], [-7, -8, -13], [7, 9, 14], [-7, -9, -14], [8, 9, 15], [-8, -9, -15], [10, 11, 13], [-10, -1
1, -12, -15], [13, 14, 15], [-13, -14, -15]]
solutions:
end.
manfred@mscheuch:~/SoCG/structures/examples$
```

```
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 3
edge vars: {(0, 1): 1, (0, 2): 2, (1, 2): 3}
clauses: [[1, 2, 3], [-1, -2, -3]]
solutions:
# 0 : [1, -2, -3] -> [(0, 1)]
# 1 : [1, 2, -3] -> [(0, 1), (0, 2)]
# 2 : [-1, 2, 3] -> [(0, 2), (1, 2)]
# 3 : [-1, -2, 3] -> [(1, 2)]
# 4 : [-1, 2, -3] -> [(0, 2)]
# 5 : [1, -2, 3] -> [(0, 1), (1, 2)]
end.
```

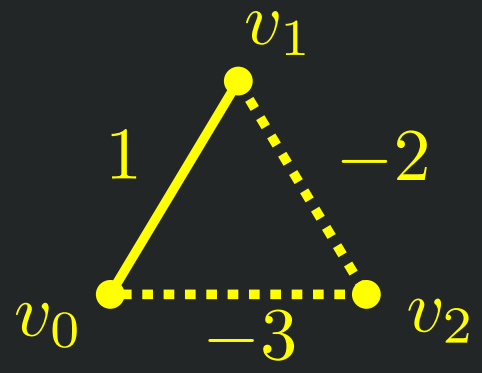



```
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 5
edge vars: {(0, 1): 1, (0, 2): 2, (0, 3): 3, (0, 4): 4, (1, 2): 5, (1, 3): 6, (1, 4): 7, (2, 3): 8, (2, 4): 9, (3,
clauses: [[1, 2, 5], [-1, -2, -5], [1, 3, 6], [-1, -3, -6], [1, 4, 7], [-1, -4, -7], [2, 3, 8], [-2, -3, -8], [2,
, [-5, -6, -8], [5, 7, 9], [-5, -7, -9], [6, 7, 10], [-6, -7, -10], [8, 9, 10], [-8, -9, -10]]
solutions:
# 0 : [-1, -2, 3, 4, 5, -6, 7, 8, -9, -10] -> [(0, 3), (0, 4), (1, 2), (1, 4), (2, 3)]
# 1 : [-1, 2, -3, 4, -5, 6, 7, 8, -9, -10] -> [(0, 2), (0, 4), (1, 3), (1, 4), (2, 3)]
# 2 : [1, -2, 3, -4, -5, -6, 7, 8, 9, -10] -> [(0, 1), (0, 3), (1, 4), (2, 3), (2, 4)]
# 3 : [1, -2, -3, 4, -5, 6, -7, 8, 9, -10] -> [(0, 1), (0, 4), (1, 3), (2, 3), (2, 4)]
# 4 : [1, -2, -3, 4, 5, -6, -7, 8, -9, 10] -> [(0, 1), (0, 4), (1, 2), (2, 3), (3, 4)]
# 5 : [1, 2, -3, -4, -5, -6, 7, 8, -9, 10] -> [(0, 1), (0, 2), (1, 4), (2, 3), (3, 4)]
# 6 : [1, 2, -3, -4, -5, 6, -7, -8, 9, 10] -> [(0, 1), (0, 2), (1, 3), (2, 4), (3, 4)]
# 7 : [-1, 2, -3, 4, 5, 6, -7, -8, -9, 10] -> [(0, 2), (0, 4), (1, 2), (1, 3), (3, 4)]
# 8 : [-1, -2, 3, 4, 5, 6, -7, -8, 9, -10] -> [(0, 3), (0, 4), (1, 2), (1, 3), (2, 4)]
# 9 : [-1, 2, 3, -4, -5, 6, 7, -8, 9, -10] -> [(0, 2), (0, 3), (1, 3), (1, 4), (2, 4)]
# 10 : [-1, 2, 3, -4, 5, -6, 7, -8, -9, 10] -> [(0, 2), (0, 3), (1, 2), (1, 4), (3, 4)]
# 11 : [1, -2, 3, -4, 5, -6, -7, -8, 9, 10] -> [(0, 1), (0, 3), (1, 2), (2, 4), (3, 4)]
end.
```

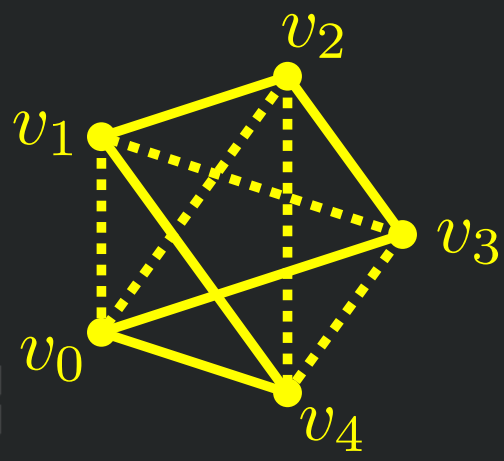



```
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 6
edge vars: {(0, 1): 1, (0, 2): 2, (0, 3): 3, (0, 4): 4, (0, 5): 5, (1, 2): 6, (1, 3): 7, (1, 4): 8, (1, 5): 9, (2,
(4, 5): 15}
clauses: [[1, 2, 6], [-1, -2, -6], [1, 3, 7], [-1, -3, -7], [1, 4, 8], [-1, -4, -8], [1, 5, 9], [-1, -5, -9], [2,
12], [-2, -5, -12], [3, 4, 13], [-3, -4, -13], [3, 5, 14], [-3, -5, -14], [4, 5, 15], [-4, -5, -15], [6, 7, 10],
 -9, -12], [7, 8, 13], [-7, -8, -13], [7, 9, 14], [-7, -9, -14], [8, 9, 15], [-8, -9, -15], [10, 11, 13], [-10, -1
1, -12, -15], [13, 14, 15], [-13, -14, -15]]
solutions:
end.
manfred@mscheuch:~/SoCG/structures/examples$
```

```
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 3
edge vars: {(0, 1): 1, (0, 2): 2, (1, 2): 3}
clauses: [[1, 2, 3], [-1, -2, -3]]
solutions:
# 0 : [1, -2, -3] -> [(0, 1)]
# 1 : [1, 2, -3] -> [(0, 1), (0, 2)]
# 2 : [-1, 2, 3] -> [(0, 2), (1, 2)]
# 3 : [-1, -2, 3] -> [(1, 2)]
# 4 : [-1, 2, -3] -> [(0, 2)]
# 5 : [1, -2, 3] -> [(0, 1), (1, 2)]
end.
```




```
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 5
edge vars: {(0, 1): 1, (0, 2): 2, (0, 3): 3, (0, 4): 4, (1, 2): 5, (1, 3): 6, (1, 4): 7, (2, 3): 8, (2, 4): 9, (3,
clauses: [[1, 2, 5], [-1, -2, -5], [1, 3, 6], [-1, -3, -6], [1, 4, 7], [-1, -4, -7], [2, 3, 8], [-2, -3, -8], [2,
, [-5, -6, -8], [5, 7, 9], [-5, -7, -9], [6, 7, 10], [-6, -7, -10], [8, 9, 10], [-8, -9, -10]]
solutions:
# 0 : [-1, -2, 3, 4, 5, -6, 7, 8, -9, -10] -> [(0, 3), (0, 4), (1, 2), (1, 4), (2, 3)]
# 1 : [-1, 2, -3, 4, -5, 6, 7, 8, -9, -10] -> [(0, 2), (0, 4), (1, 3), (1, 4), (2, 3)]
# 2 : [1, -2, 3, -4, -5, -6, 7, 8, 9, -10] -> [(0, 1), (0, 3), (1, 4), (2, 3), (2, 4)]
# 3 : [1, -2, -3, 4, -5, 6, -7, 8, 9, -10] -> [(0, 1), (0, 4), (1, 3), (2, 3), (2, 4)]
# 4 : [1, -2, -3, 4, 5, -6, -7, 8, -9, 10] -> [(0, 1), (0, 4), (1, 2), (2, 3), (3, 4)]
# 5 : [1, 2, -3, -4, -5, -6, 7, 8, -9, 10] -> [(0, 1), (0, 2), (1, 4), (2, 3), (3, 4)]
# 6 : [1, 2, -3, -4, -5, 6, -7, -8, 9, 10] -> [(0, 1), (0, 2), (1, 3), (2, 4), (3, 4)]
# 7 : [-1, 2, -3, 4, 5, 6, -7, -8, -9, 10] -> [(0, 2), (0, 4), (1, 2), (1, 3), (3, 4)]
# 8 : [-1, -2, 3, 4, 5, 6, -7, -8, 9, -10] -> [(0, 3), (0, 4), (1, 2), (1, 3), (2, 4)]
# 9 : [-1, 2, 3, -4, -5, 6, 7, -8, 9, -10] -> [(0, 2), (0, 3), (1, 3), (1, 4), (2, 4)]
# 10 : [-1, 2, 3, -4, 5, -6, 7, -8, -9, 10] -> [(0, 2), (0, 3), (1, 2), (1, 4), (3, 4)]
# 11 : [1, -2, 3, -4, 5, -6, -7, -8, 9, 10] -> [(0, 1), (0, 3), (1, 2), (2, 4), (3, 4)]
end.
```




```
manfred@mscheuch:~/SoCG/structures/examples$ python ramsey.py 6
edge vars: {(0, 1): 1, (0, 2): 2, (0, 3): 3, (0, 4): 4, (0, 5): 5, (1, 2): 6, (1, 3): 7, (1, 4): 8, (1, 5): 9, (2,
(4, 5): 15}
clauses: [[1, 2, 6], [-1, -2, -6], [1, 3, 7], [-1, -3, -7], [1, 4, 8], [-1, -4, -8], [1, 5, 9], [-1, -5, -9], [2,
12], [-2, -5, -12], [3, 4, 13], [-3, -4, -13], [3, 5, 14], [-3, -5, -14], [4, 5, 15], [-4, -5, -15], [6, 7, 10],
 -9, -12], [7, 8, 13], [-7, -8, -13], [7, 9, 14], [-7, -9, -14], [8, 9, 15], [-8, -9, -15], [10, 11, 13], [-10, -1
1, -12, -15], [13, 14, 15], [-13, -14, -15]]
solutions:
end.
manfred@mscheuch:~/SoCG/structures/examples$
```

no solutions $\Rightarrow R(3,3) = 6$

Point Configurations

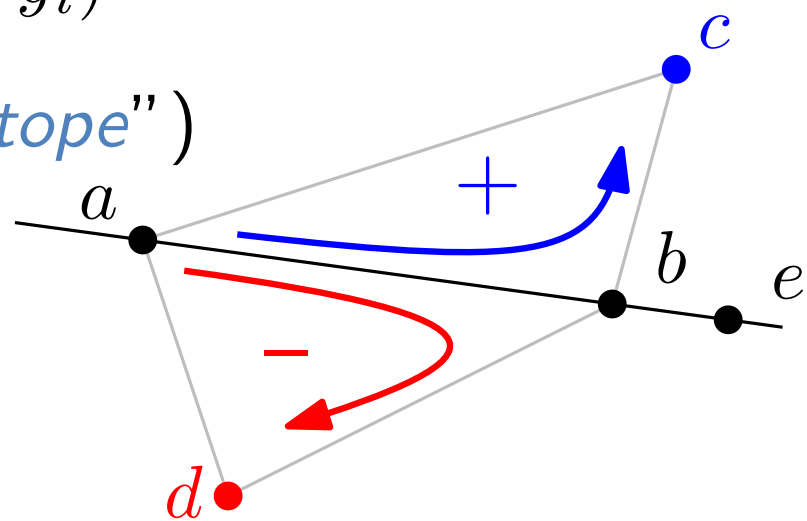
Point Configurations

$P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, $p_i = (x_i, y_i)$

induces triple-orientations ("*chirotope*")

$\chi : [n]^3 \to \{+, 0, -\}$

"positive, collinear, negative"



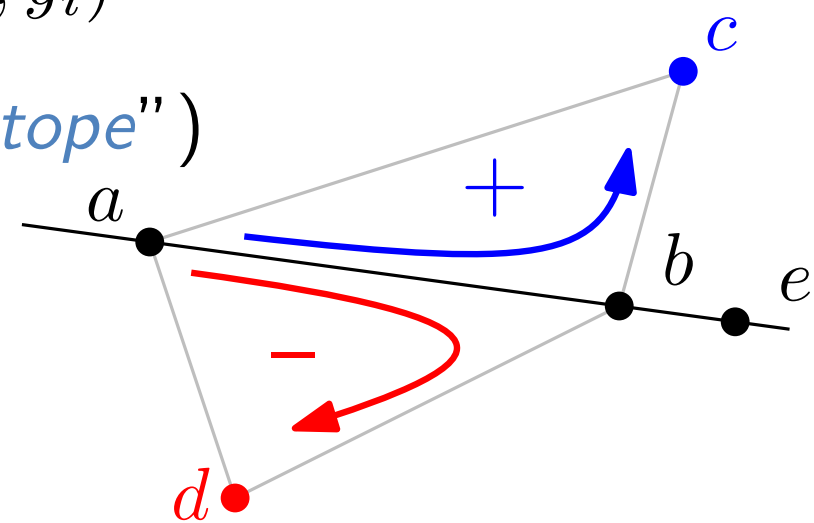
Point Configurations

$P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, $p_i = (x_i, y_i)$

induces triple-orientations ("*chirotope*")

$\chi : [n]^3 \to \{+, 0, -\}$

"positive, collinear, negative"



formally:

$$\chi(a,b,c) := \operatorname{sgn} \det \begin{pmatrix} 1 & 1 & 1 \\ x_a & x_b & x_c \\ y_a & y_b & y_c \end{pmatrix}$$

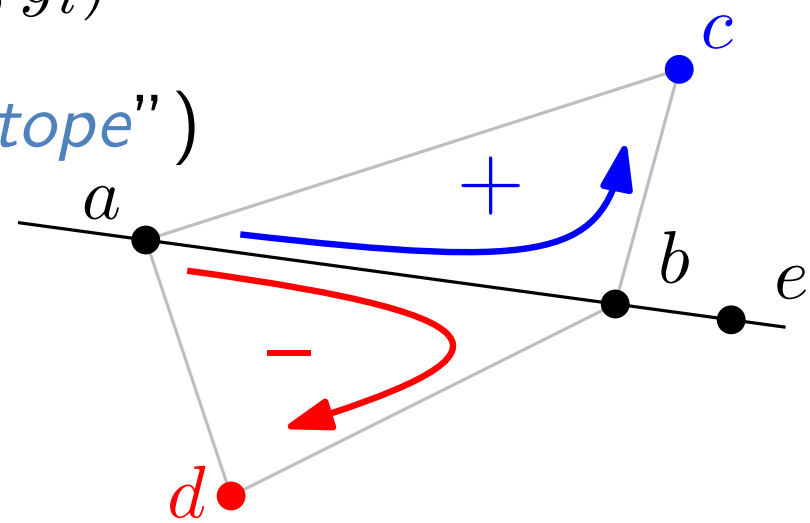
Point Configurations

$P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, $p_i = (x_i, y_i)$

induces triple-orientations ("*chirotope*")

$\chi : [n]^3 \to \{+, 0, -\}$

"positive, collinear, negative"



1.) *alternating axioms*:

$\chi(a, b, c) = \chi(b, c, a), \quad \chi(a, b, c) = -\chi(c, b, a)$

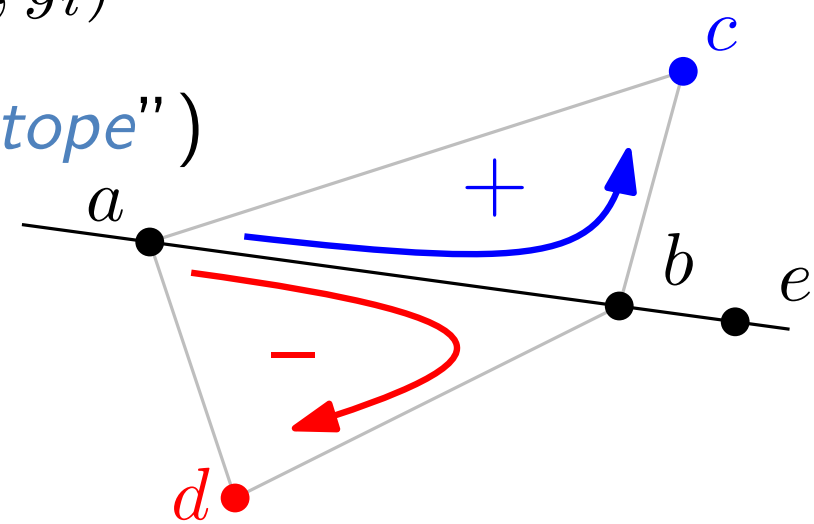
Point Configurations

$P = \{p_1, \ldots, p_n\} \subset \mathbb{R}^2$, $p_i = (x_i, y_i)$

induces triple-orientations ("*chirotope*")

$\chi : [n]^3 \to \{+, 0, -\}$

"positive, collinear, negative"



1.) *alternating axioms*:

$$\chi(a,b,c) = \chi(b,c,a), \quad \chi(a,b,c) = -\chi(c,b,a)$$

2.) *exchange axioms*:

$$\left.\begin{array}{l} \chi(dbc)\cdot\chi(aef) > 0 \\ \chi(ebc)\cdot\chi(daf) > 0 \\ \chi(fbc)\cdot\chi(dea) > 0 \end{array}\right\} \Rightarrow \chi(abc)\cdot\chi(def) > 0$$

Point Configurations

$$\det(x_1,\ldots,x_r)\cdot\det(y_1,\ldots,y_r)=$$

$$\sum_{i=1}^{r}\det(y_i,x_2,\ldots,x_r)\cdot\det(y_1,\ldots,y_{i-1},x_1,y_{i+1},\ldots,y_r)$$

2.) *exchange axioms*:

$$\left.\begin{array}{l}\chi(dbc)\cdot\chi(aef)>0\\ \chi(ebc)\cdot\chi(daf)>0\\ \chi(fbc)\cdot\chi(dea)>0\end{array}\right\}\ \Rightarrow\chi(abc)\cdot\chi(def)>0$$

(via Grassman-Plücker relations)

Point Configurations

$$\det(\overset{a}{x_1}, \overset{b}{\ldots}, \overset{c}{x_r}) \cdot \det(\overset{d}{y_1}, \overset{e}{\ldots}, \overset{f}{y_r}) =$$

$$\sum_{i=1}^{r} \det(y_i, x_2, \ldots, x_r) \cdot \det(y_1, \ldots, y_{i-1}, x_1, y_{i+1}, \ldots, y_r)$$

2.) *exchange axioms*:

$$\left.\begin{array}{l} \chi(dbc) \cdot \chi(aef) > 0 \\ \chi(ebc) \cdot \chi(daf) > 0 \\ \chi(fbc) \cdot \chi(dea) > 0 \end{array}\right\} \Rightarrow \chi(abc) \cdot \chi(def) > 0$$

(via Grassman-Plücker relations)

Point Configurations

(via Laplace expansion)

$$\det(x_1,\ldots,x_r)\cdot\det(y_1,\ldots,y_r) =$$

$$\sum_{i=1}^{r}\det(y_i,x_2,\ldots,x_r)\cdot\det(y_1,\ldots,y_{i-1},x_1,y_{i+1},\ldots,y_r)$$

2.) *exchange axioms*:

$$\left.\begin{array}{l}\chi(dbc)\cdot\chi(aef)>0\\ \chi(ebc)\cdot\chi(daf)>0\\ \chi(fbc)\cdot\chi(dea)>0\end{array}\right\}\ \Rightarrow \chi(abc)\cdot\chi(def)>0$$

(via Grassman-Plücker relations)

Point Configurations

alternating and exchange axioms are necessary conditions
(problem: realizability of chirotopes ETR-hard)

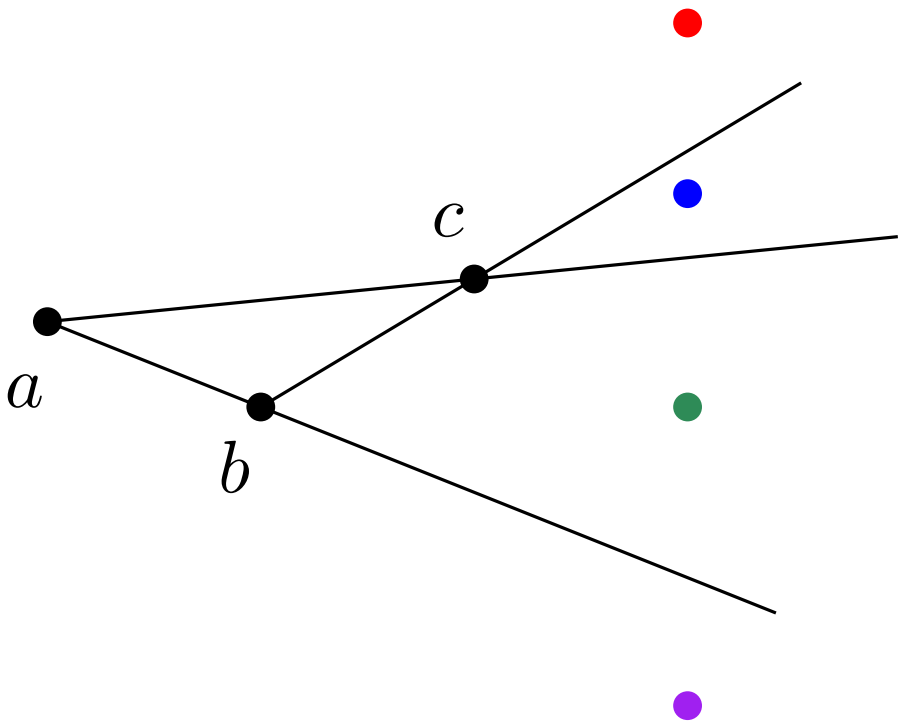
Point Configurations

alternating and exchange axioms are necessary conditions
(problem: realizability of chirotopes ETR-hard)

exchange axioms give $\Theta(n^6)$ constraints,
$\Theta(n^5)$ of them sufficient

Sorted Point Configurations

if point sorted left-to-right

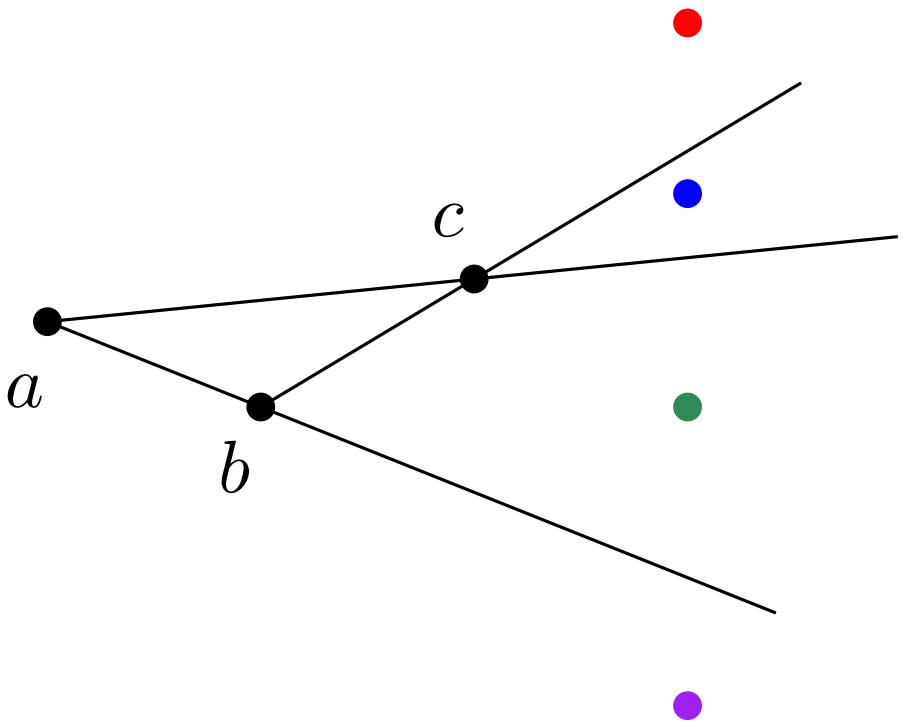


$\forall a, b, c, d$: "$\leq$ 1 sign change"

abc	abd	acd	bcd
+	+	+	+
+	+	+	−
+	+	−	−
+	−	−	−

Sorted Point Configurations

if point sorted left-to-right

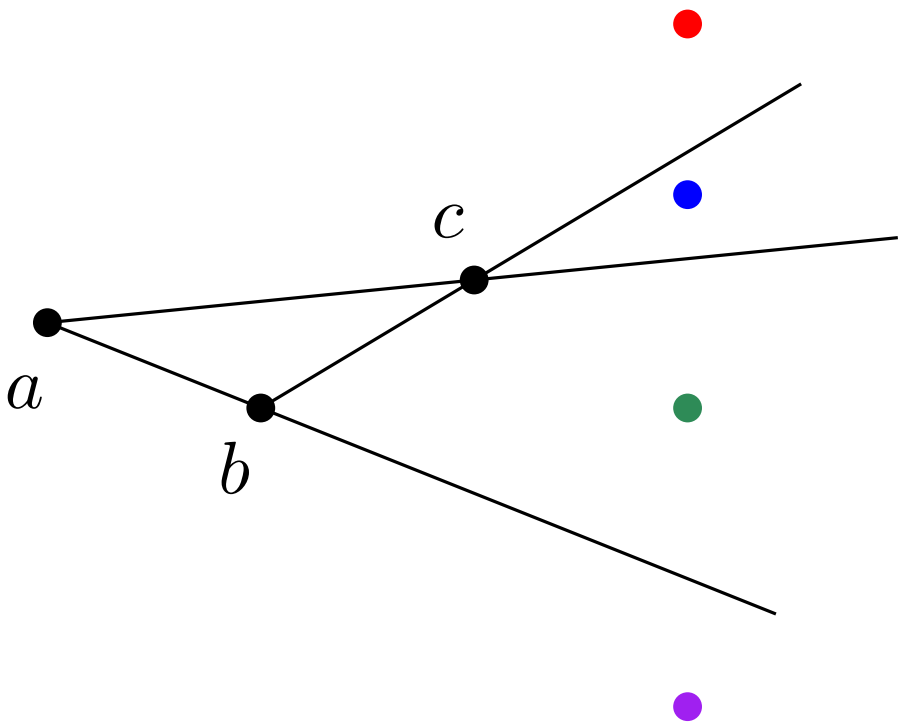


$\forall a, b, c, d$: "$\leq$ 1 sign change"

abc	abd	acd	bcd
+	+	+	+
+	+	+	−
+	+	−	−
+	−	−	−
−	−	−	−
−	−	−	+
−	−	+	+
−	+	+	+

Sorted Point Configurations

if point sorted left-to-right



$\forall a, b, c, d$: "≤ 1 sign change"

abc	abd	acd	bcd
+	+	+	+
+	+	+	−
+	+	−	−
+	−	−	−
−	−	−	−
−	−	−	+
−	−	+	+
−	+	+	+

$\Theta(n^4)$ conditions ("*signotope*")

SAT Encoding for Signotopes

variable S_{abc} for each $a < b < c$,
indicating whether (a, b, c) is $+$ or not

and forbid invalid configurations
(with > 1 sign changes in abc, abd, acd, bcd)

SAT Encoding for Signotopes

variable S_{abc} for each $a < b < c$,
indicating whether (a, b, c) is $+$ or not

and forbid invalid configurations
(with > 1 sign changes in abc, abd, acd, bcd)

$\forall a, b, c, d \in \binom{[n]}{4}$:

$$\begin{array}{cccccccccc} + & & - & & + & & & - & & + & & - \\ \neg S_{abc} & \vee & S_{abd} & \vee & \neg S_{acd} & \text{and} & S_{abc} & \vee & \neg S_{abd} & \vee & S_{acd} \\ \neg S_{abc} & \vee & S_{abd} & \vee & \neg S_{bcd} & \text{and} & S_{abc} & \vee & \neg S_{abd} & \vee & S_{bcd} \\ \neg S_{abc} & \vee & S_{acd} & \vee & \neg S_{bcd} & \text{and} & S_{abc} & \vee & \neg S_{acd} & \vee & S_{bcd} \\ \neg S_{abd} & \vee & S_{acd} & \vee & \neg S_{bcd} & \text{and} & S_{abd} & \vee & \neg S_{acd} & \vee & S_{bcd} \end{array}$$

SAT Encoding for Signotopes

variable S_{abc} for each $a < b < c$,
indicating whether (a, b, c) is $+$ or not

and forbid invalid configurations
(with > 1 sign changes in abc, abd, acd, bcd)

```
for a,b,c,d in combinations(range(n),4):
  cnf.append([-var_s(a,b,c),+var_s(a,b,d),-var_s(a,c,d)])
  cnf.append([+var_s(a,b,c),-var_s(a,b,d),+var_s(a,c,d)])

  cnf.append([-var_s(a,b,c),+var_s(a,b,d),-var_s(b,c,d)])
  cnf.append([+var_s(a,b,c),-var_s(a,b,d),+var_s(b,c,d)])

  cnf.append([-var_s(a,b,c),+var_s(a,c,d),-var_s(b,c,d)])
  cnf.append([+var_s(a,b,c),-var_s(a,c,d),+var_s(b,c,d)])

  cnf.append([-var_s(a,b,d),+var_s(a,c,d),-var_s(b,c,d)])
  cnf.append([+var_s(a,b,d),-var_s(a,c,d),+var_s(b,c,d)])
```

SAT Encoding for Signotopes

variable S_{abc} for each $a < b < c$,
indicating whether (a, b, c) is $+$ or not

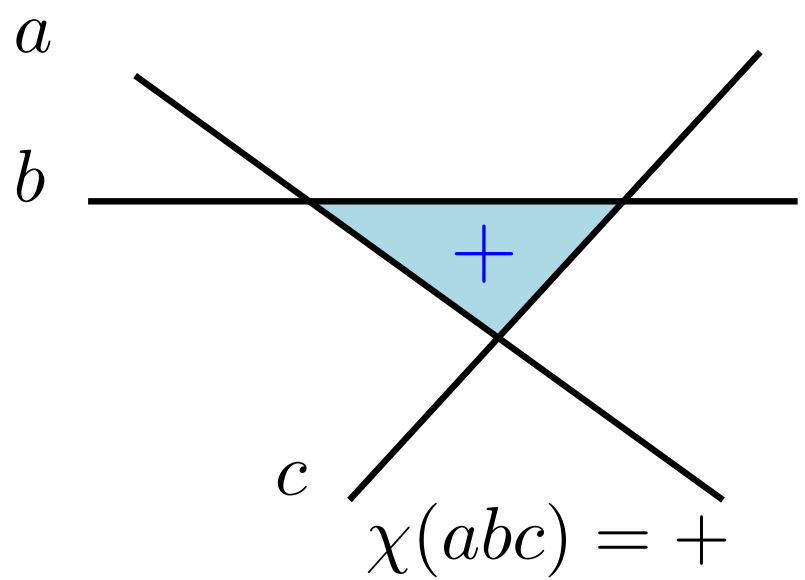
and forbid invalid configurations
(with > 1 sign changes in abc, abd, acd, bcd)

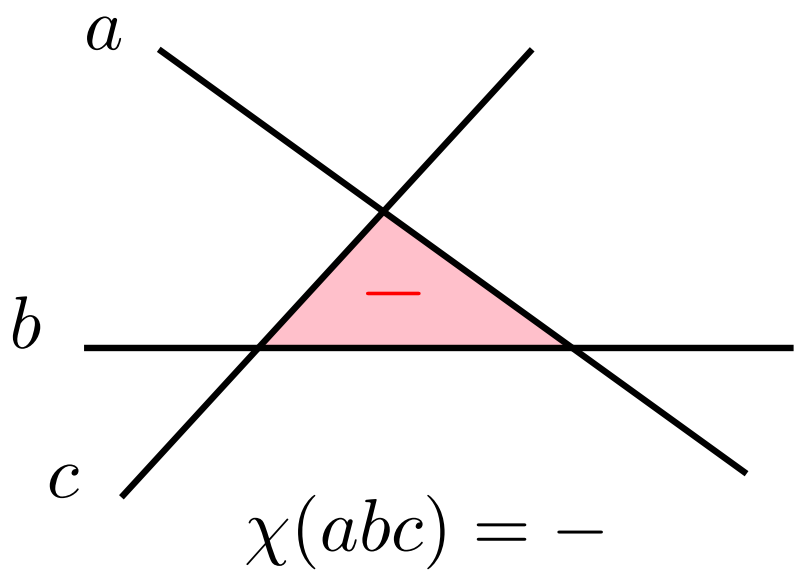
```
manfred@mscheuch:~/SoCG/structures/examples$ python signotopes.py 4
solutions:
# 0 : [(0, 1, 2, '-'), (0, 1, 3, '-'), (0, 2, 3, '-'), (1, 2, 3, '-')]
# 1 : [(0, 1, 2, '-'), (0, 1, 3, '-'), (0, 2, 3, '-'), (1, 2, 3, '+')]
# 2 : [(0, 1, 2, '-'), (0, 1, 3, '-'), (0, 2, 3, '+'), (1, 2, 3, '+')]
# 3 : [(0, 1, 2, '-'), (0, 1, 3, '+'), (0, 2, 3, '+'), (1, 2, 3, '+')]
# 4 : [(0, 1, 2, '+'), (0, 1, 3, '+'), (0, 2, 3, '+'), (1, 2, 3, '+')]
# 5 : [(0, 1, 2, '+'), (0, 1, 3, '+'), (0, 2, 3, '+'), (1, 2, 3, '-')]
# 6 : [(0, 1, 2, '+'), (0, 1, 3, '+'), (0, 2, 3, '-'), (1, 2, 3, '-')]
# 7 : [(0, 1, 2, '+'), (0, 1, 3, '-'), (0, 2, 3, '-'), (1, 2, 3, '-')]
end.
```

Line Arrangement

Line Arrangement

L set of lines $\ell_1, \ldots, \ell_n$ with $\ell : y = \alpha x + \beta$

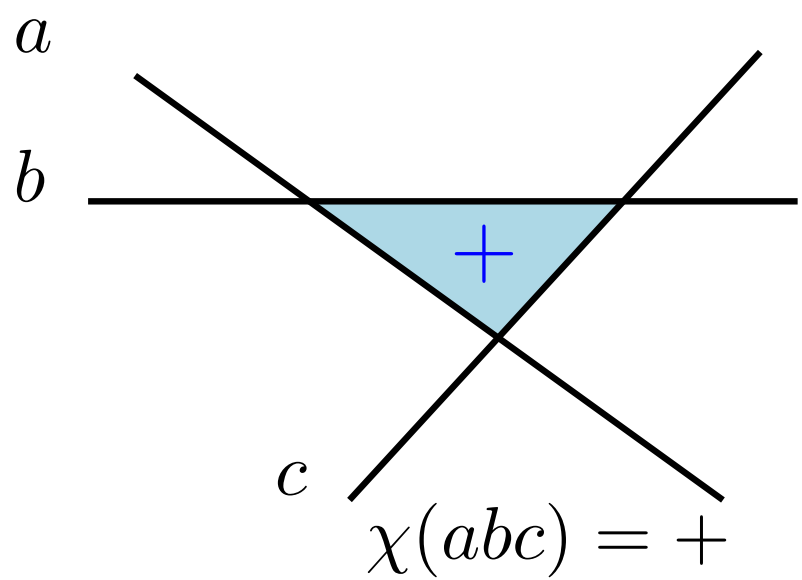


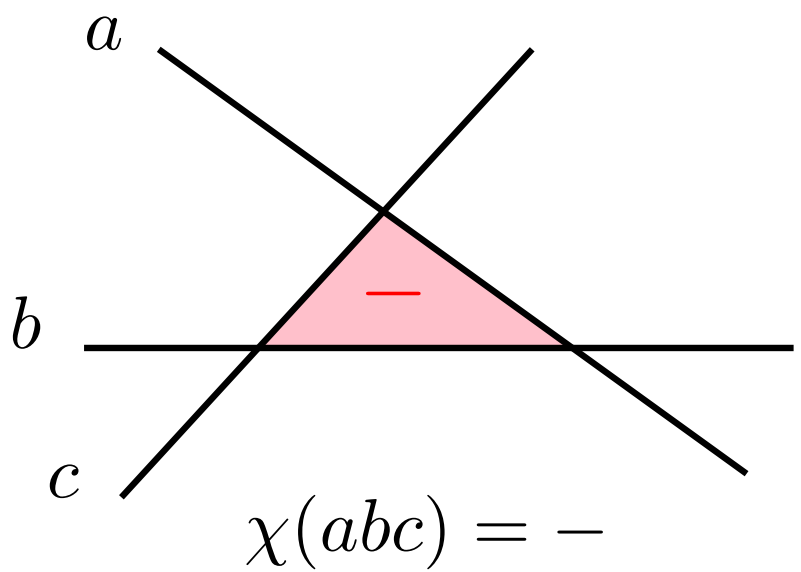


1. L induces chirotope χ

Line Arrangement

L set of lines $\ell_1, \ldots, \ell_n$ with $\ell : y = \alpha x + \beta$

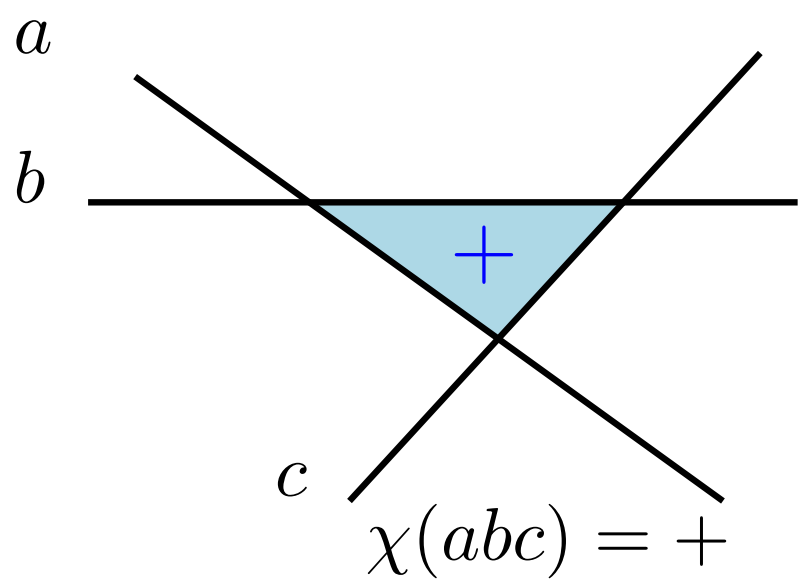


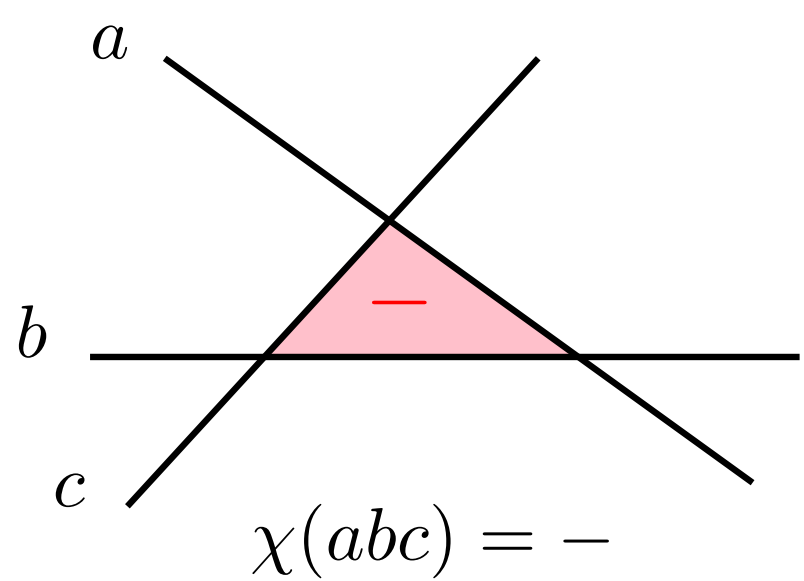


1. L induces chirotope χ
2. if sorted top-to-bottom $\Rightarrow$ signotope

Line Arrangement

L set of lines $\ell_1, \ldots, \ell_n$ with $\ell : y = \alpha x + \beta$



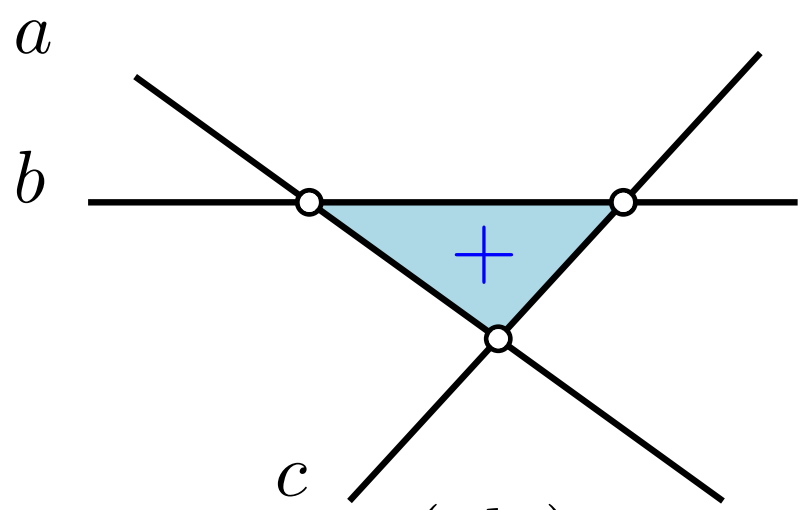


1. L induces chirotope χ
2. if sorted top-to-bottom $\Rightarrow$ signoto

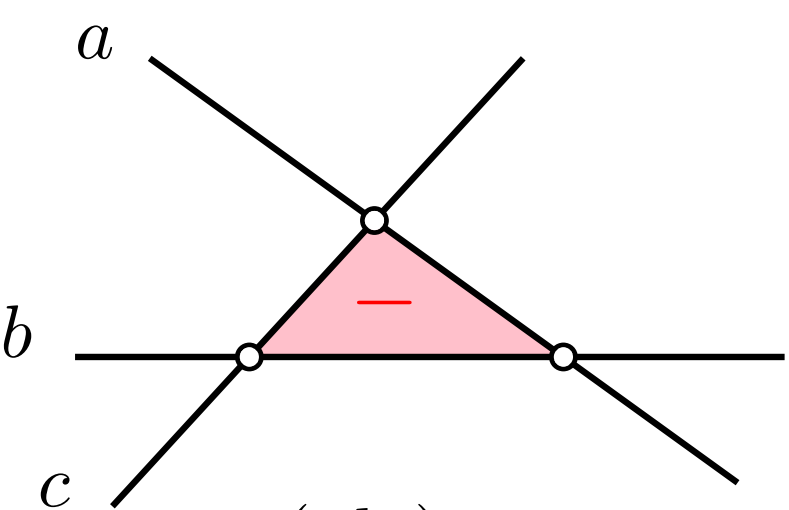
no big surprise because of point-line duality:
$(\alpha, \beta) \leftrightarrow y = \alpha x + \beta$

Line Arrangement

L set of lines $\ell_1, \ldots, \ell_n$ with $\ell : y = \alpha x + \beta$



$\chi(abc) = +$



$\chi(abc) = -$

1. L induces chirotope χ
2. if sorted top-to-bottom $\Rightarrow$ signotope

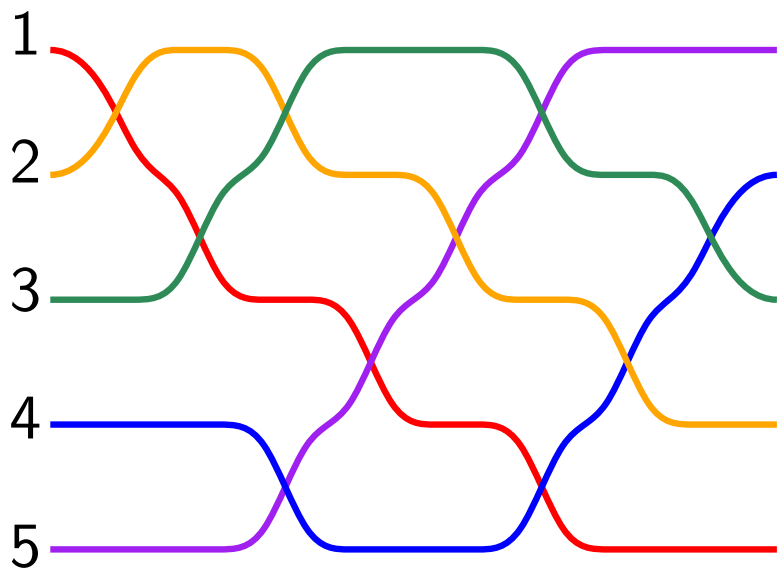
2b: moreover, intersection order $\leftrightarrow$ χ

$\chi(abc) = +$ $\Leftrightarrow$ a intersects b before c
$\Leftrightarrow$ b intersects a before c
$\Leftrightarrow$ c intersects a before b

Pseudoline Arrangements

Folkman–Lawrence 1978, Goodman 1980:
every chirotope has a representation as *pseudoline arrangement*

(x-monotone curves, each two cross in one point)

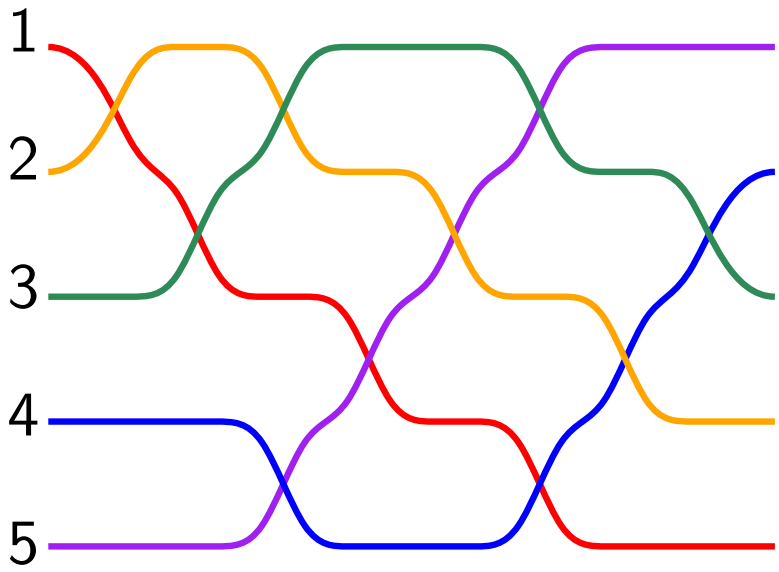


Pseudoline Arrangements

Folkman–Lawrence 1978, Goodman 1980:
every chirotope has a representation as *pseudoline arrangement*

(x-monotone curves, each two cross in one point)

Felsner-Weil 2001: signotopes $\leftrightarrow$ PLA, sorted top-to-bottom

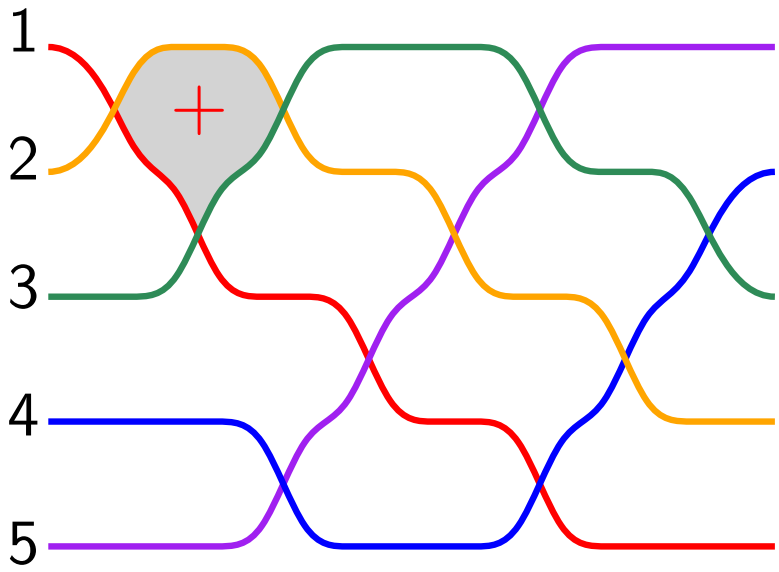


Pseudoline Arrangements

Folkman–Lawrence 1978, Goodman 1980:
every chirotope has a representation as *pseudoline arrangement*

(x-monotone curves, each two cross in one point)

Felsner-Weil 2001: signotopes $\leftrightarrow$ PLA, sorted top-to-bottom



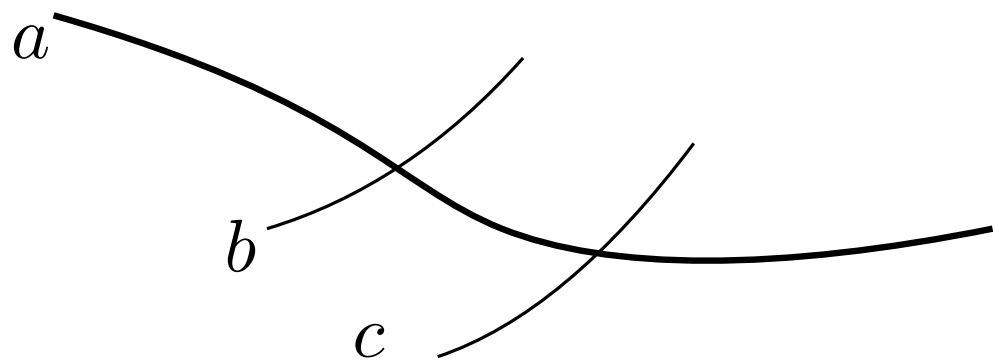
Pseudoline Arrangements

Folkman–Lawrence 1978, Goodman 1980:
every chirotope has a representation as *pseudoline arrangement*

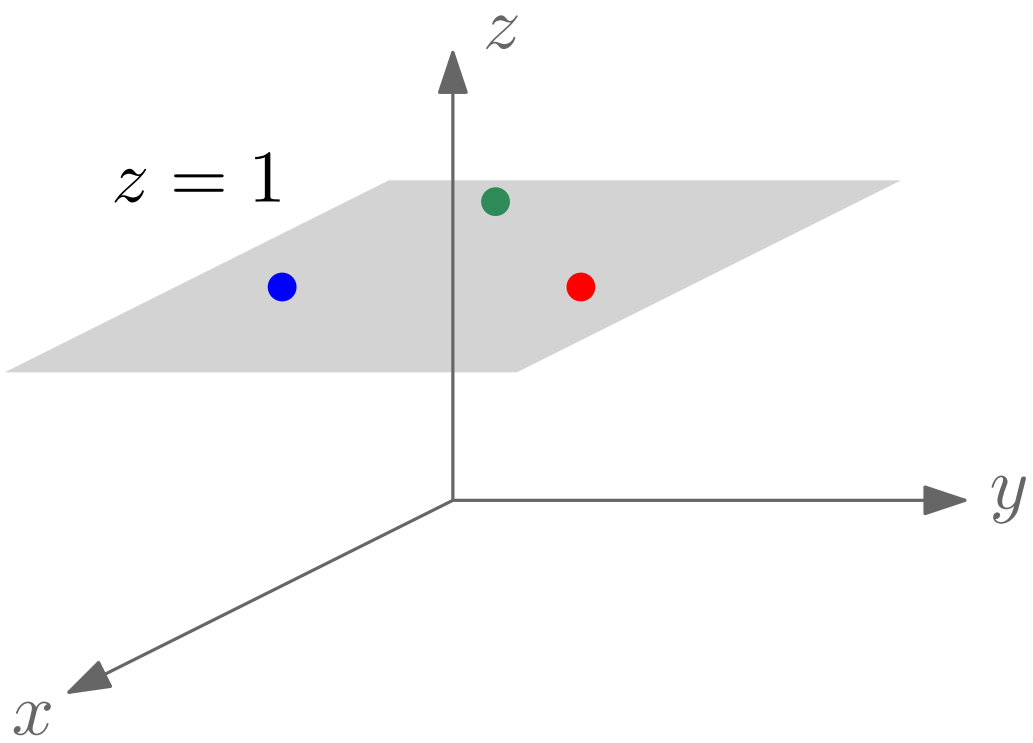
(x-monotone curves, each two cross in one point)

Felsner-Weil 2001: signotopes $\leftrightarrow$ PLA, sorted top-to-bottom

proof idea: χ determines order of intersections
($\chi(abc) = +$ iff a intersects b before c)

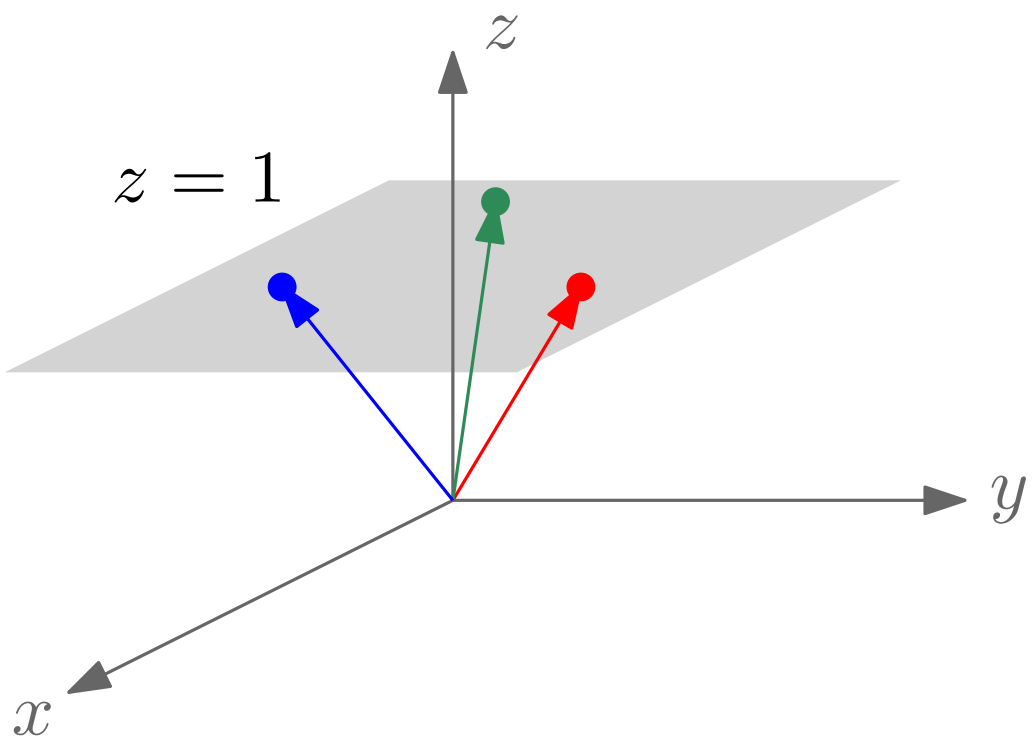


Point-Line Duality Revised



points in plane $(z = 1)$

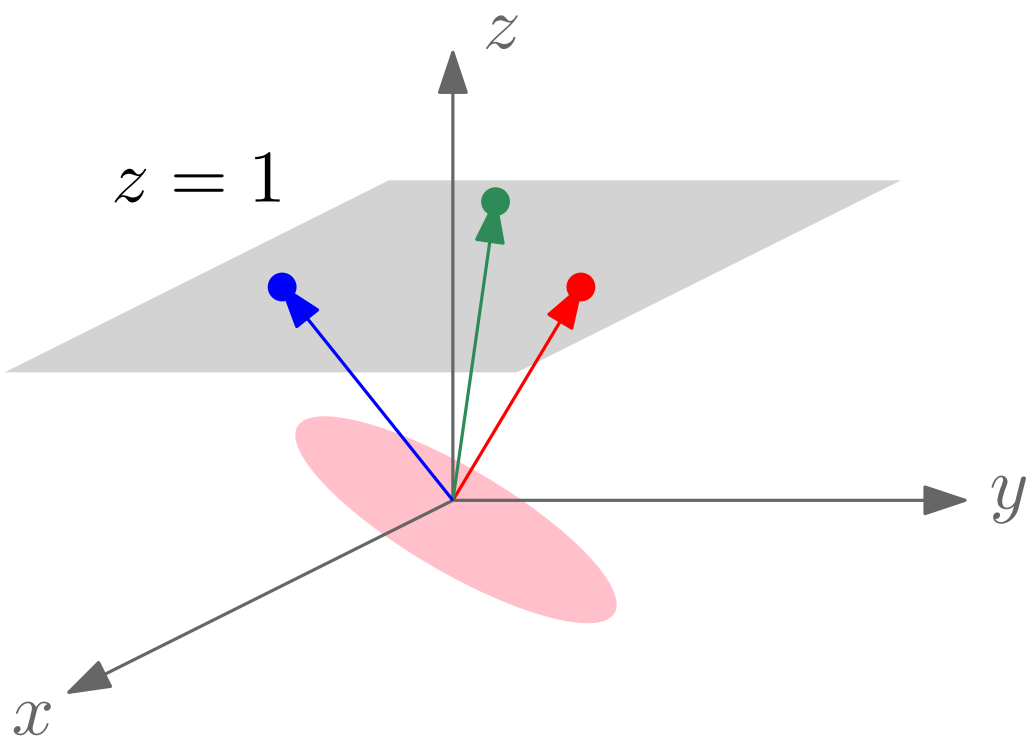
Point-Line Duality Revised



points in plane $(z = 1)$

lines through origin

Point-Line Duality Revised

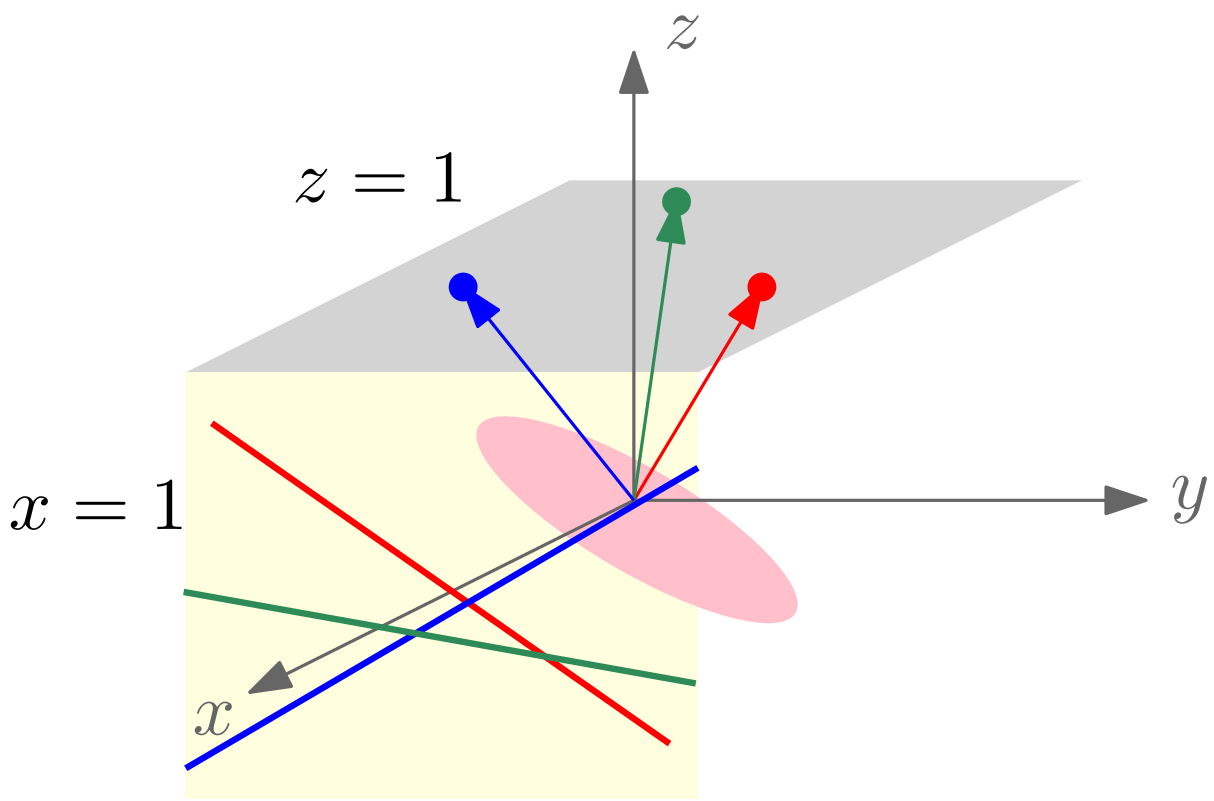


points in plane $(z = 1)$

lines through origin

orthogonal planes

Point-Line Duality Revised



points in plane $(z = 1)$

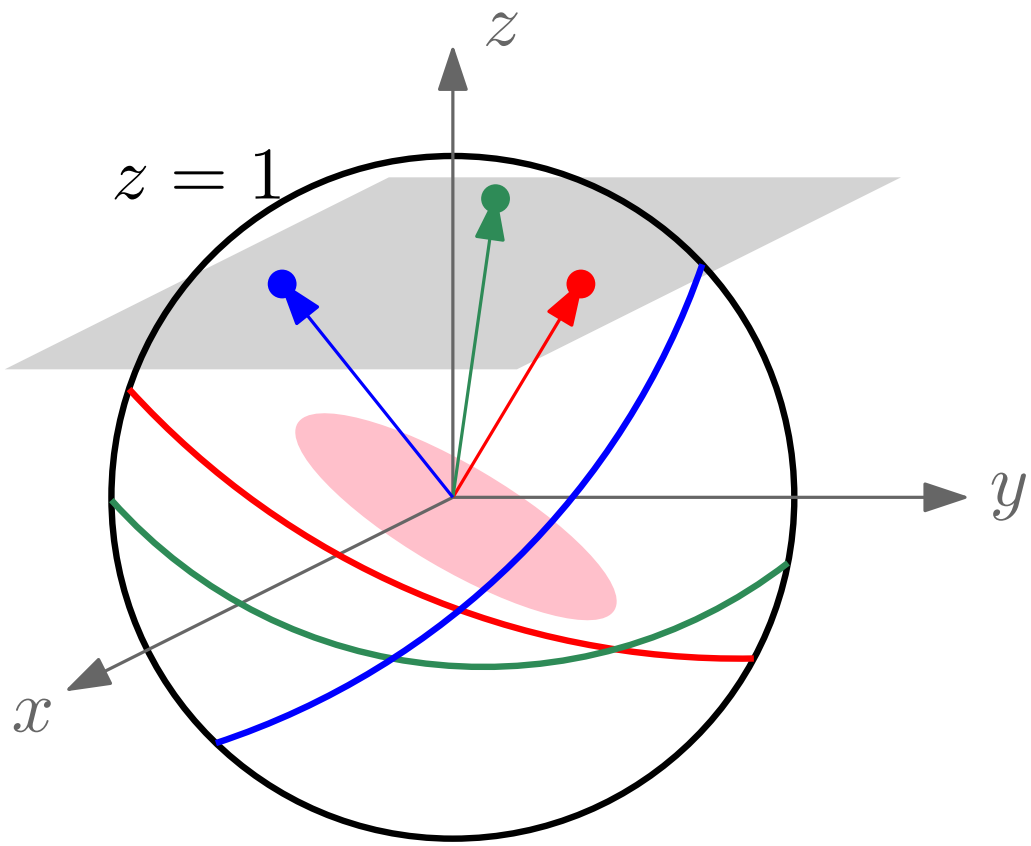
lines through origin

orthogonal planes

duality I:
intersect with plane $x = 1$

line arrangement in $x = 1$

Point-Line Duality Revised



points in plane ($z = 1$)

lines through origin

orthogonal planes

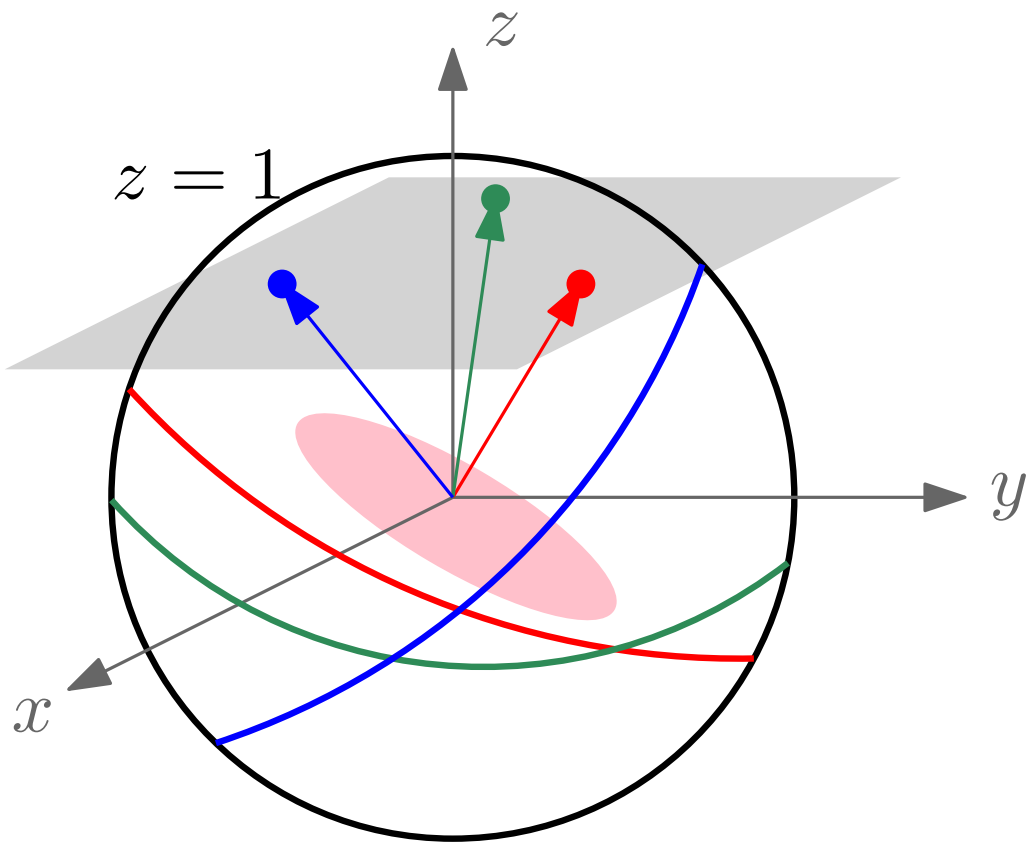
duality I:
intersect with plane $x = 1$

line arrangement in $x = 1$

duality II:
intersect with unit sphere

great-circle arrangement

Point-Line Duality Revised



(front hemisphere like plane)

points in plane $(z = 1)$

lines through origin

orthogonal planes

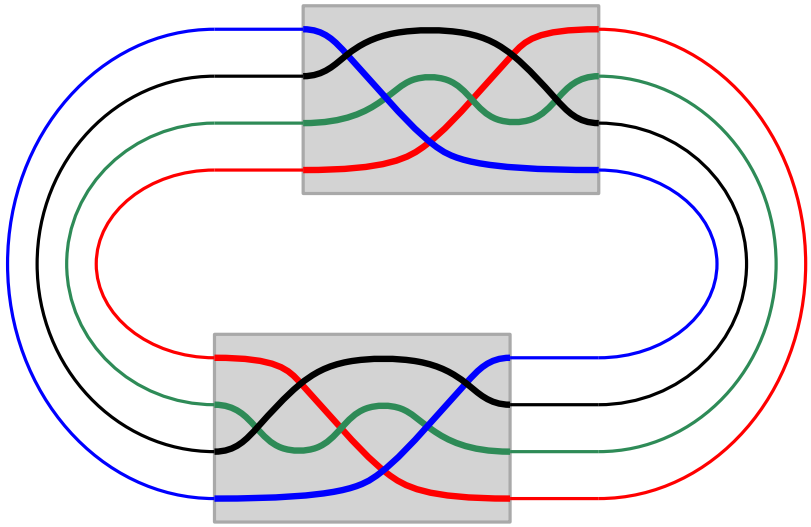
duality I:
intersect with plane $x = 1$

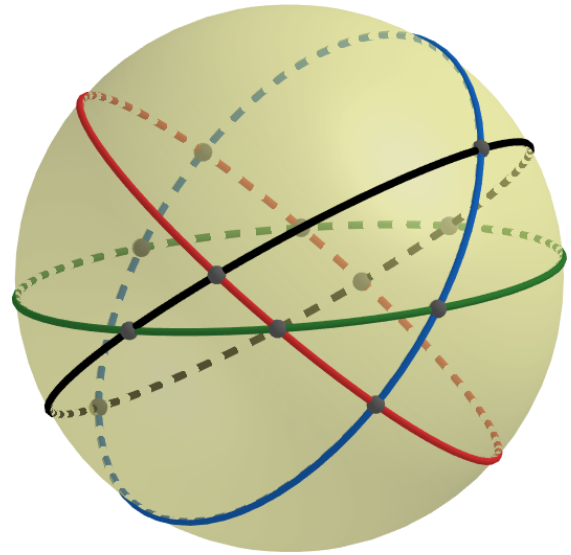
line arrangement in $x = 1$

duality II:
intersect with unit sphere

great-circle arrangement

Great-(Pseudo)Circle Arrangements



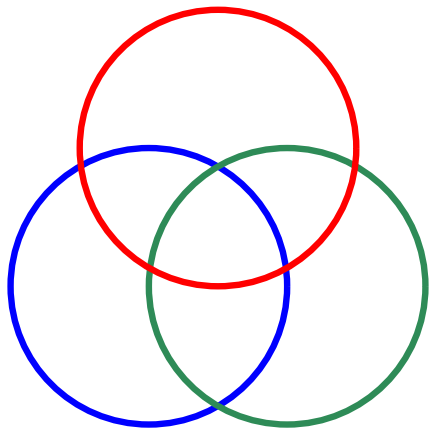


1-to-1 correspondence to (pseudo)line arrangements

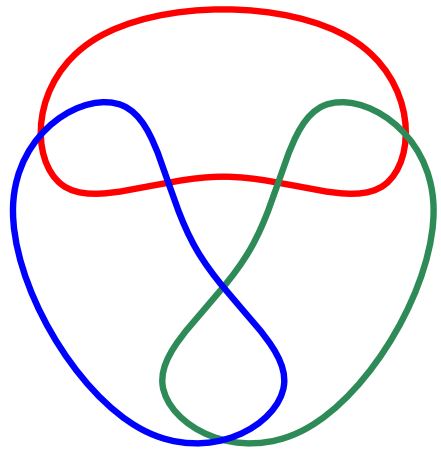
Pseudocircle Arrangements

pseudocircle . . . simple closed curve

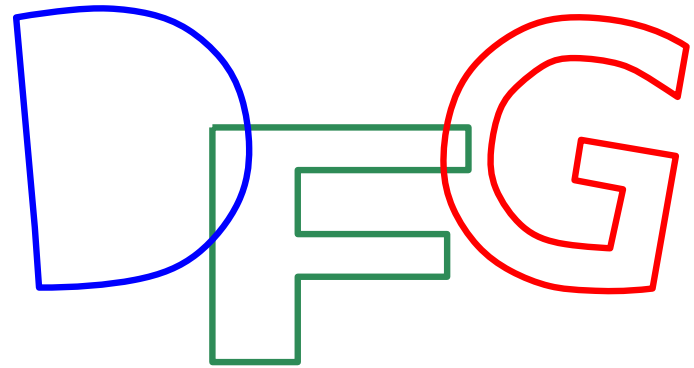
arrangement . . . collection of pcs. s.t. intersection of any two pcs. either empty or 2 points where curves cross



Krupp

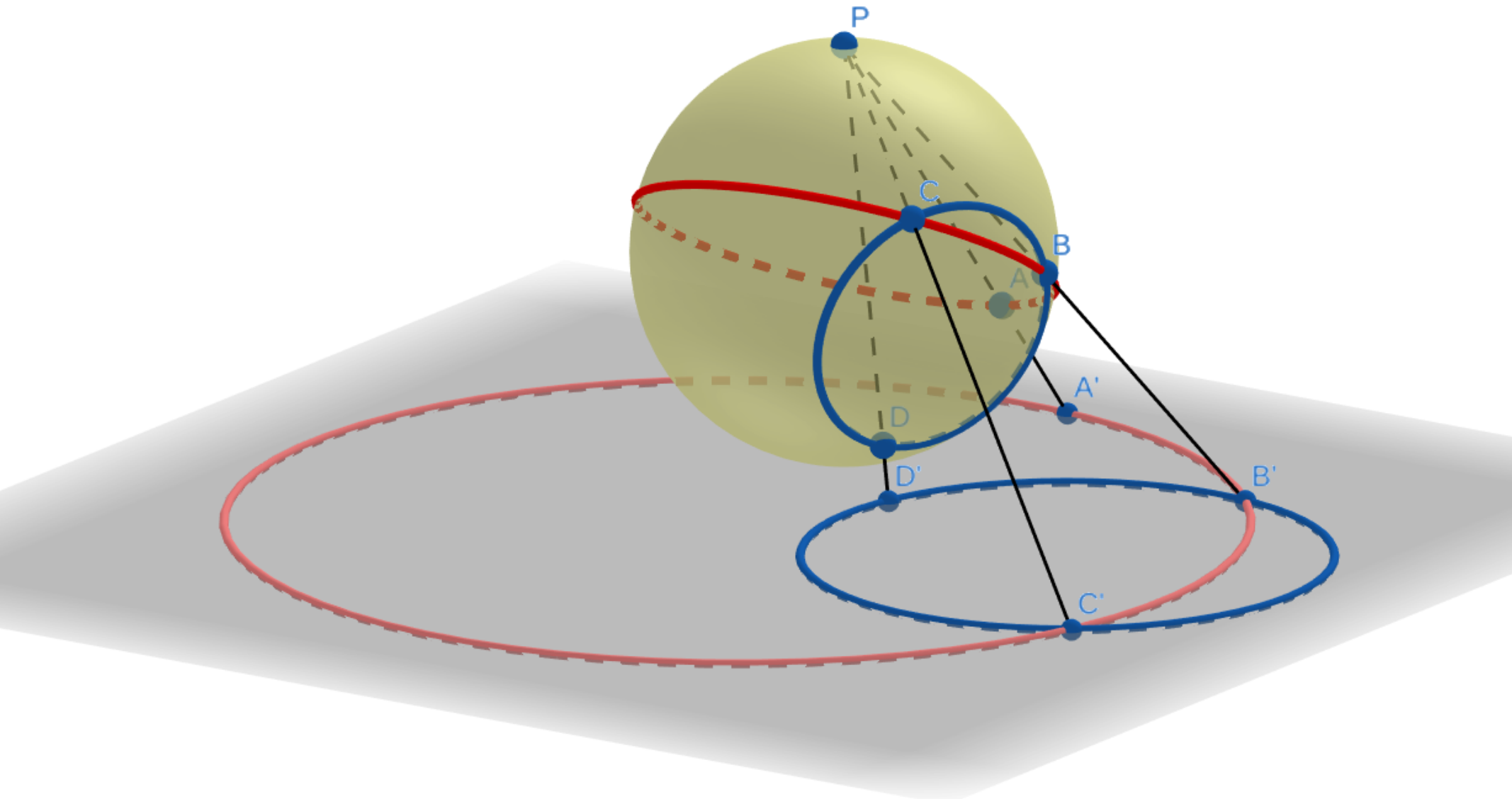


NonKrupp



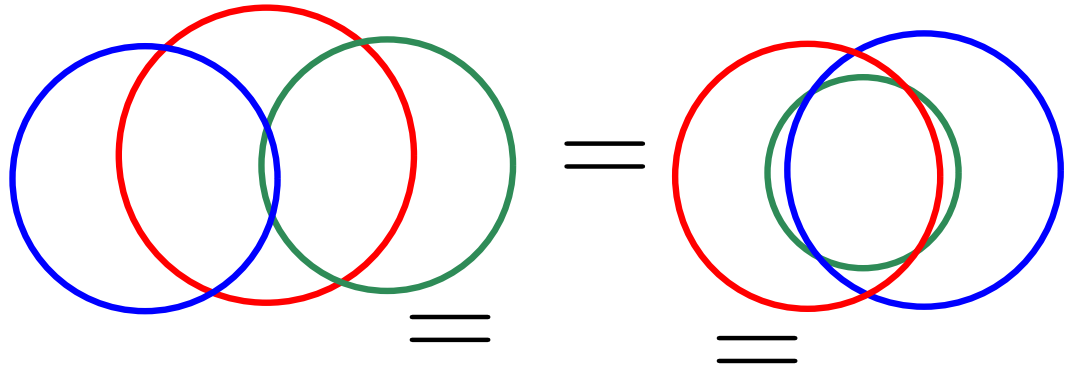
3-Chain

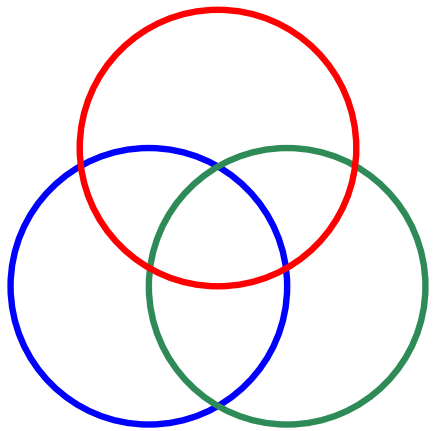
Plane VS Sphere



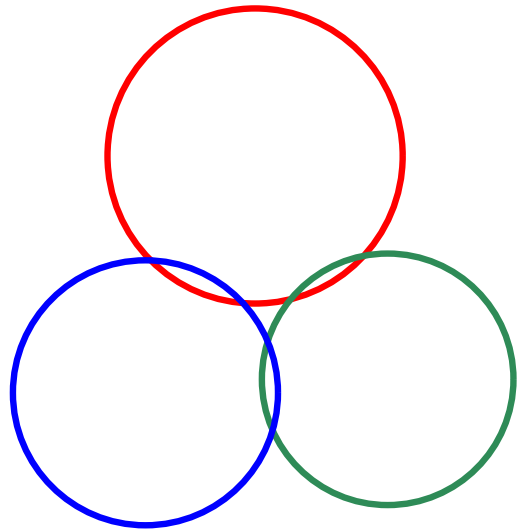
Hierarchy of Pseudocircle Arrangements

connected . . . graph of arrangement is connected

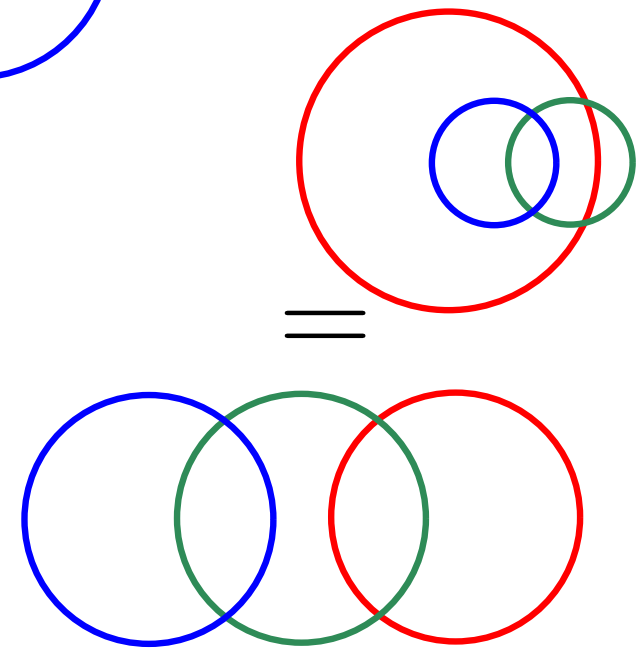




Krupp



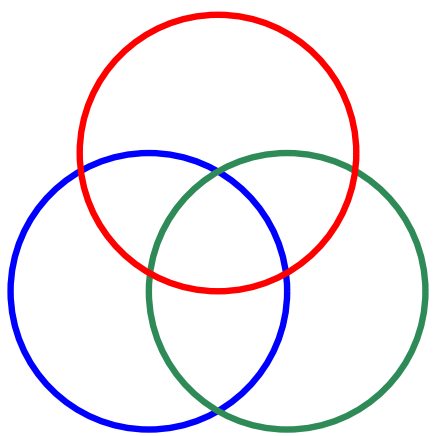
NonKrupp



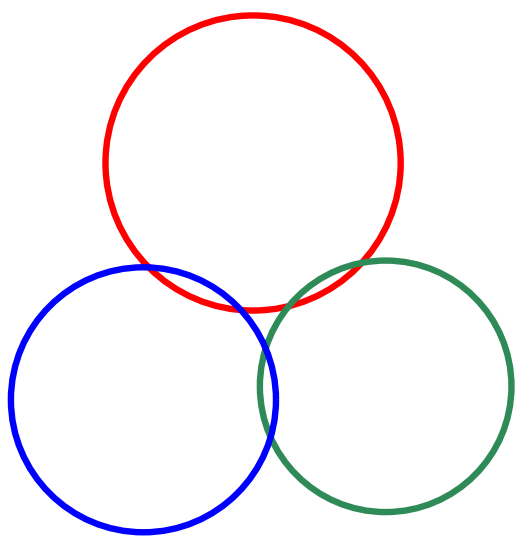
3-Chain

Hierarchy of Pseudocircle Arrangements

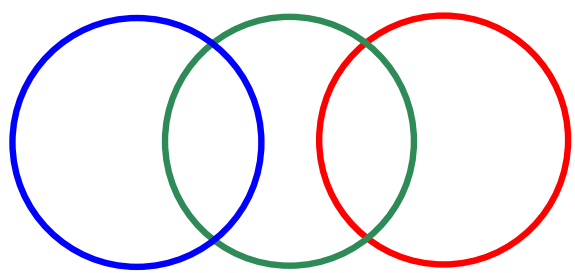
connected . . . graph of arrangement is connected



Krupp



NonKrupp

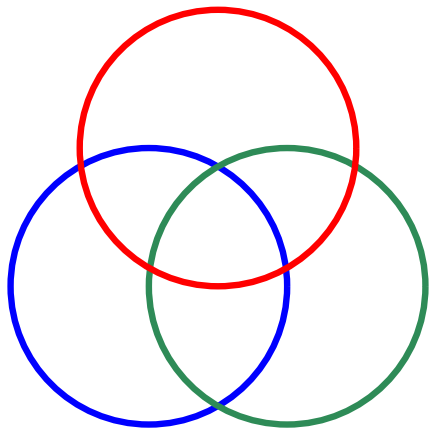


3-Chain

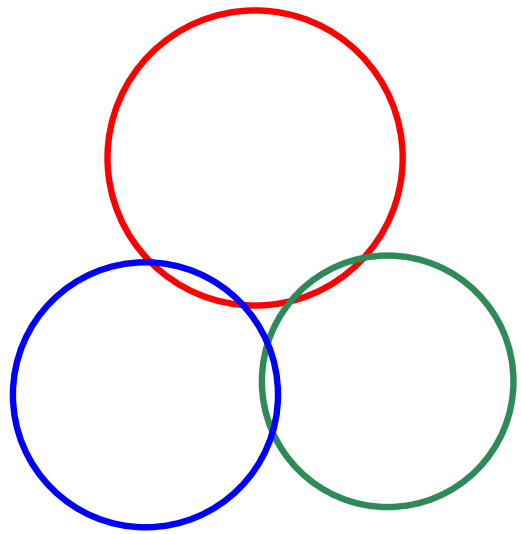
Hierarchy of Pseudocircle Arrangements

connected . . . graph of arrangement is connected

intersecting . . . any 2 pseudocircles cross twice



Krupp



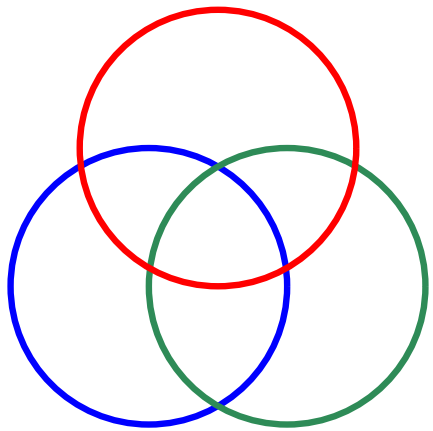
NonKrupp

Hierarchy of Pseudocircle Arrangements

connected . . . graph of arrangement is connected

intersecting . . . any 2 pseudocircles cross twice

arr. of great-pseudocircles . . . any 3 pcs. form a Krupp



Krupp

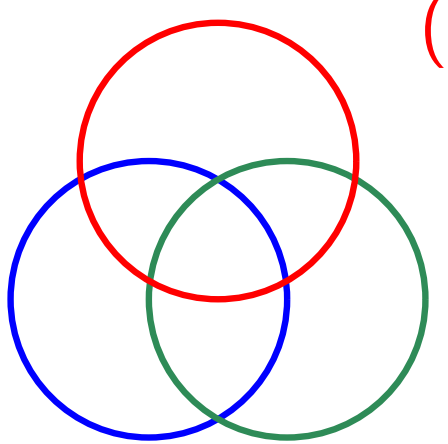
Hierarchy of Pseudocircle Arrangements

connected . . . graph of arrangement is connected

intersecting . . . any 2 pseudocircles cross twice

arr. of great-pseudocircles . . . any 3 pcs. form a Krupp

(characterization via chirotopes/signotopes)



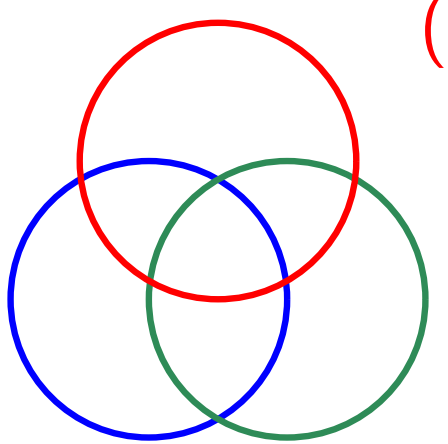
Krupp

Hierarchy of Pseudocircle Arrangements

connected . . . graph of arrangement is connected

intersecting . . . any 2 pseudocircles cross twice
(characterization via 4-tuples, Ortner 2008)

arr. of great-pseudocircles . . . any 3 pcs. form a Krupp
(characterization via chirotopes/signotopes)



Krupp

Hierarchy of Pseudocircle Arrangements

connected . . . graph of arrangement is connected

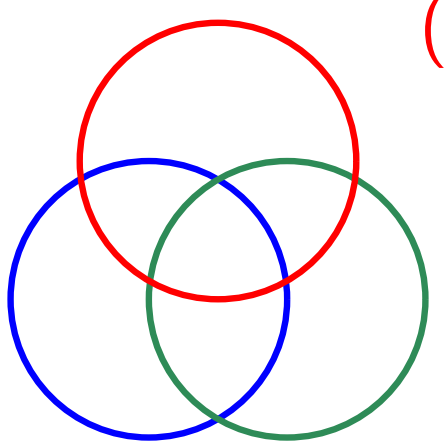
(characterization?)

intersecting . . . any 2 pseudocircles cross twice

(characterization via 4-tuples, Ortner 2008)

arr. of great-pseudocircles . . . any 3 pcs. form a Krupp

(characterization via chirotopes/signotopes)

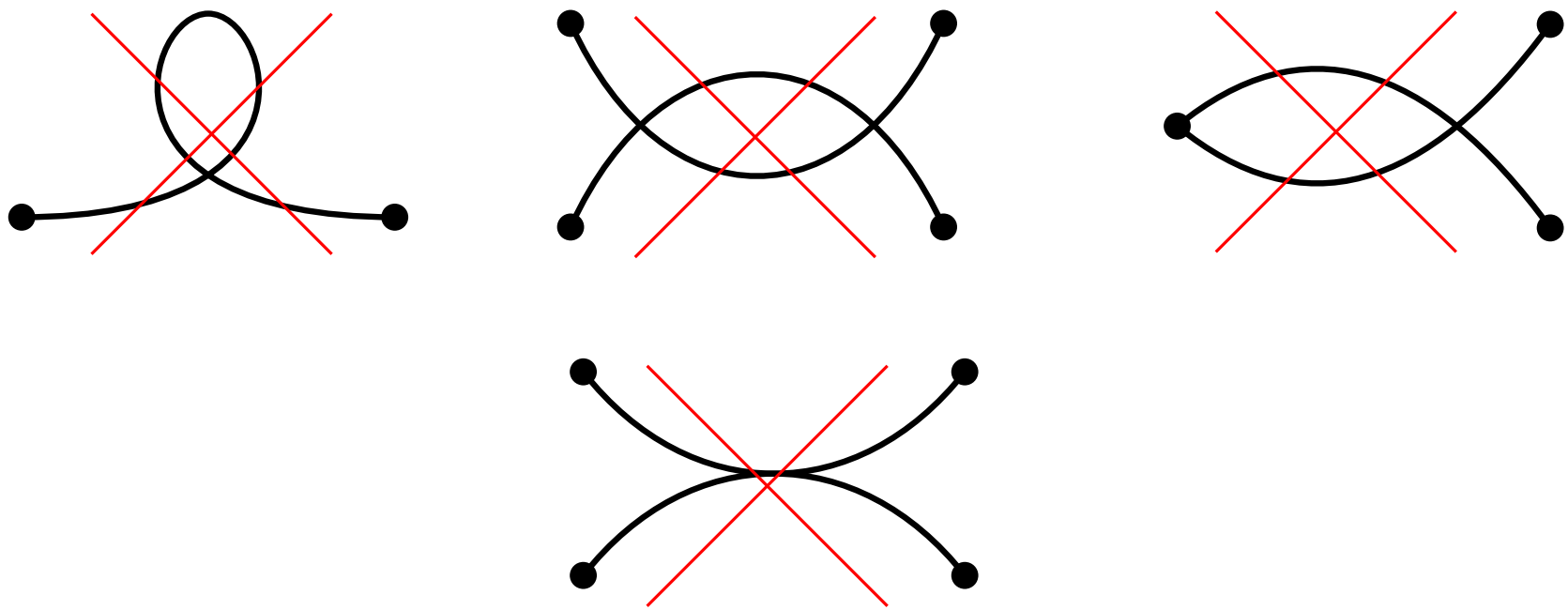


Krupp

Topological Drawings of K_n

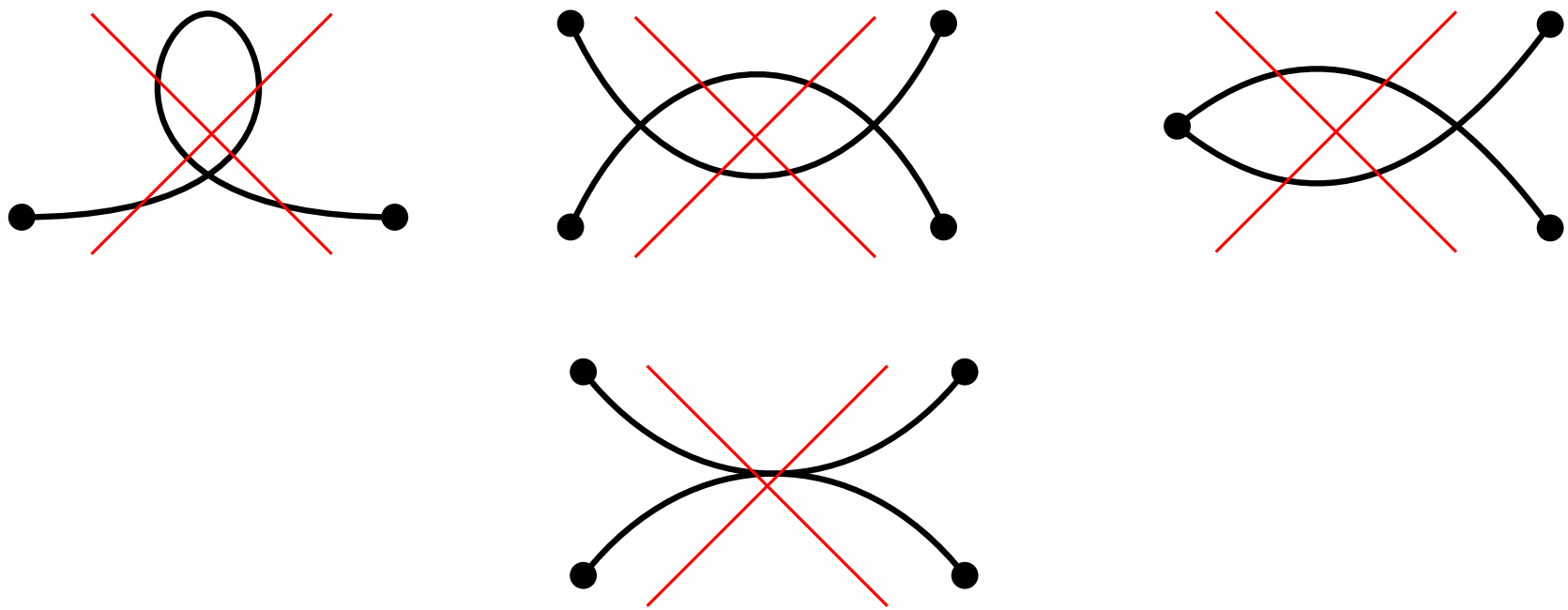
Topological Drawings of K_n

A *(simple) topological drawing* is a drawing of a complete graph without



Topological Drawings of K_n

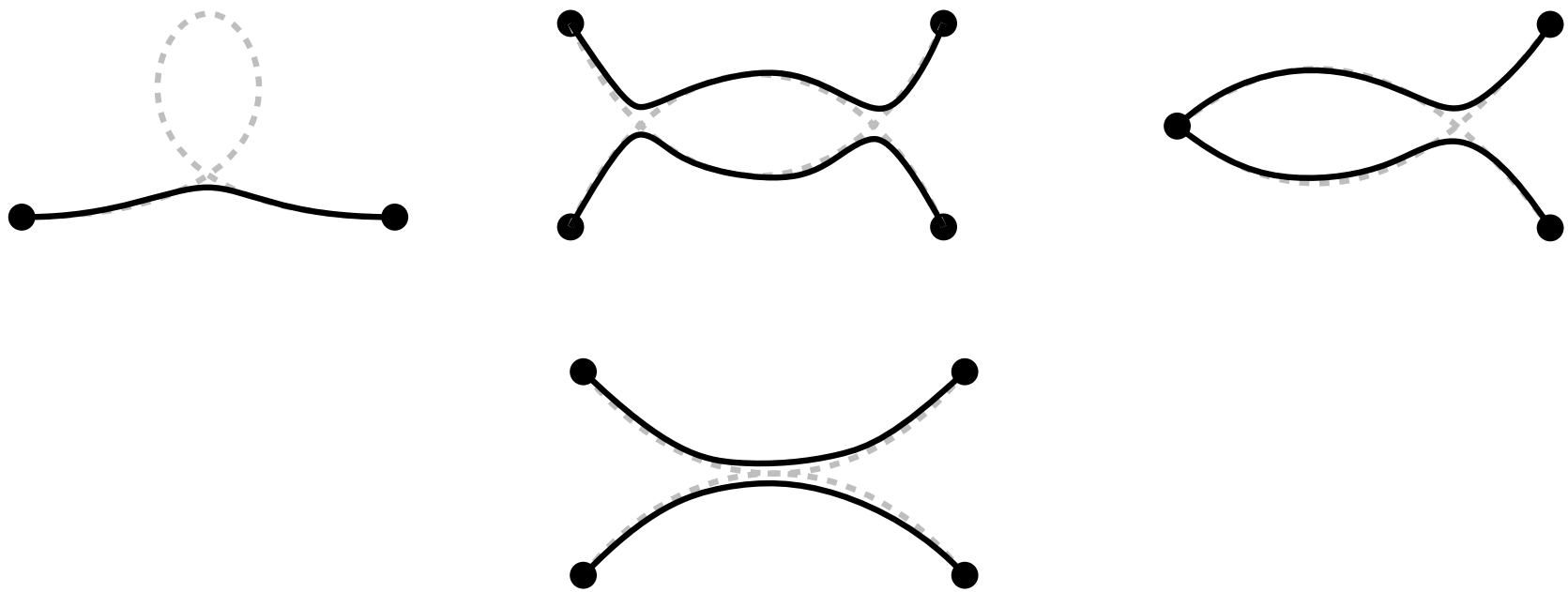
A *(simple) topological drawing* is a drawing of a complete graph without



similar to straight-line and crossing-minimal drawings

Topological Drawings of K_n

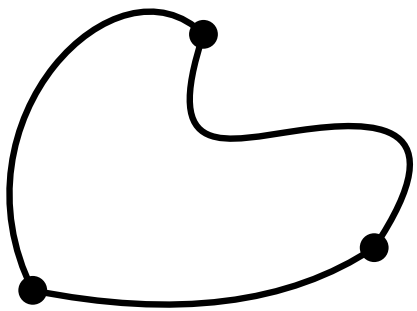
A *(simple) topological drawing* is a drawing of a complete graph without



similar to straight-line and crossing-minimal drawings

Topological Drawings of K_n

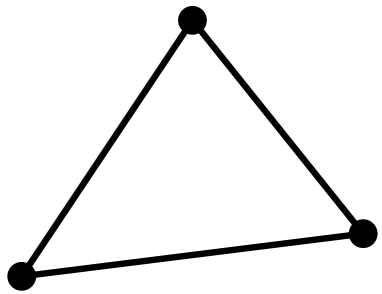
$n = 3$



unique: triangle!

Topological Drawings of K_n

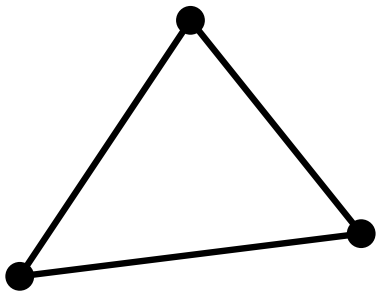
$n = 3$



unique: triangle!

Topological Drawings of K_n

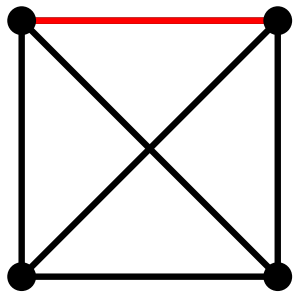
$n = 3$



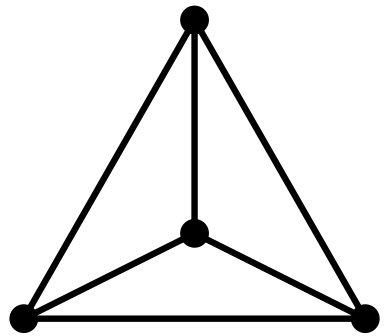
unique: triangle!

$n = 4$

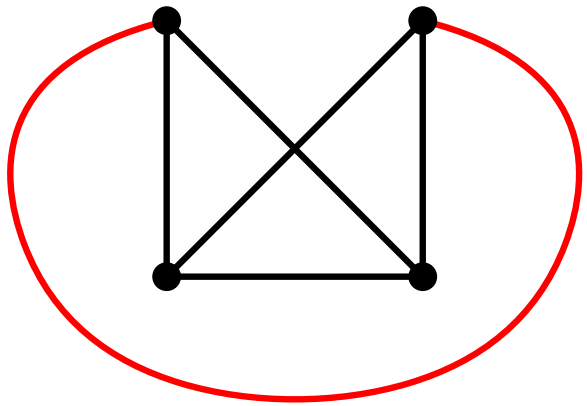
(1 crossing)



(no crossing)

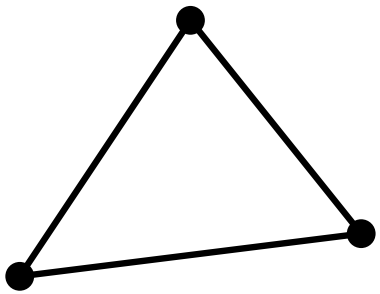


$=$



Topological Drawings of K_n

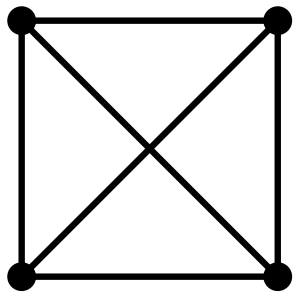
$n = 3$



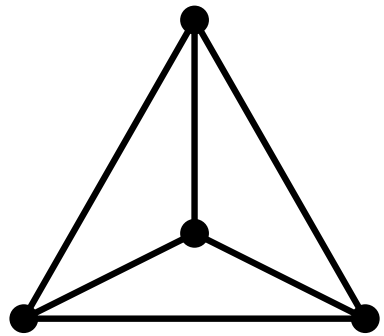
unique: triangle!

$n = 4$

(1 crossing)



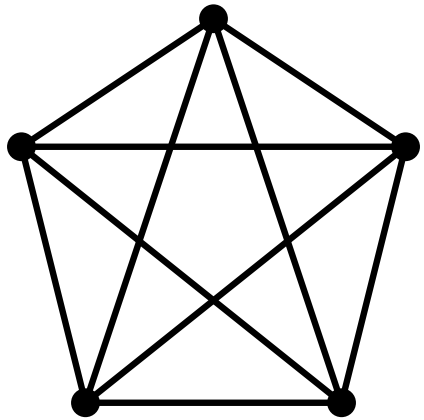
(no crossing)



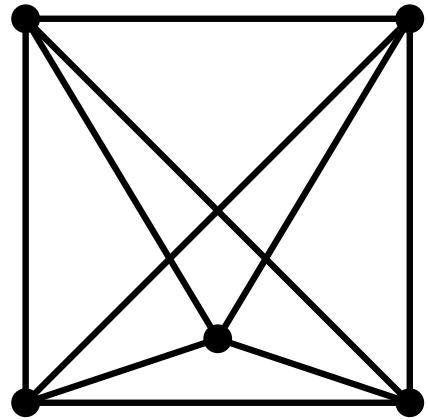
Topological Drawings of K_n

$n = 5$

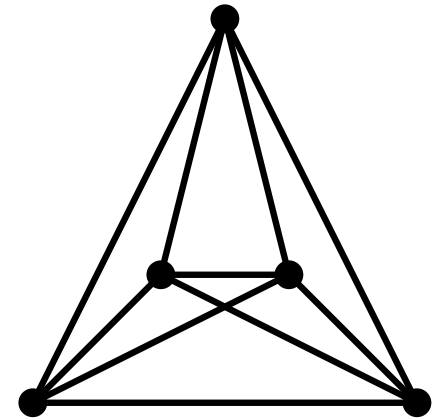
(5 crossings)

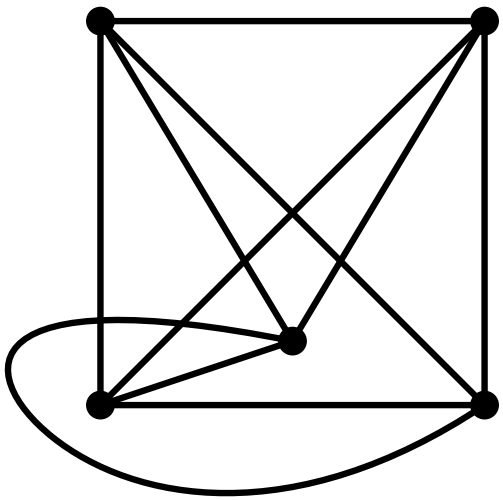


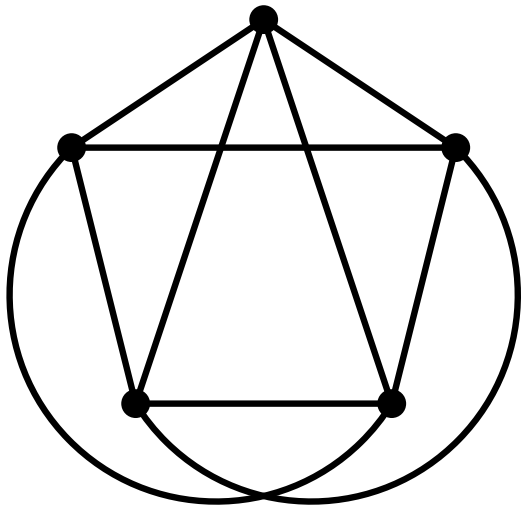
(3 crossings)



(1 crossing)



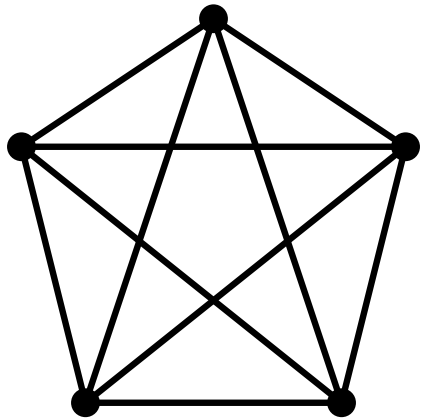




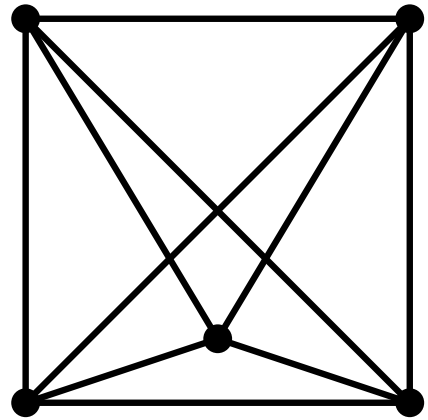
Topological Drawings of K_n

$n = 5$

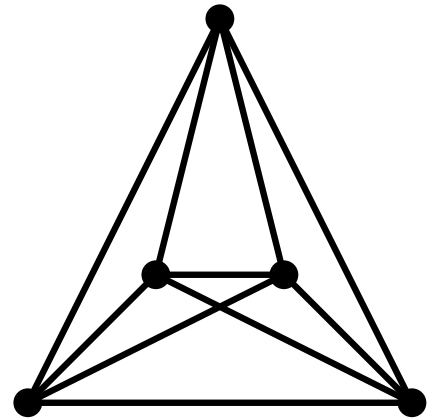
(5 crossings)



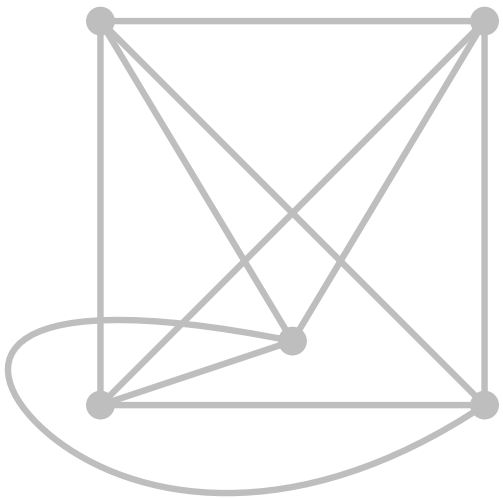
(3 crossings)



(1 crossing)

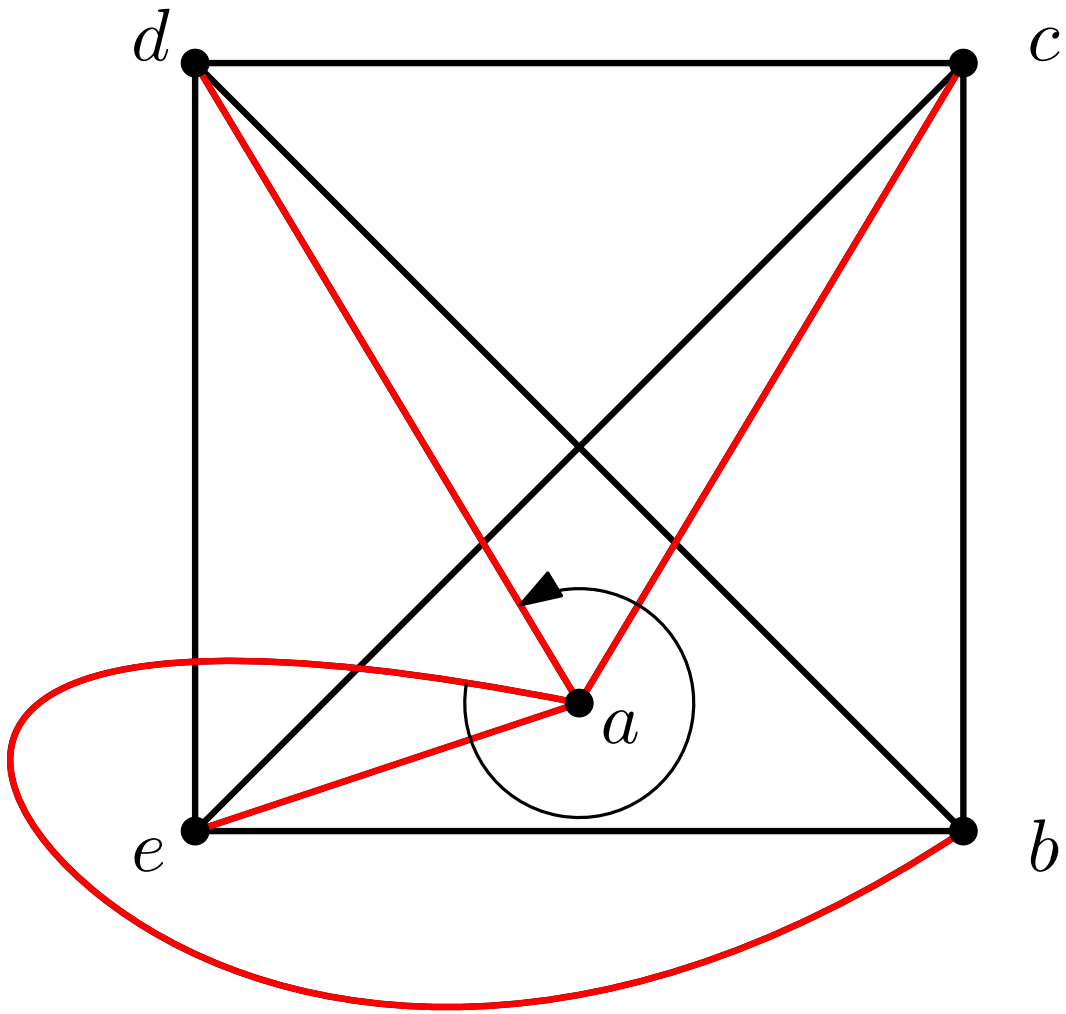


straight-line / pseudolinear



Rotation Systems

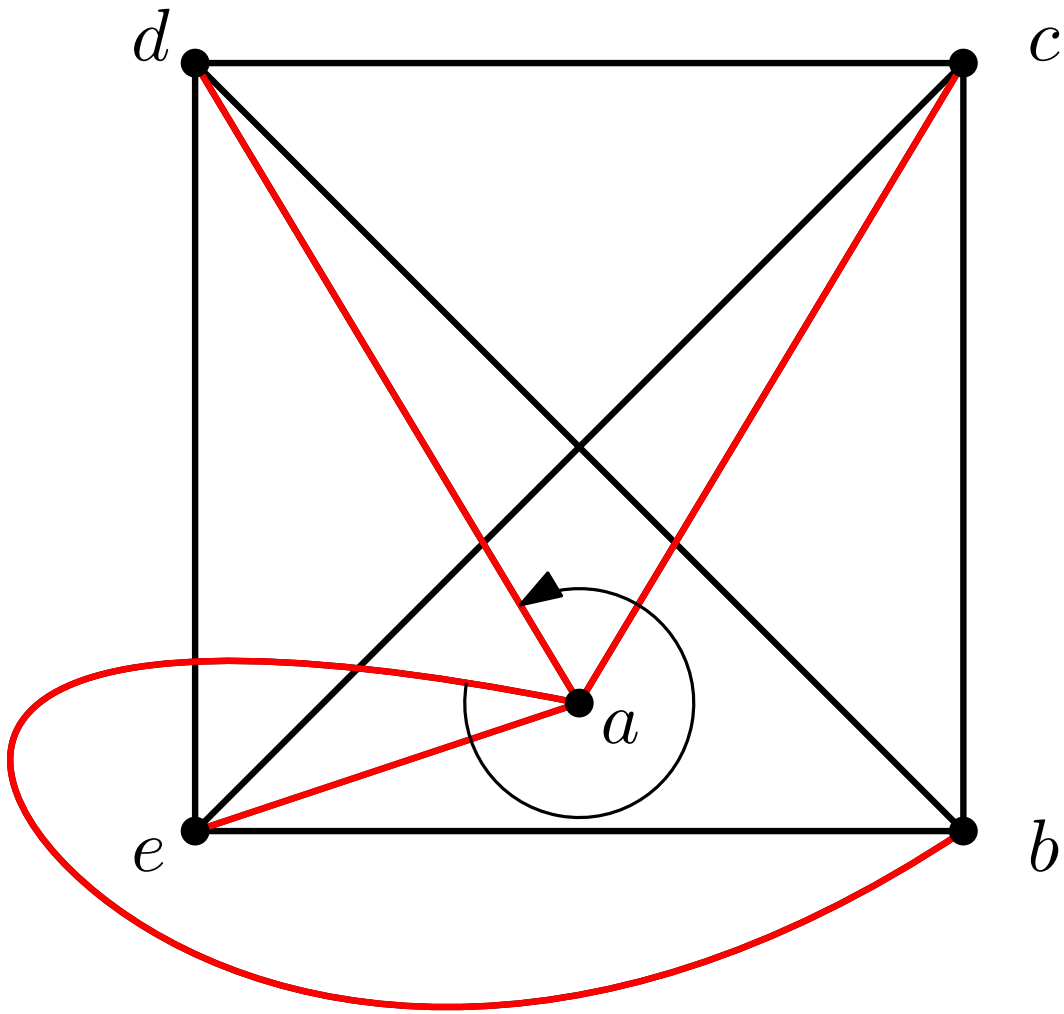
rotation system: cyclic order around each vertex



a sees b, e, c, d

Rotation Systems

rotation system: cyclic order around each vertex



a sees b, e, c, d

b sees a, c, d, e

c sees a, b, d, e

d sees a, b, c, e

e sees e, a, b, c

Rotation Systems

rotation system: cyclic order around each vertex

Kynčl 2015: rotation system representable by topological drawing $\Leftrightarrow$ all 6-tuples are

Rotation Systems

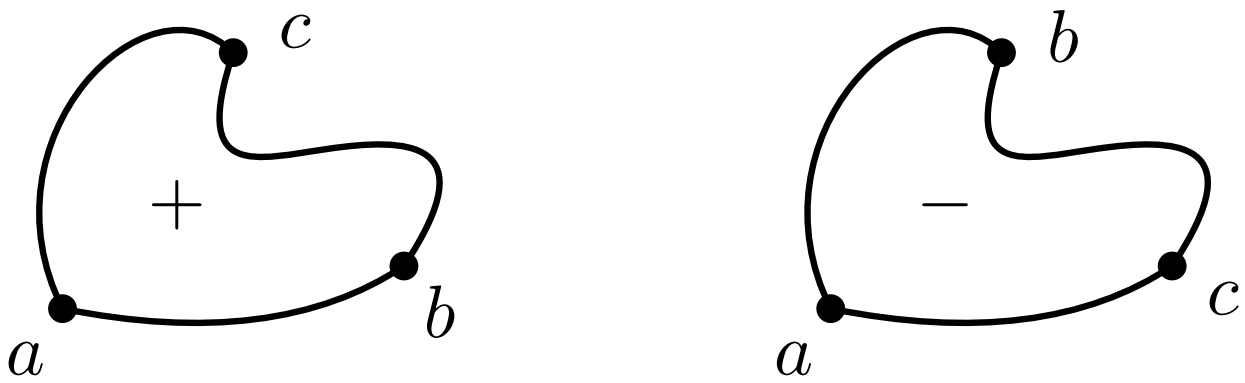
rotation system: cyclic order around each vertex

Kynčl 2015: rotation system representable by topological drawing $\Leftrightarrow$ all 6-tuples are

$\Leftrightarrow$ 5-tuples (Ábrego et al. 2015, Bergold et al. 2022+)

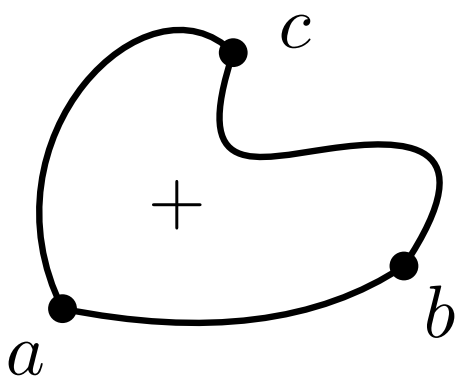
Generalized Signotopes

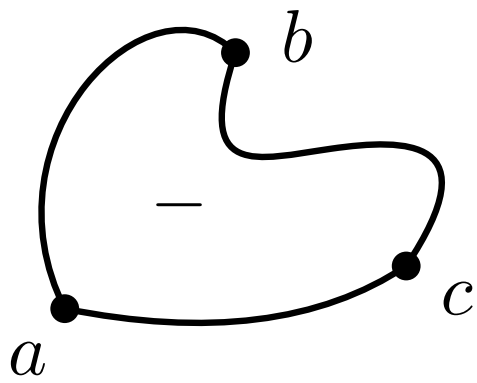
each K_3 has orientation



Generalized Signotopes

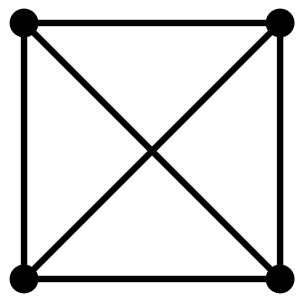
each K_3 has orientation

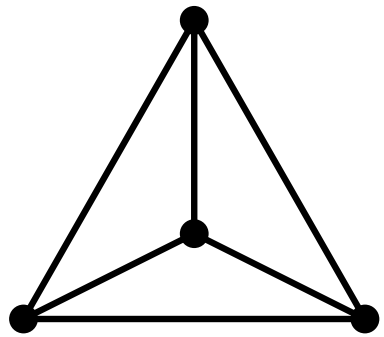




each K_4 gives sequence, except

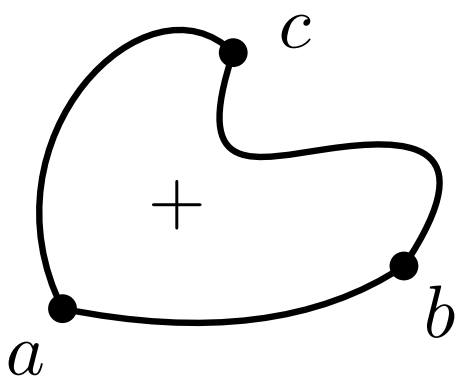
abc	abd	acd	bcd
$+$	$-$	$+$	$-$
$-$	$+$	$-$	$+$

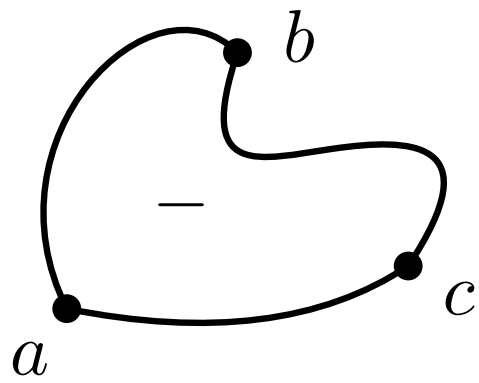




Generalized Signotopes

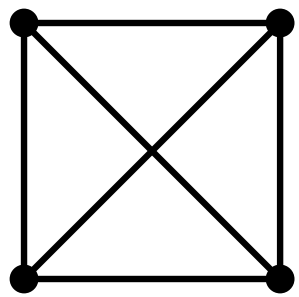
each K_3 has orientation

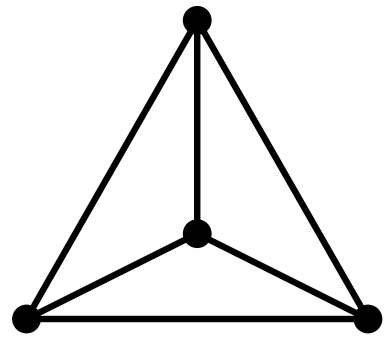




each K_4 gives sequence, except

abc	abd	acd	bcd
$+$	$-$	$+$	$-$
$-$	$+$	$-$	$+$





no characterization known yet

(possibly infinitely many obstructions / NP-hard?)