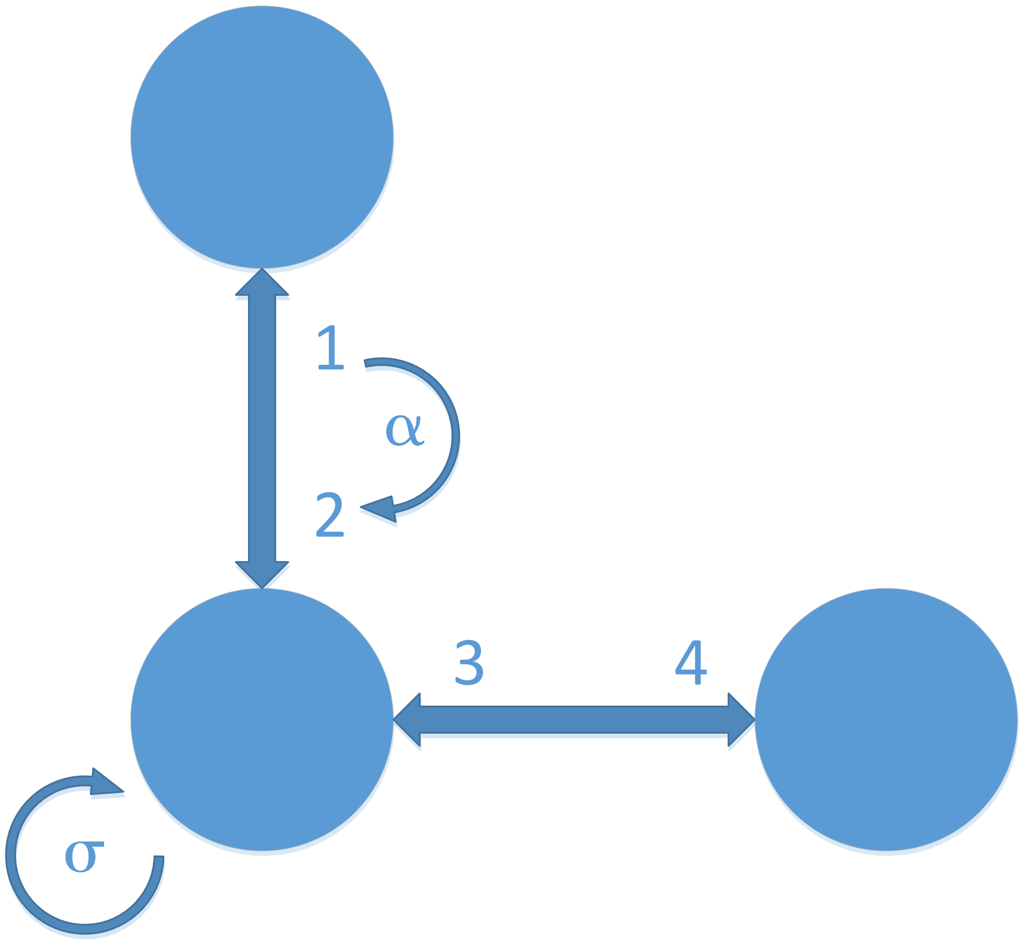
Combinatorial Pyramids – Development and Lessons Learned

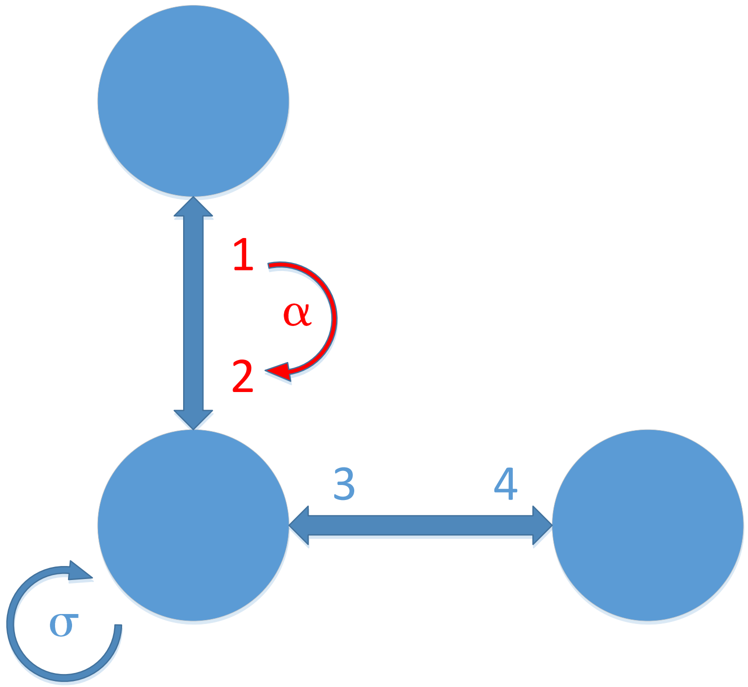
David Pfahler

Structural Pattern Recognition

Recap Combinatorial Maps

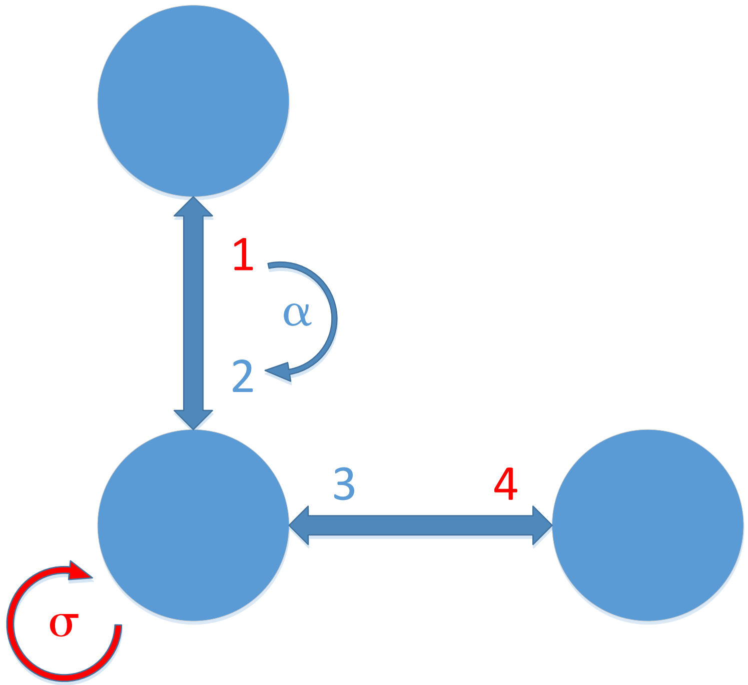


Recap Combinatorial Maps



	1	2	3	4
α	2	1	4	3
σ	4	2	3	1

Recap Combinatorial Maps



	1	2	3	4
α	2	1	4	3
σ	4	2	3	1

Development Phases

Phase 1:
Existing Work

↓

Phase 2:
Meet the Maps

↓

Phase 3:
Implementation

↓

Phase 4:
Results

Development Phases

Phase 1:
Existing Work

↓

Phase 2:
Meet the Maps

↓

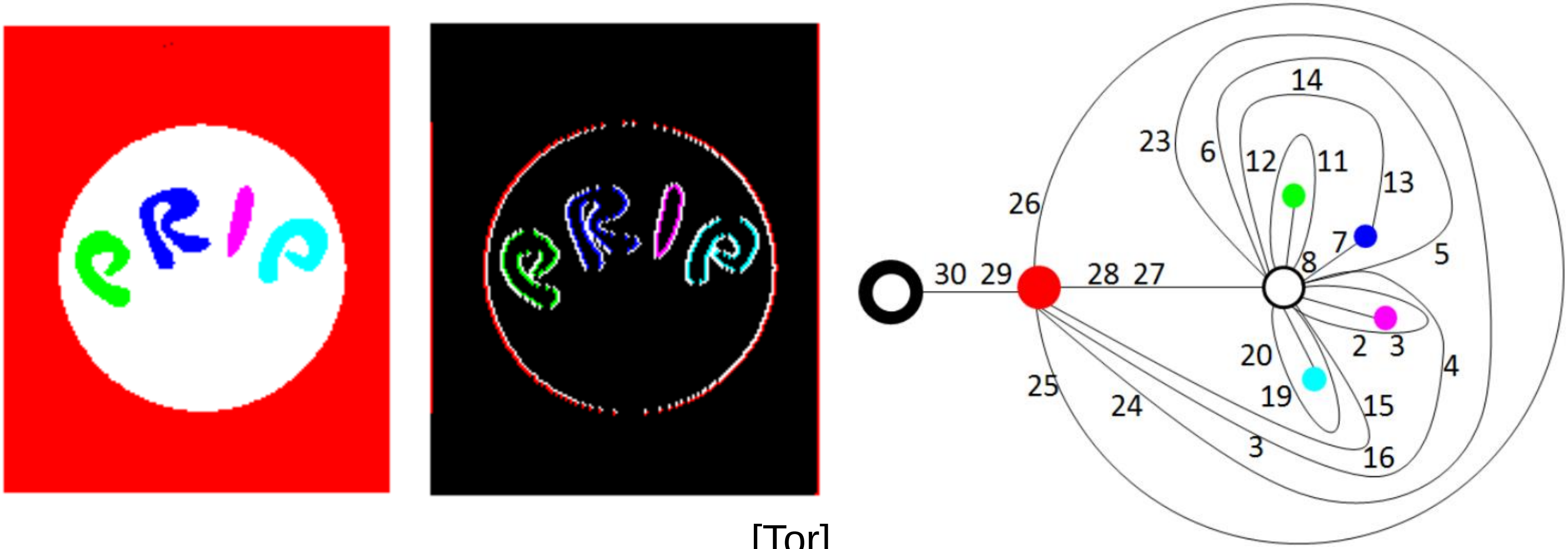
Phase 3:
Implementation

↓

Phase 4:
Results

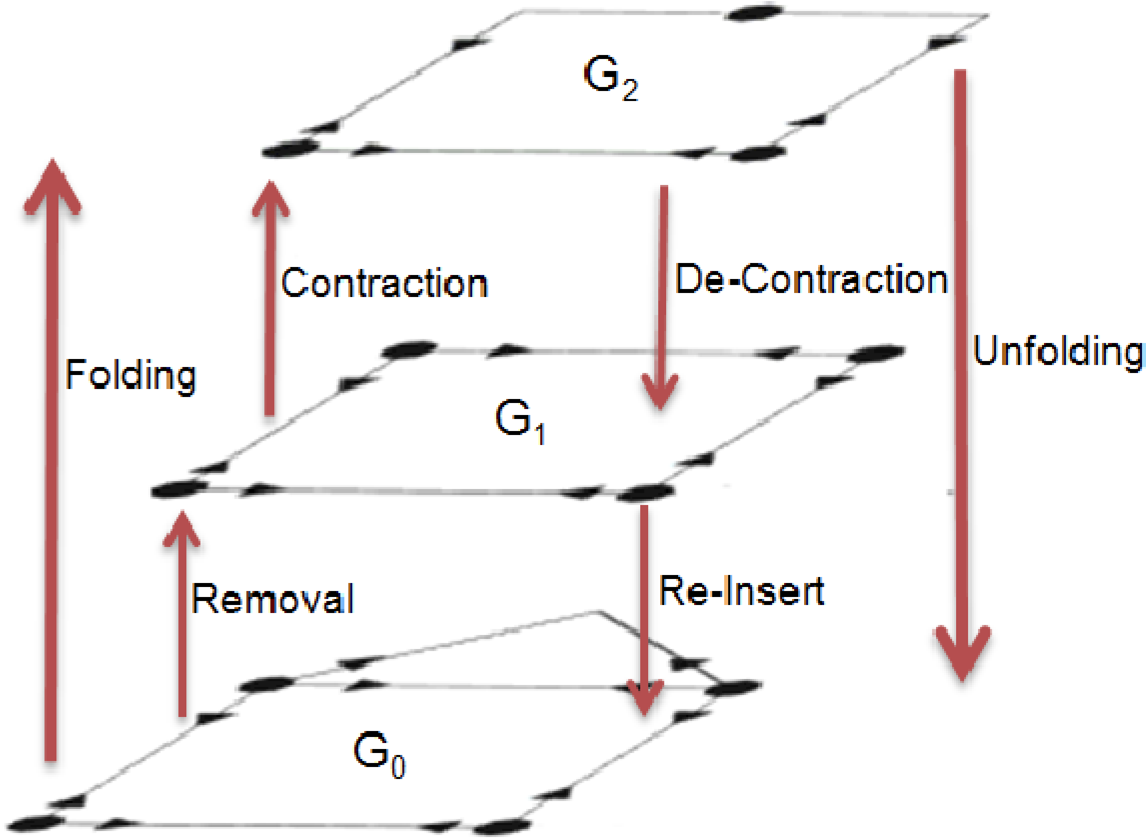
Phase 1: Existing Work

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



[Tor]

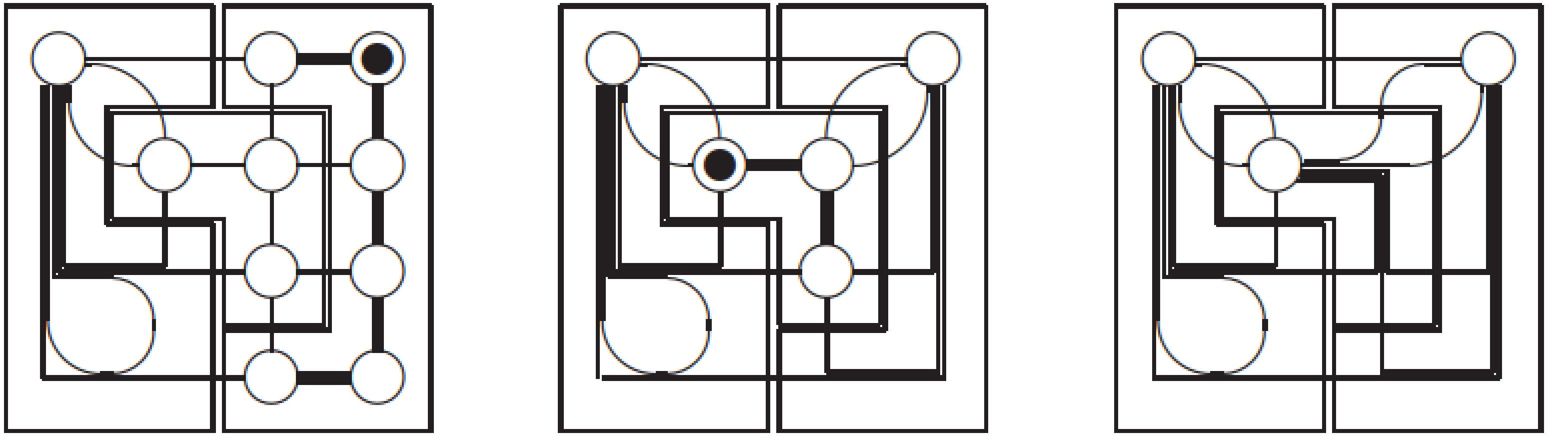
- Recap: Operations on a Combinatorial Map (CM):
  - Remove
  - Contract



[Tor]

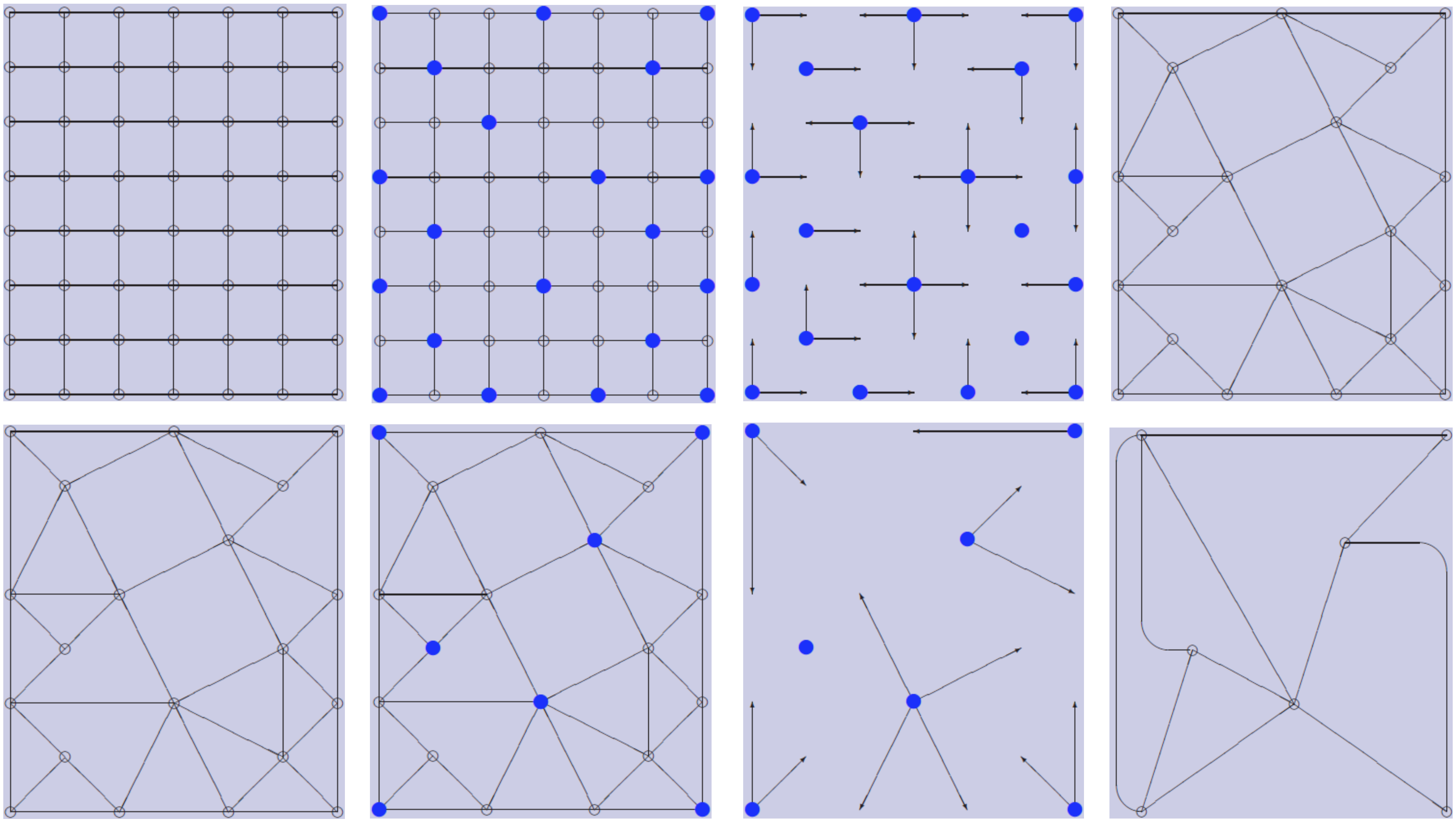
Phase 1: Existing Work

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



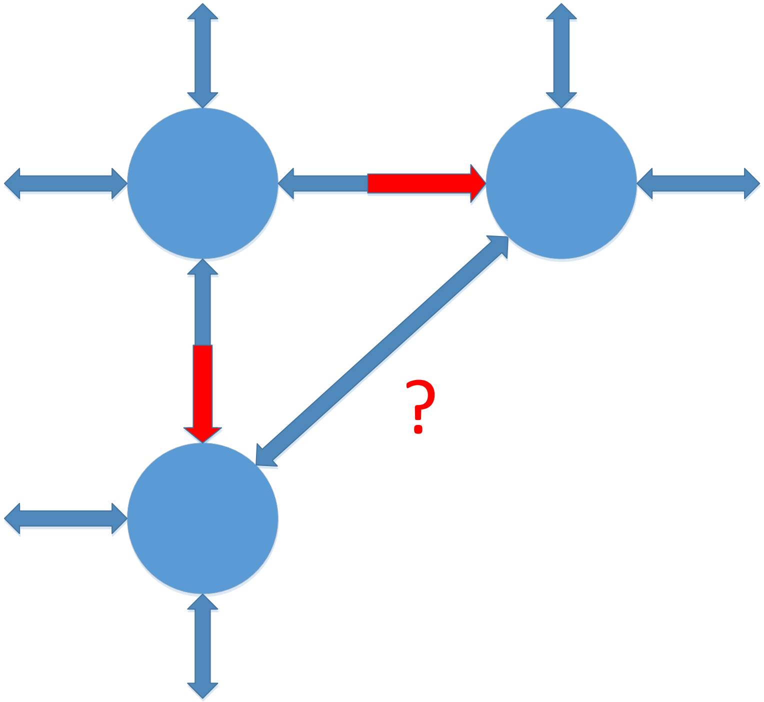
[Bru01]

Recap: Contraction Kernel



Phase 1: Existing Work

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



Development Phases

Phase 1:
Existing Work

Phase 2:
Meet the Maps

Phase 3:
Implementation

Phase 4:
Results

Phase 2: Meet the Maps

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

```
import CombinatorialMap
map=CombinatorialMap()
map.setSize(5, 5)
map.printNodes()
```

Phase 2: Meet the Maps

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

Phase 2: Meet the Maps

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

Phase 2: Meet the Maps

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

```
nodes:
     0 -  5, 2 -     0 -  7, 4 -     0
     |               |               |
    1, 8            3,11            6,15
     |               |               |
     0 - 14,10 -     0 - 17,13 -     0
     |               |               |
    9,18           12,20           16,23
     |               |               |
     0 - 22,19 -     0 - 24,21 -     0
```

d | σ(d) | σ(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
```

Development Phases

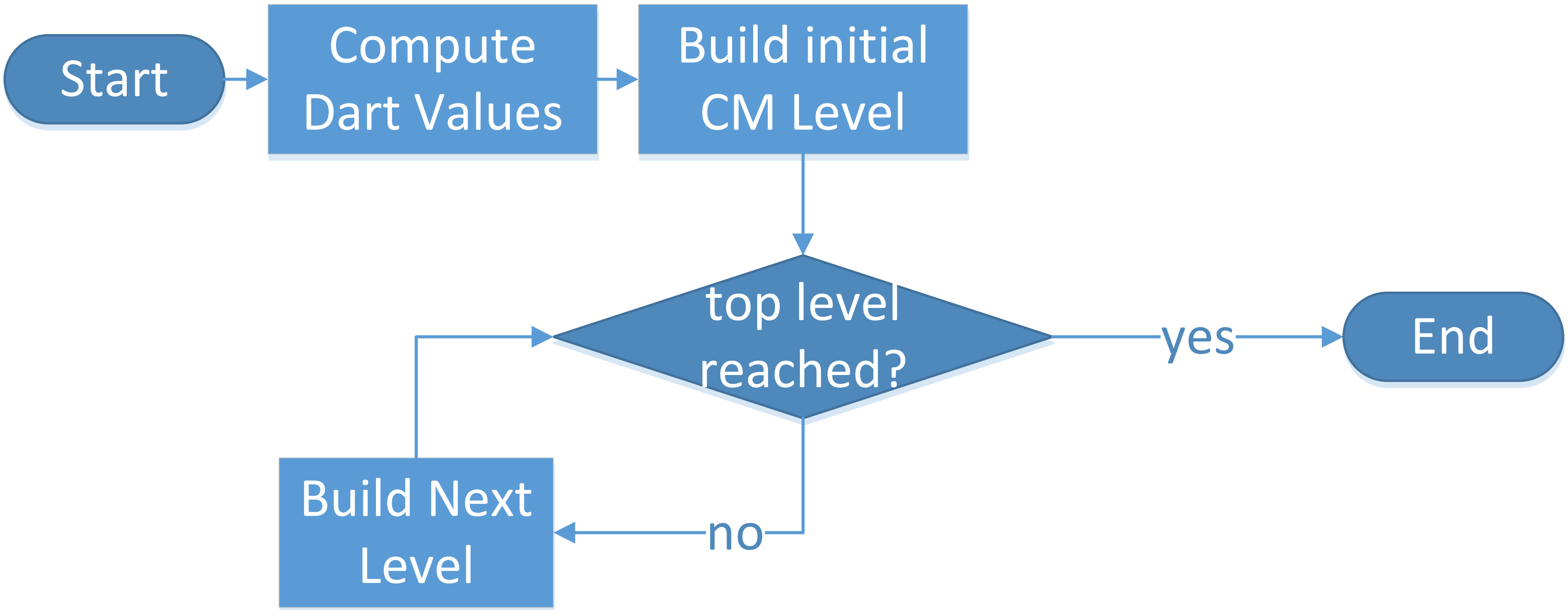
Phase 1:
Existing Work

Phase 2:
Meet the Maps

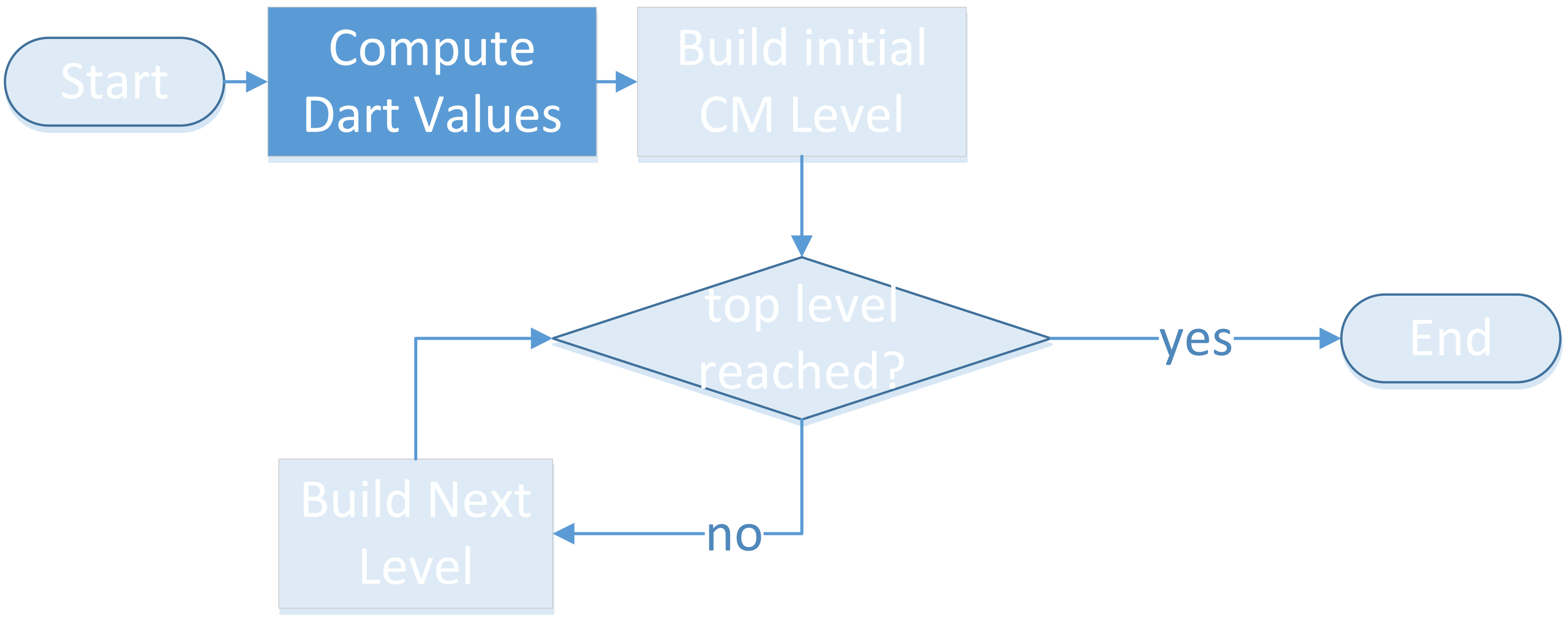
Phase 3:
Implementation

Phase 4:
Results

Phase 3: Implementation

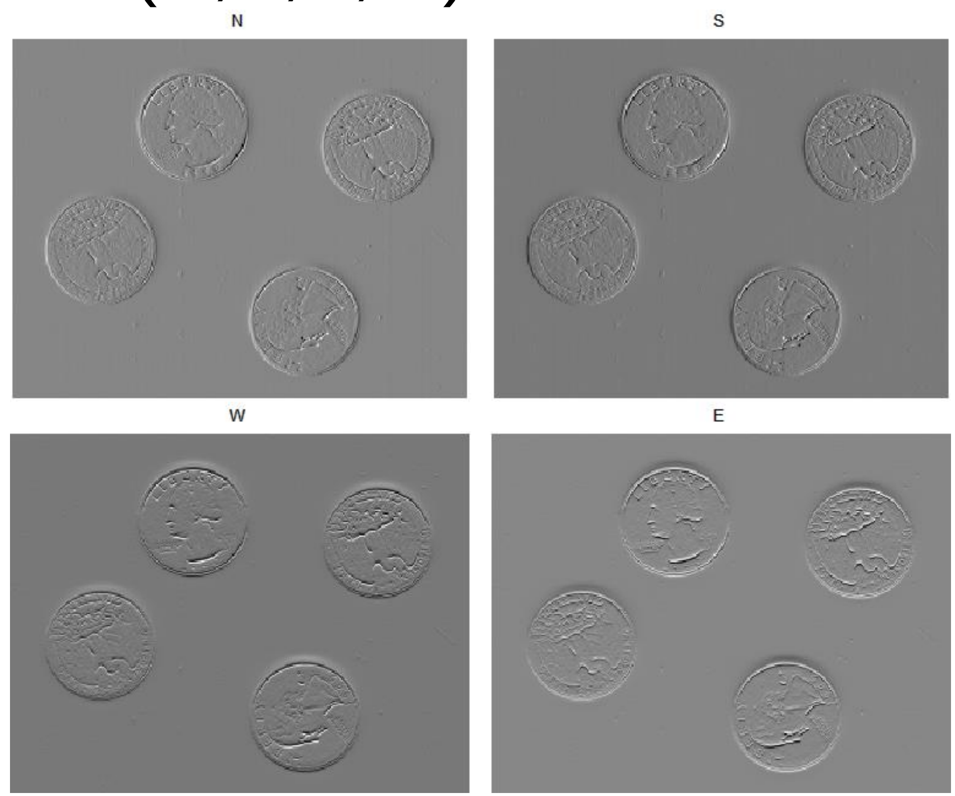


Phase 3: Implementation

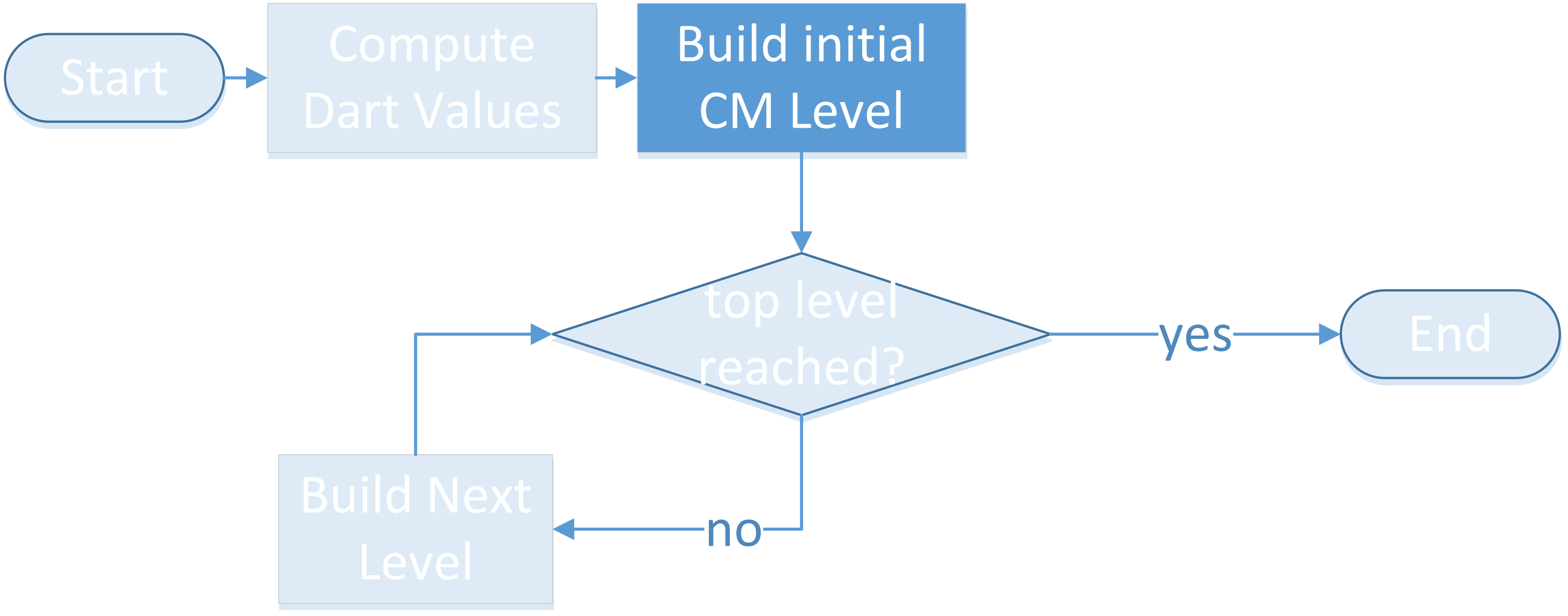


Compute Dart Values

- 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)



Phase 3: Implementation



Build Initial CP Level

- 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

Build Initial CP Level

- 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	σ(d)	σ(d)-d
1	2	1
2	1	-1
3	5	2
4	3	-1
5	4	-1

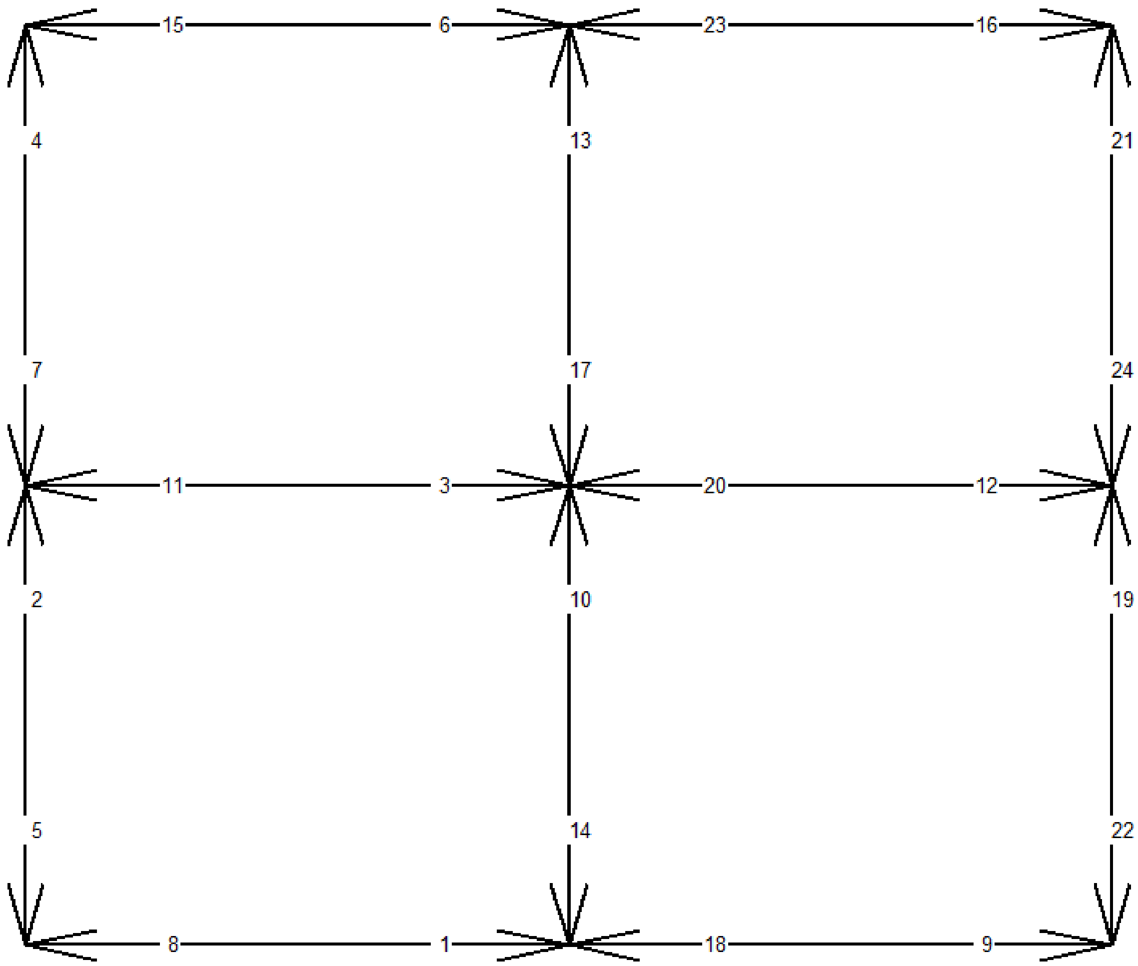
20	21	1
21	22	1
22	20	-2
23	24	1
24	23	-1

Build Initial CP Level

- Involution α
  - `cm.involution(N) = S`
  - `cm.involution(E) = W`
  - …
- Previous Dart ρ
  - `cm.prev(cm.next) = 1:num_darts`

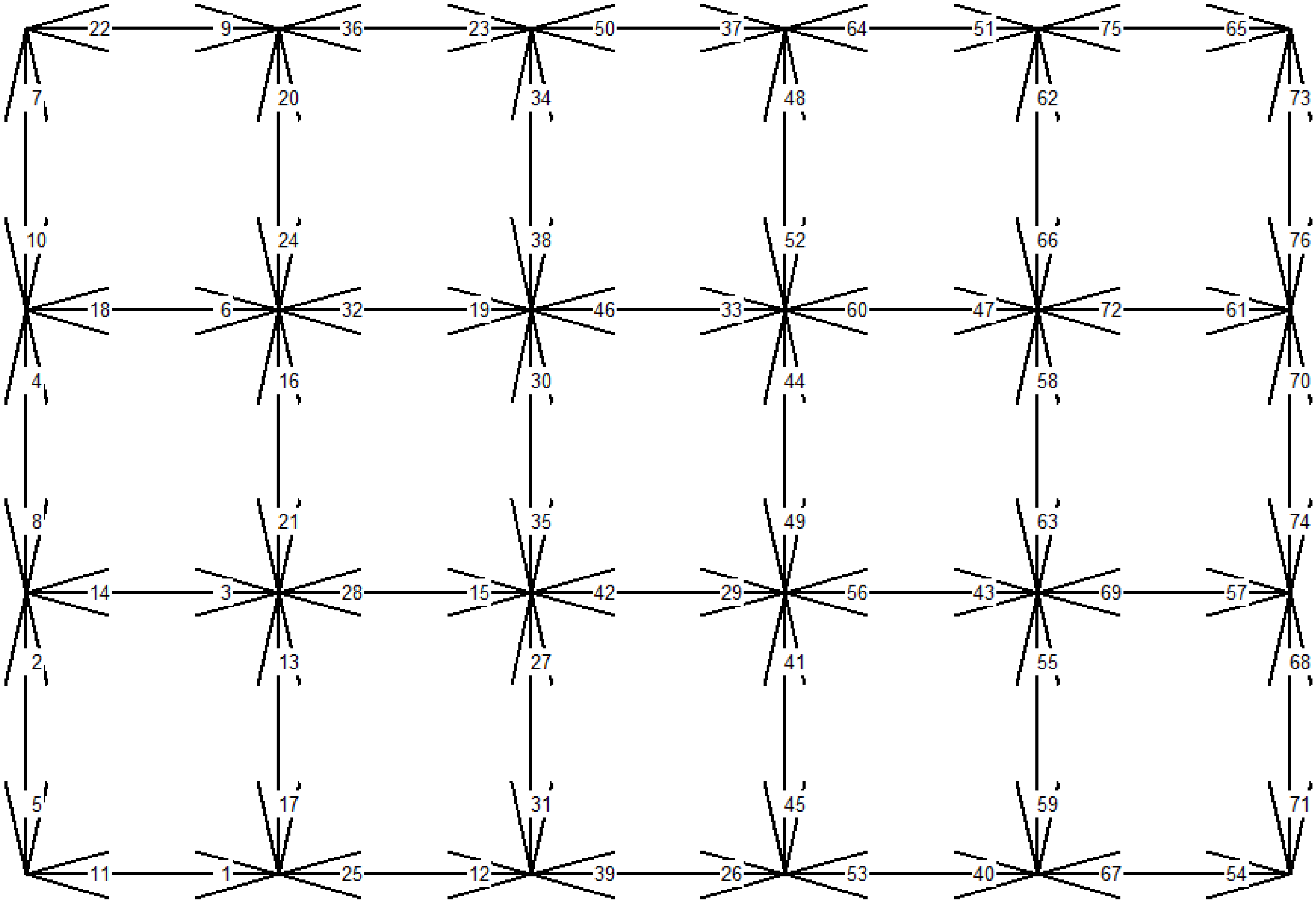
Build Initial CP Level

- Some Examples (3x3):



Build Initial CP Level

- Some Examples (4x6):

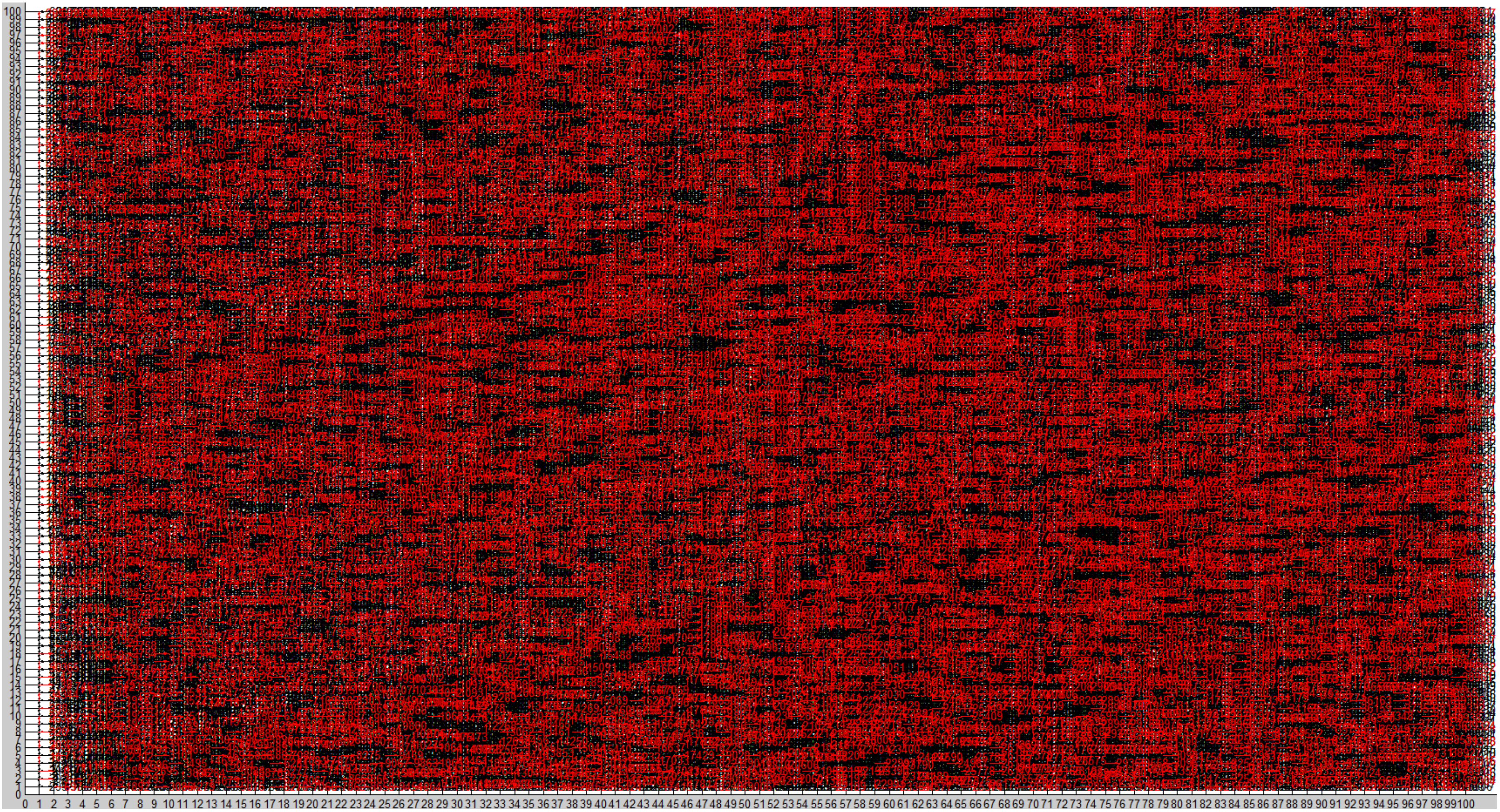


Build Initial CP Level

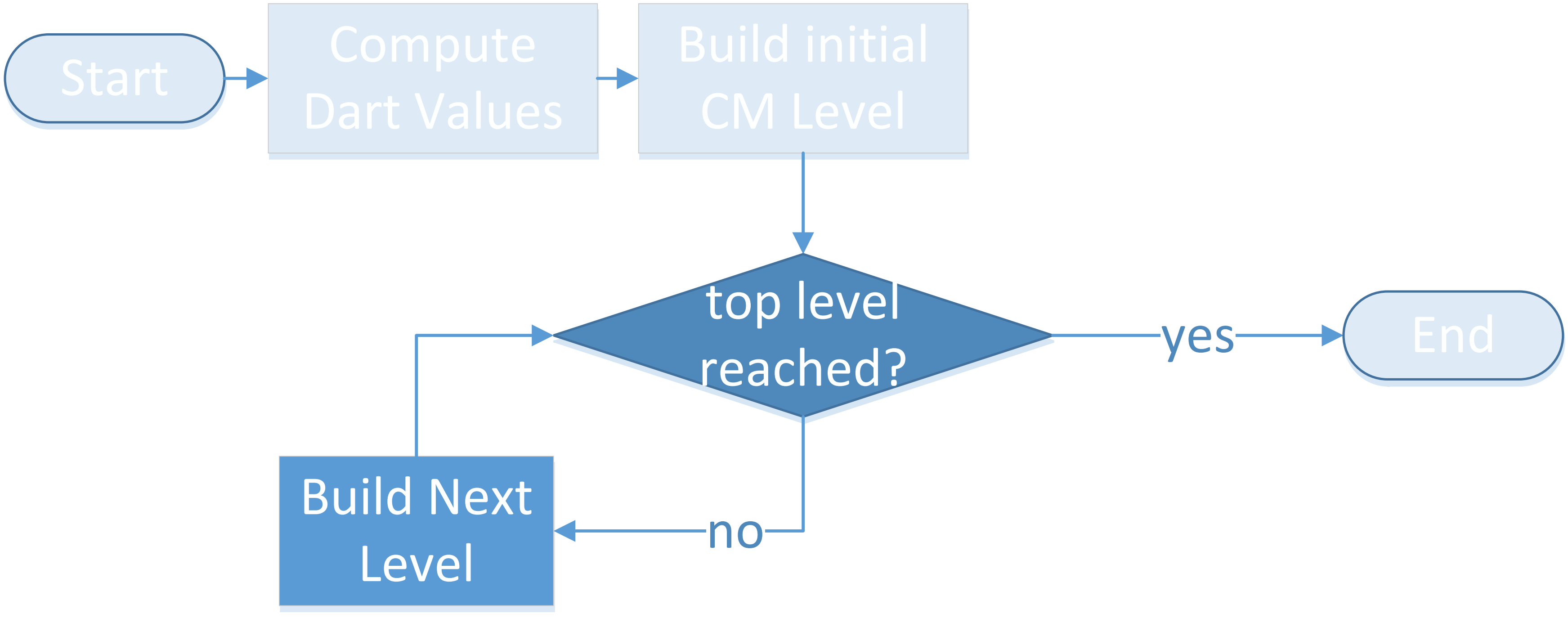
- Some Examples (20x20):

Build Initial CP Level

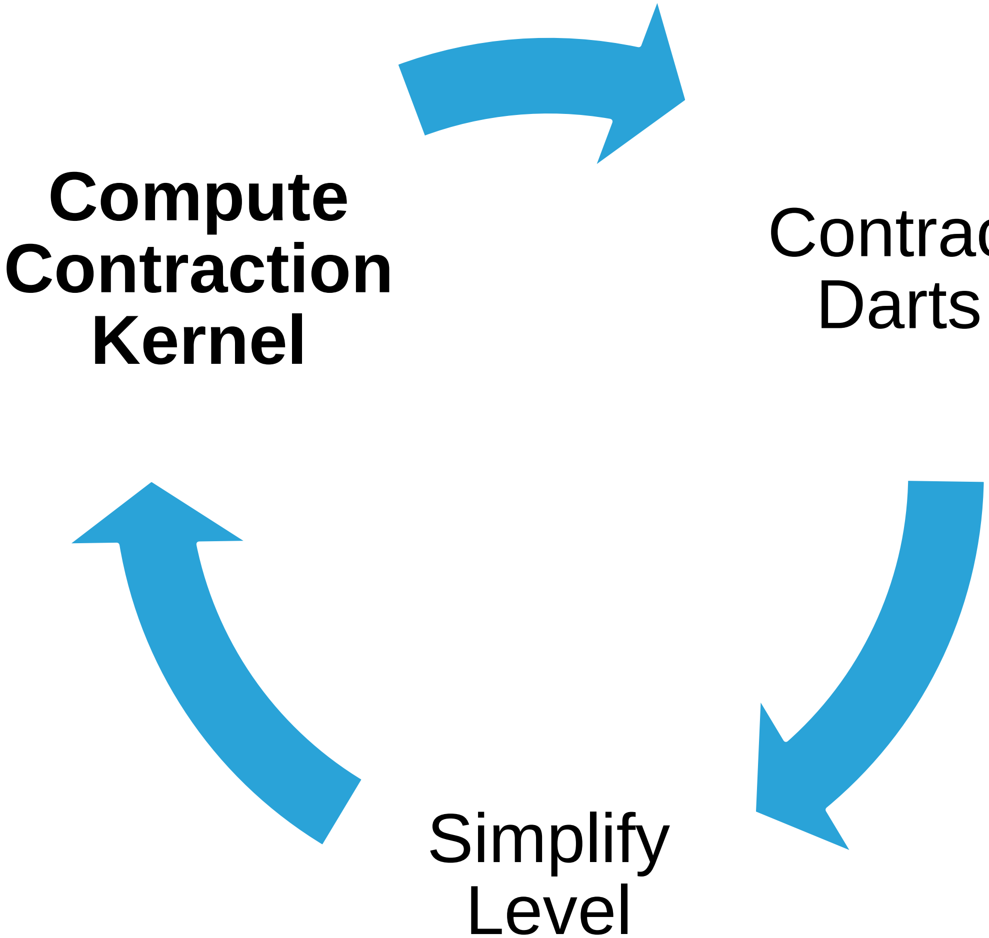
- Some Examples (100x100):



Phase 3: Implementation

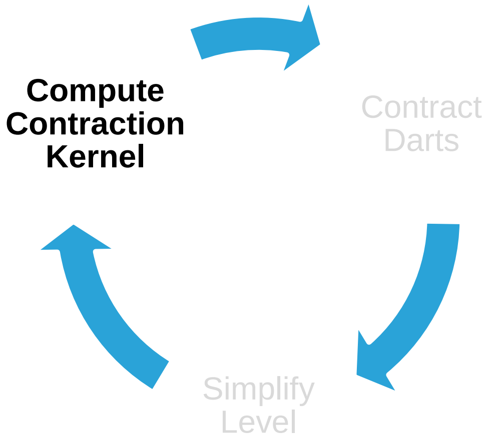


CP Creation


Compute Contraction Kernel
Contract Darts
Simplify Level

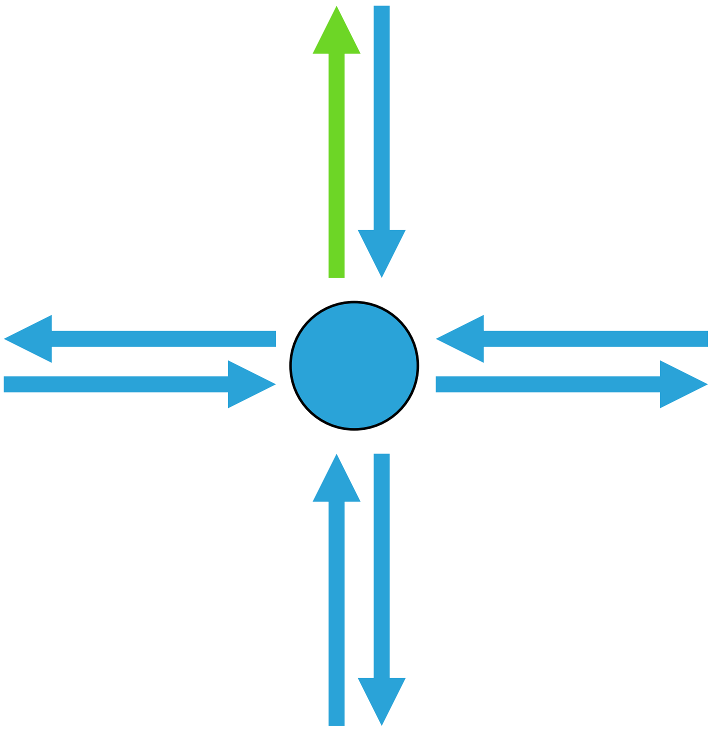
Compute the Contraction Kernel

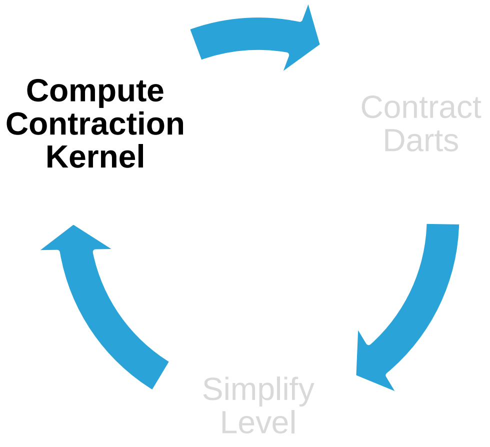
- Sort the darts by its values
- Add dart to contraction kernel if:
  - Not self loop
  - Not pending edge



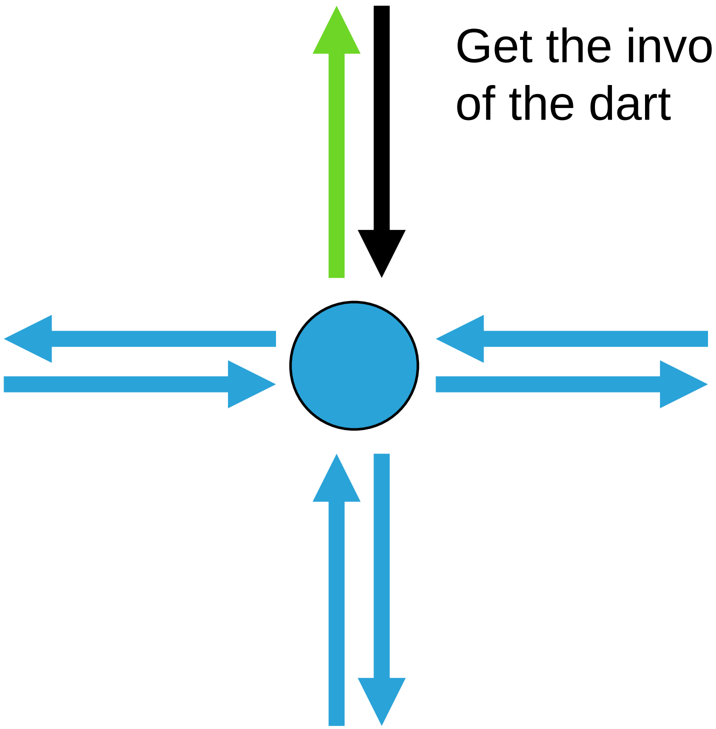
Compute the Contraction Kernel

Greedy: Assign next best Dart For Contraction Kernel

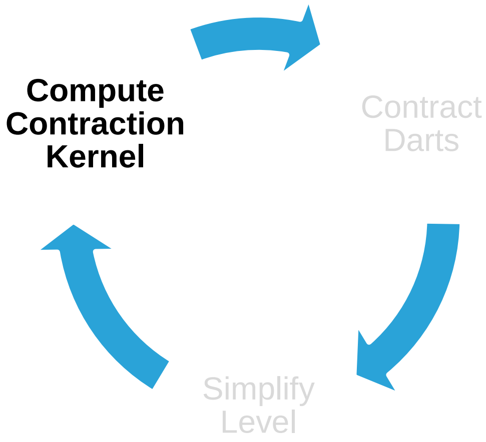




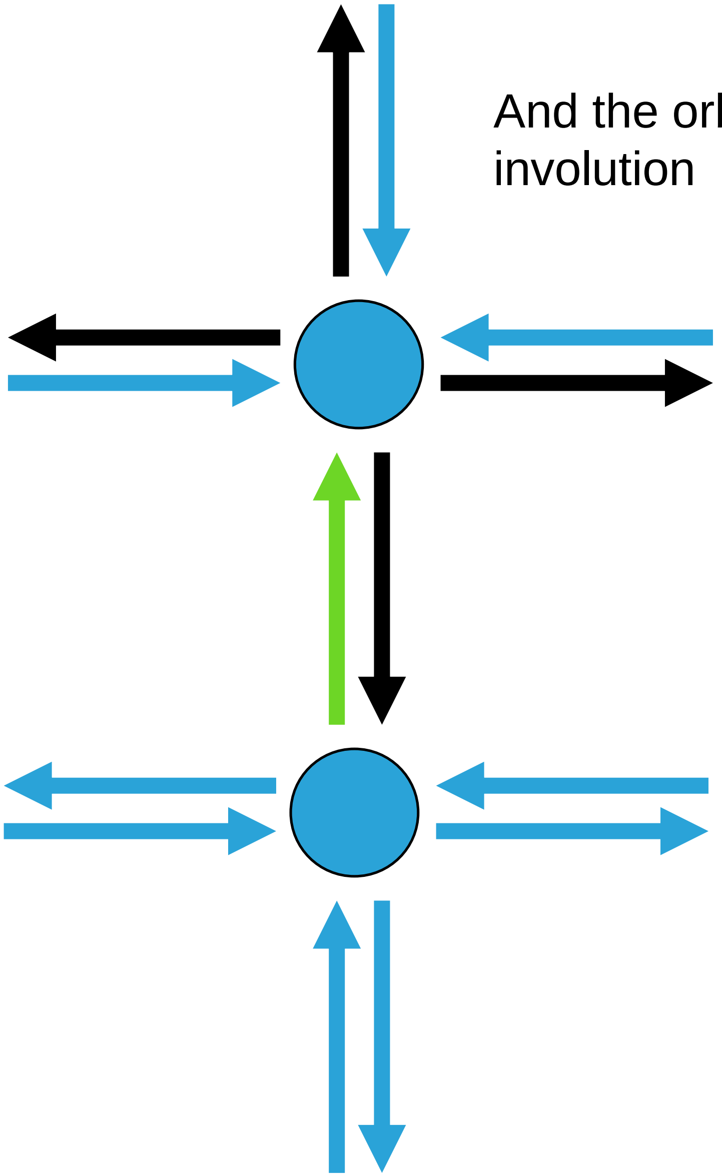
Compute the Contraction Kernel



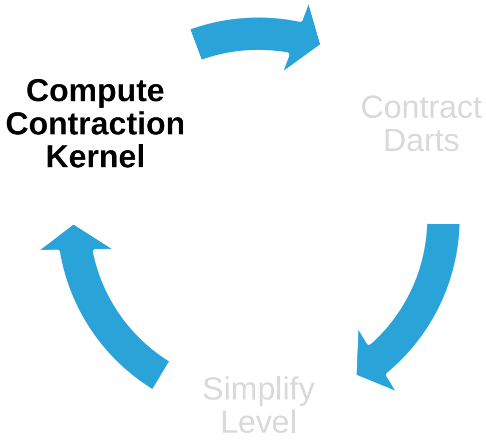
Get the involution
of the dart



Compute the Contraction Kernel

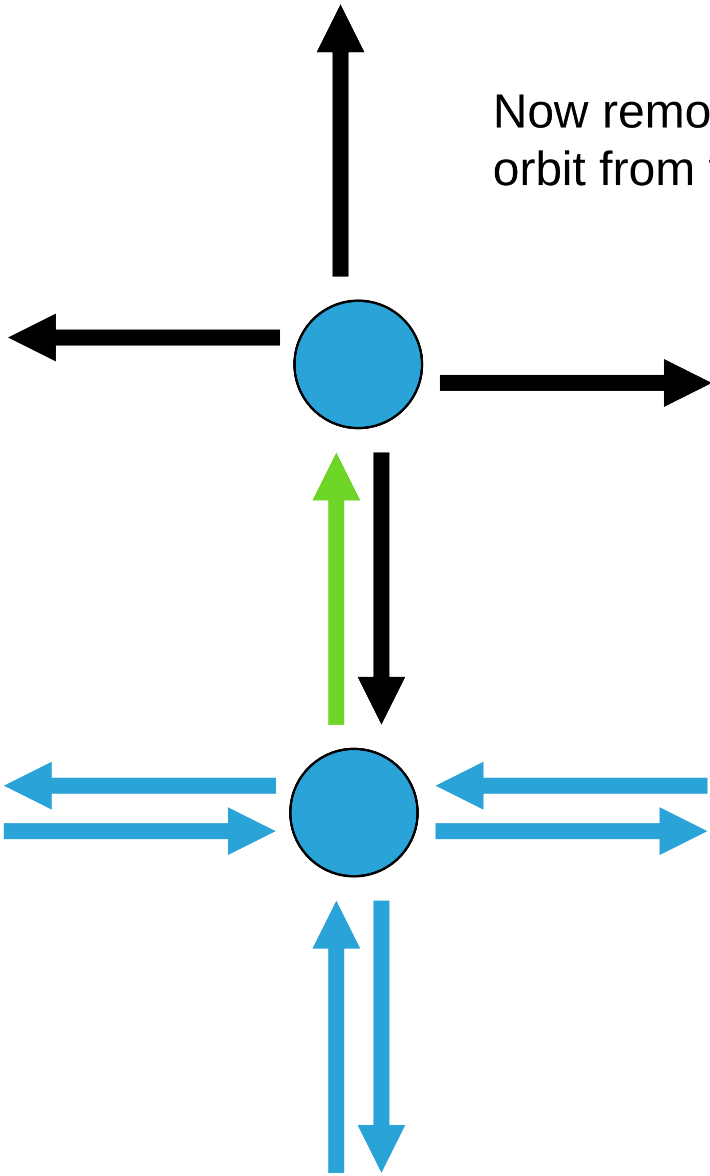


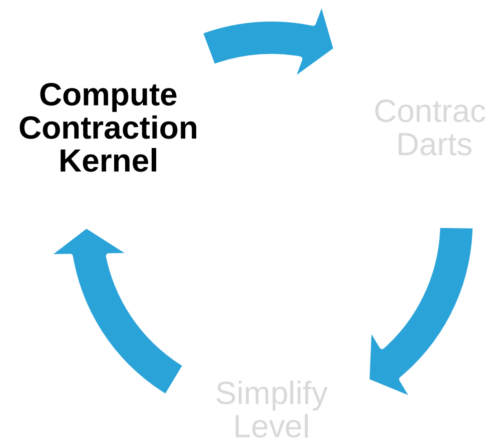
And the orbit of the involution



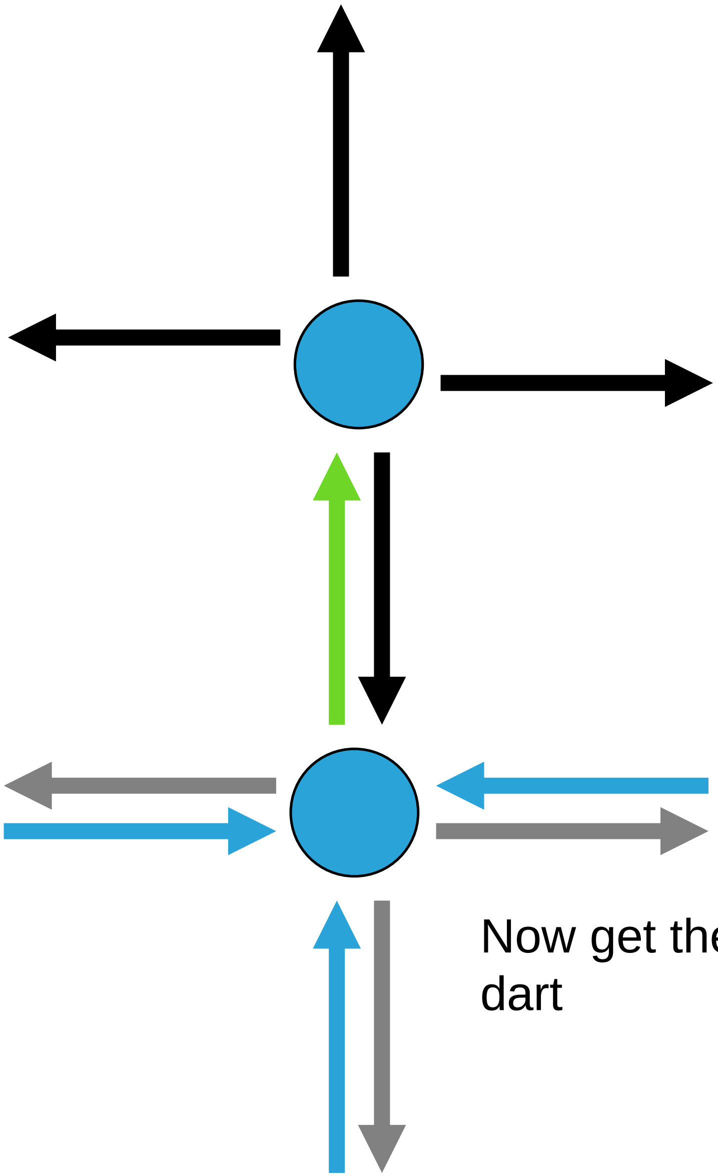
Compute the Contraction Kernel

Now remove the involution of the involution orbit from the set of valid darts

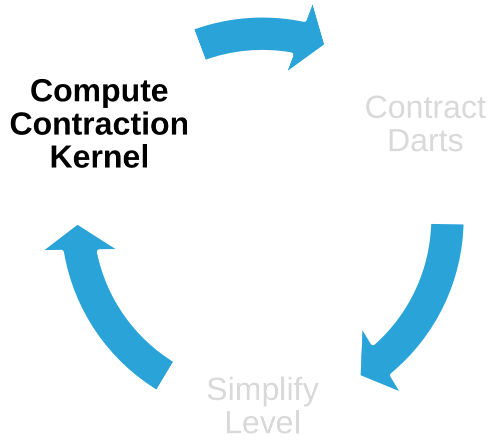




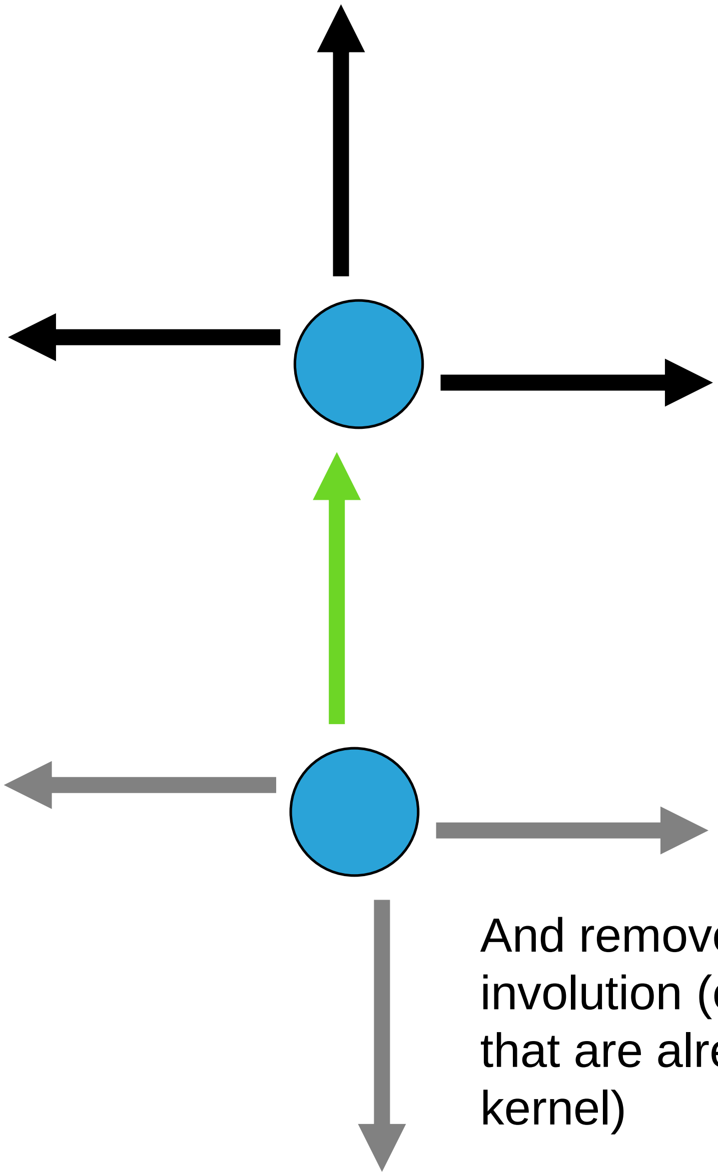
Compute the Contraction Kernel



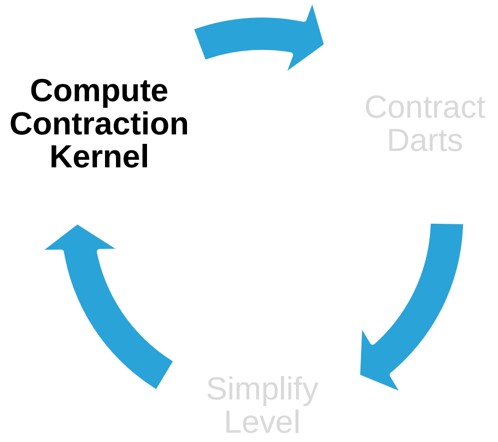
Now get the orbit of the dart



Compute the Contraction Kernel

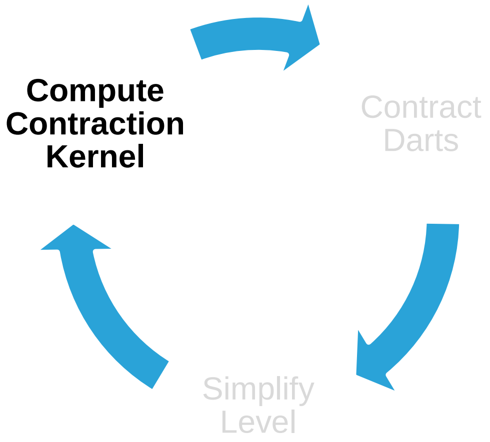


And remove its involution (except darts that are already in the kernel)

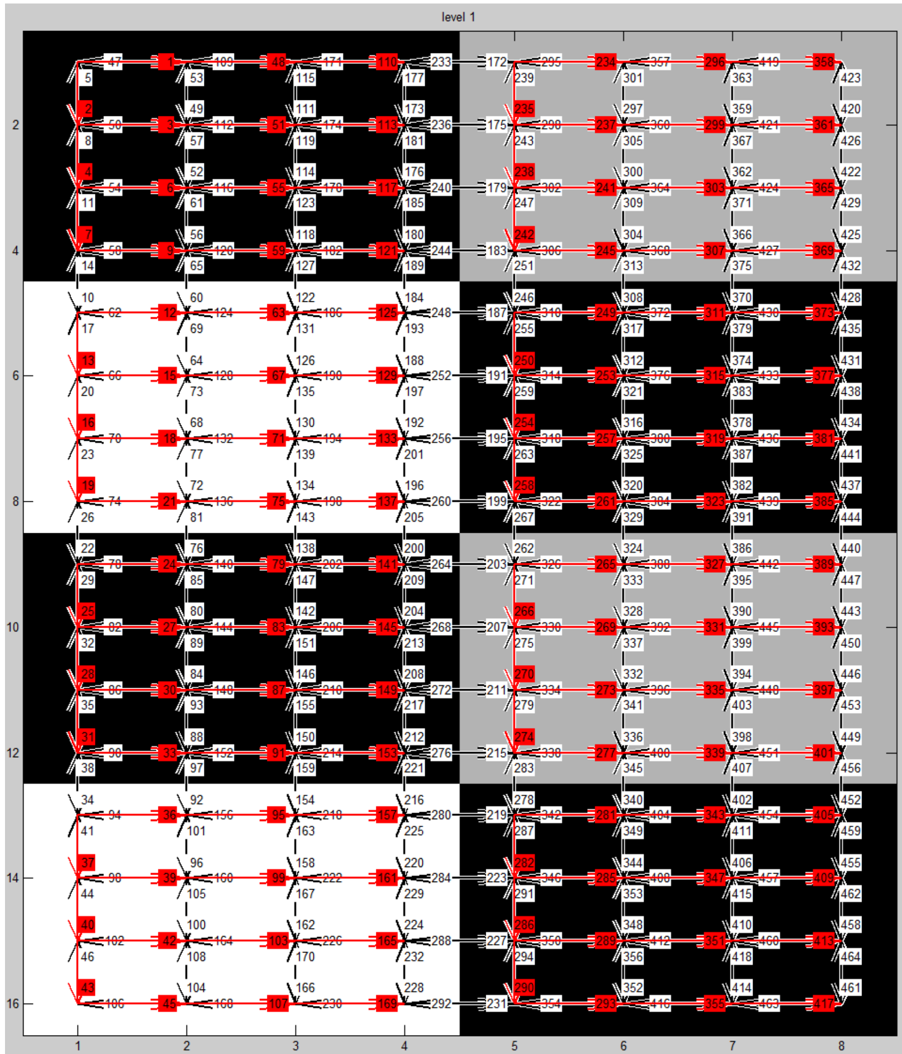


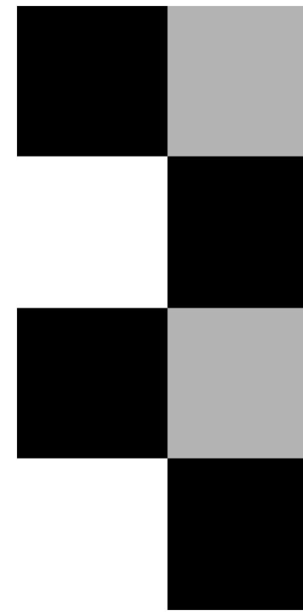
Compute the Contraction Kernel

- Exceptions:
  - Self loops
  - Pending edges



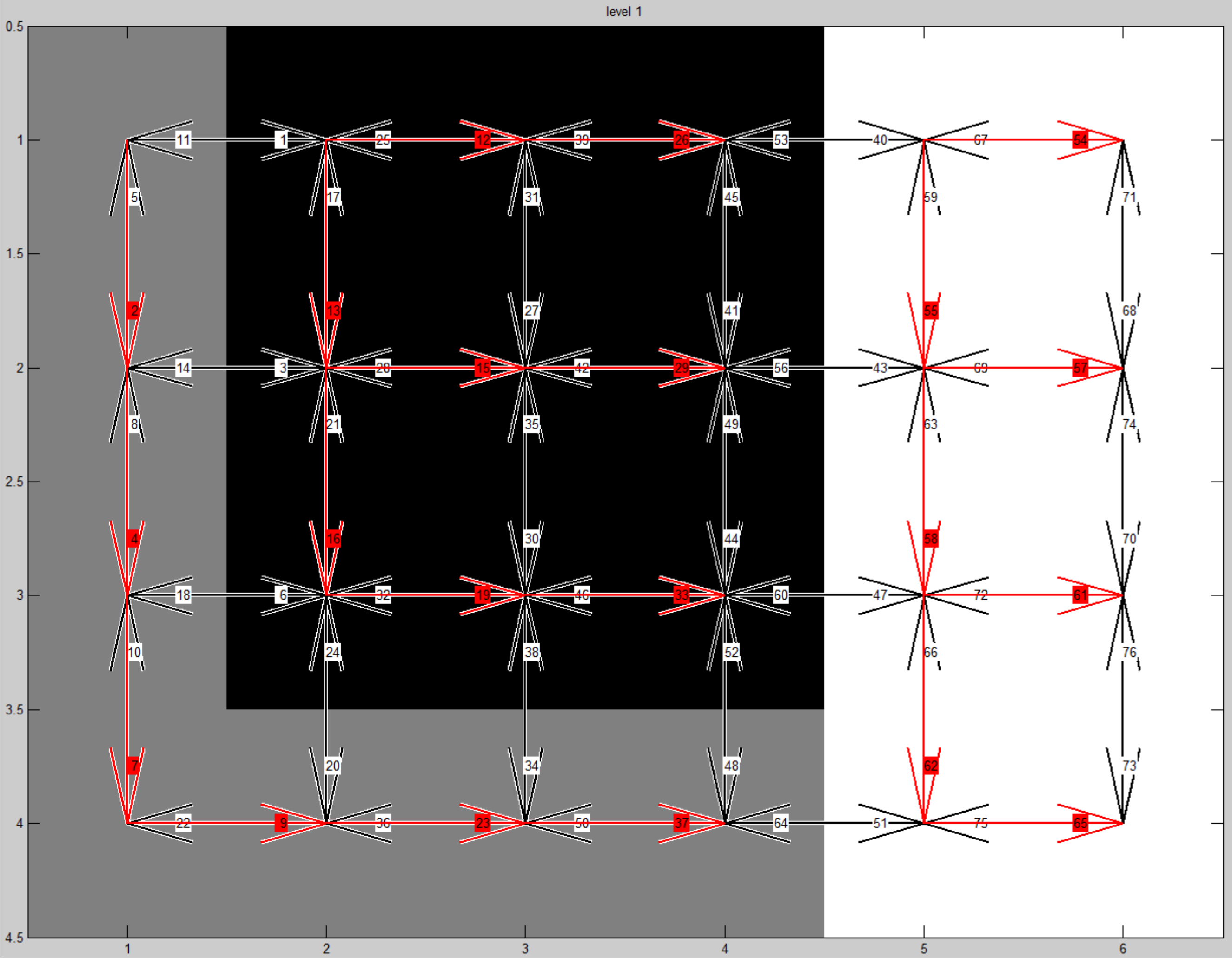
- Examples: Checkerboard





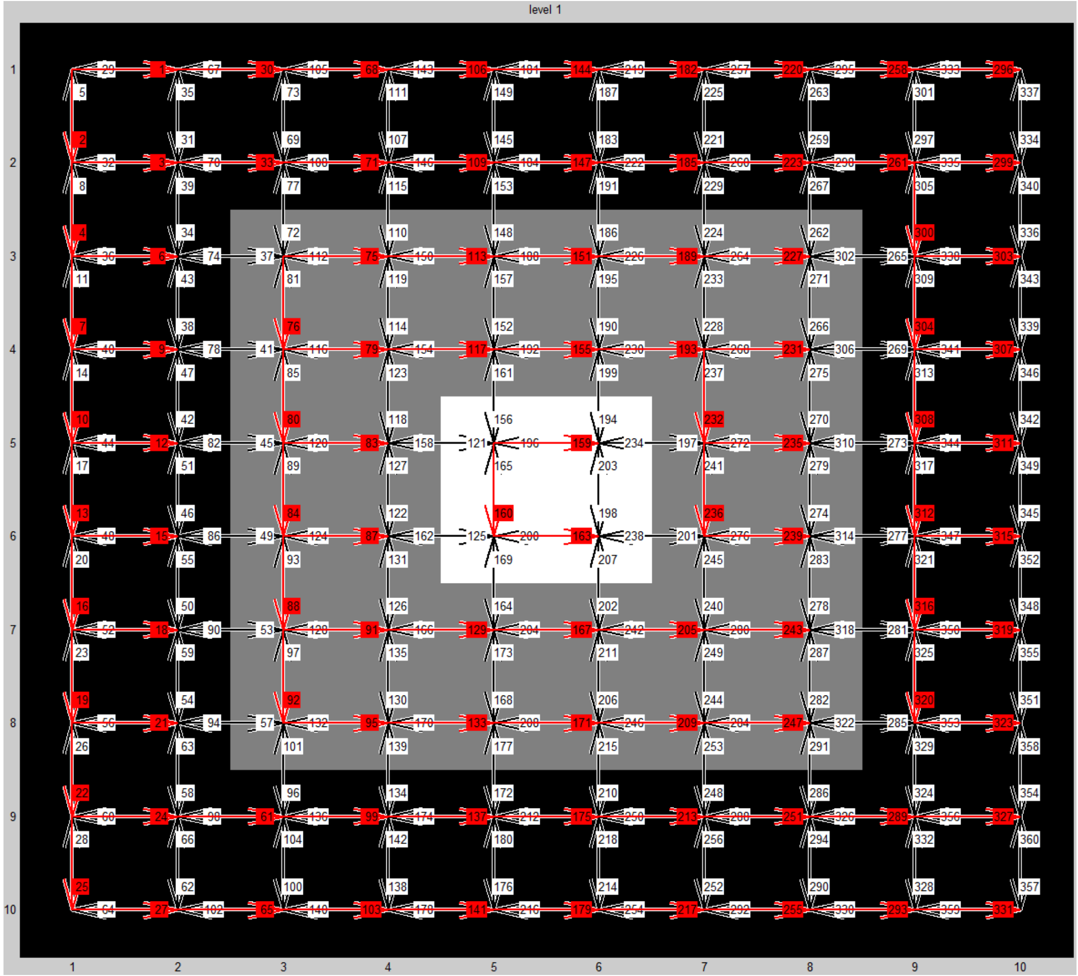
Compute the Contraction Kernel

- Examples: Lecture

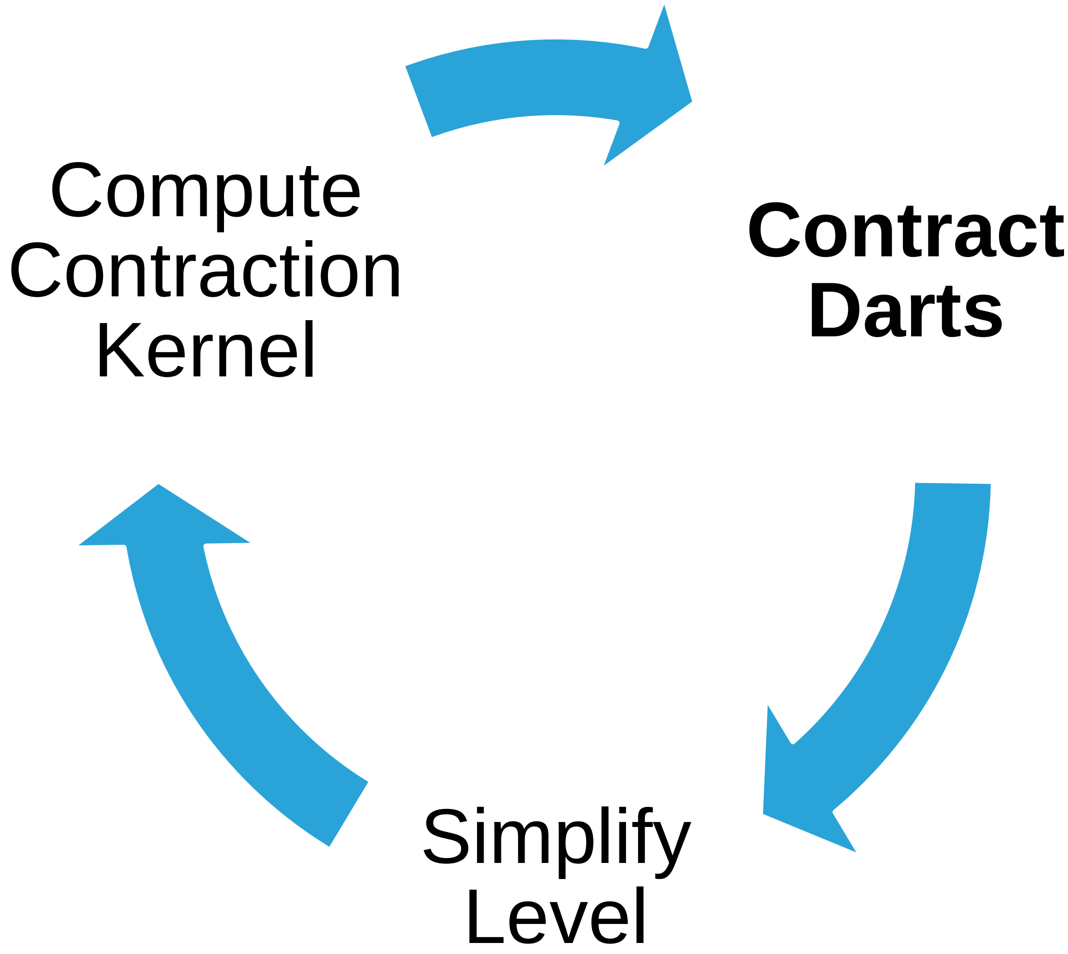


Compute the Contraction Kernel

- Examples: Enclosure

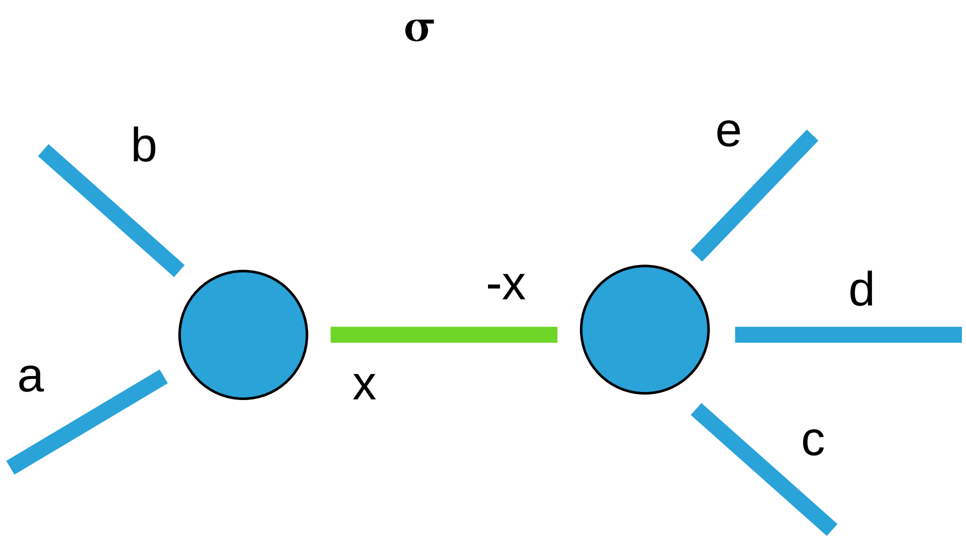


CP Creation

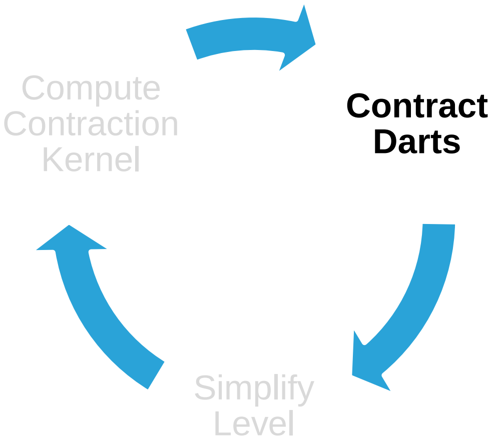

Compute Contraction Kernel
Contract Darts
Simplify Level

CP Creation

- Recap: Contract Darts

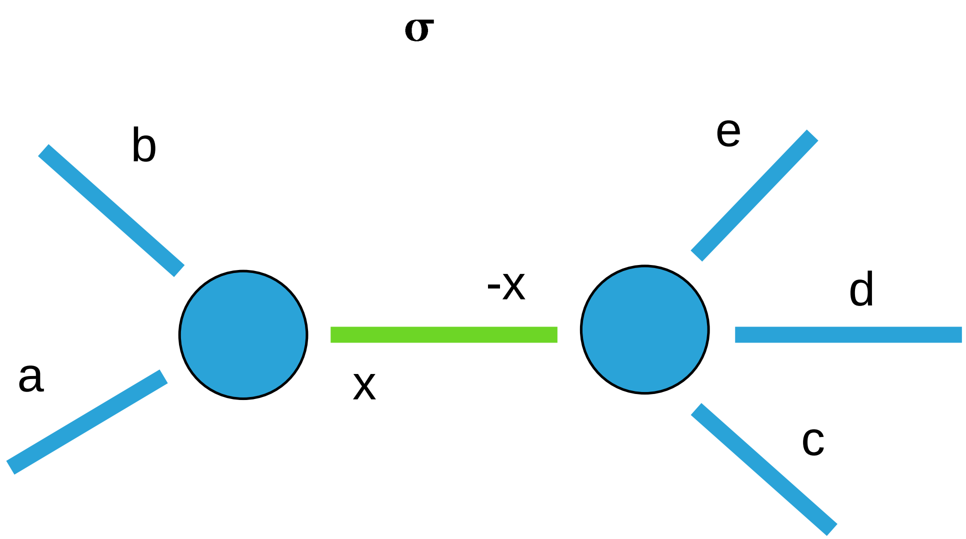


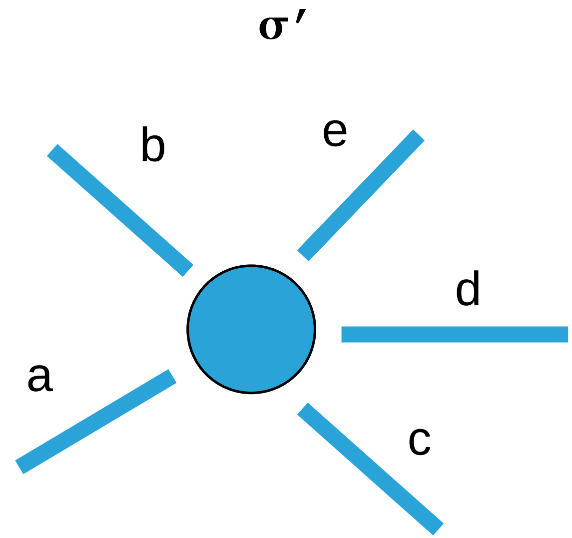
darts	a	b	c	d	e	x	-x
σ	x	a	d	e	-x	b	c
σ'	c	a	d	e	b	b	c



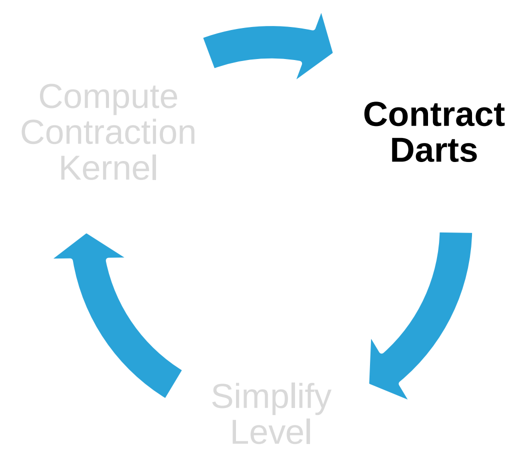
CP Creation

- Recap: Contract Darts





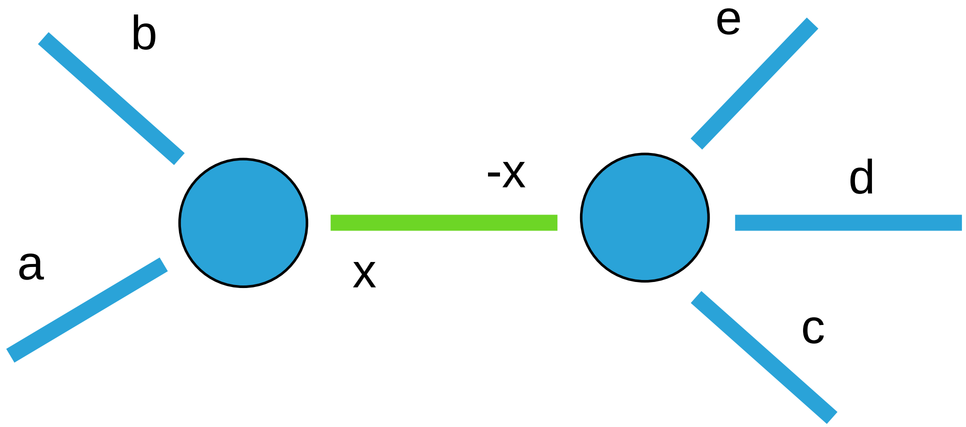
darts	a	b	c	d	e	x	-x
σ	x	a	d	e	-x	b	c
σ'	c	a	d	e	b	b	c



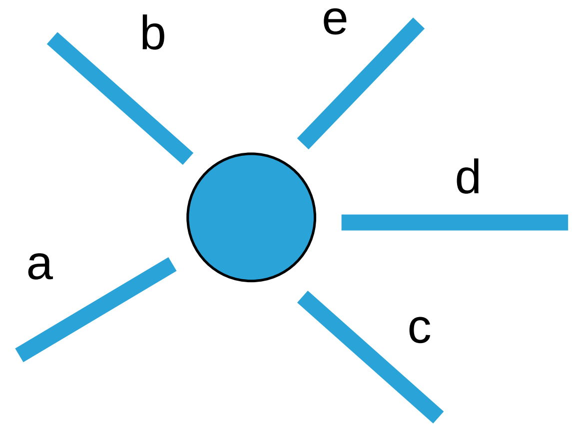
CP Creation

- Recap: Contract Darts

$\boldsymbol{\sigma}$

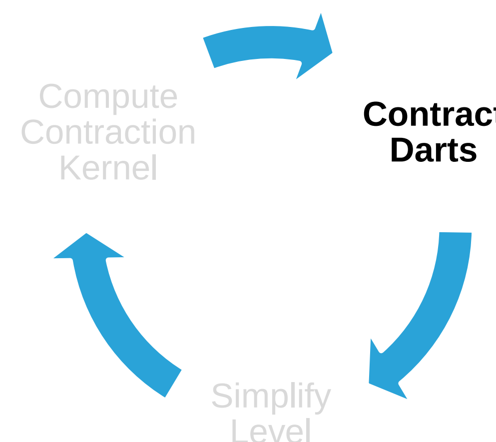


$\boldsymbol{\sigma'}$

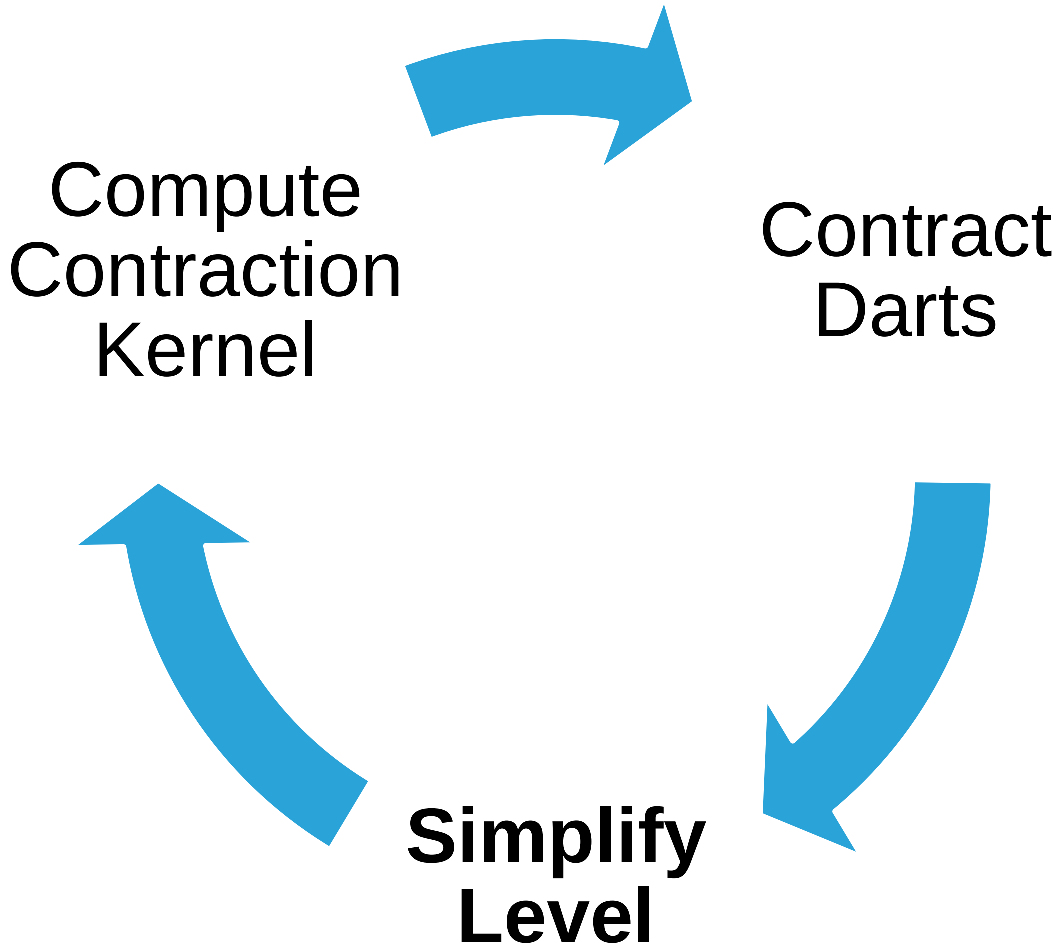


darts	a	b	c	d	e	x	-x
σ	x	a	d	e	-x	b	c
σ'	c	a	d	e	b	b	c

$$\sigma'\left(\sigma^{-1}(x)\right) \coloneqq \sigma(-x)$$
$$\sigma'\left(\sigma^{-1}(-x)\right) \coloneqq \sigma(x)$$

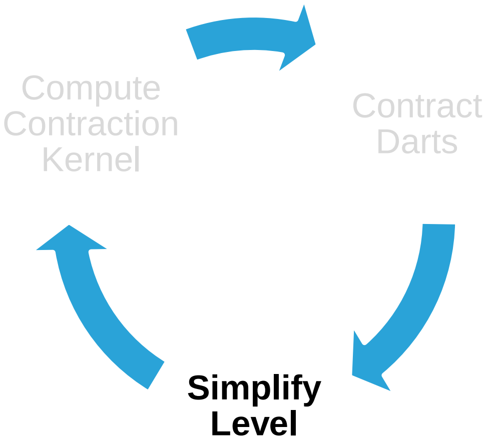


CP Creation


Compute Contraction Kernel
Contract Darts
Simplify Level

CP Creation

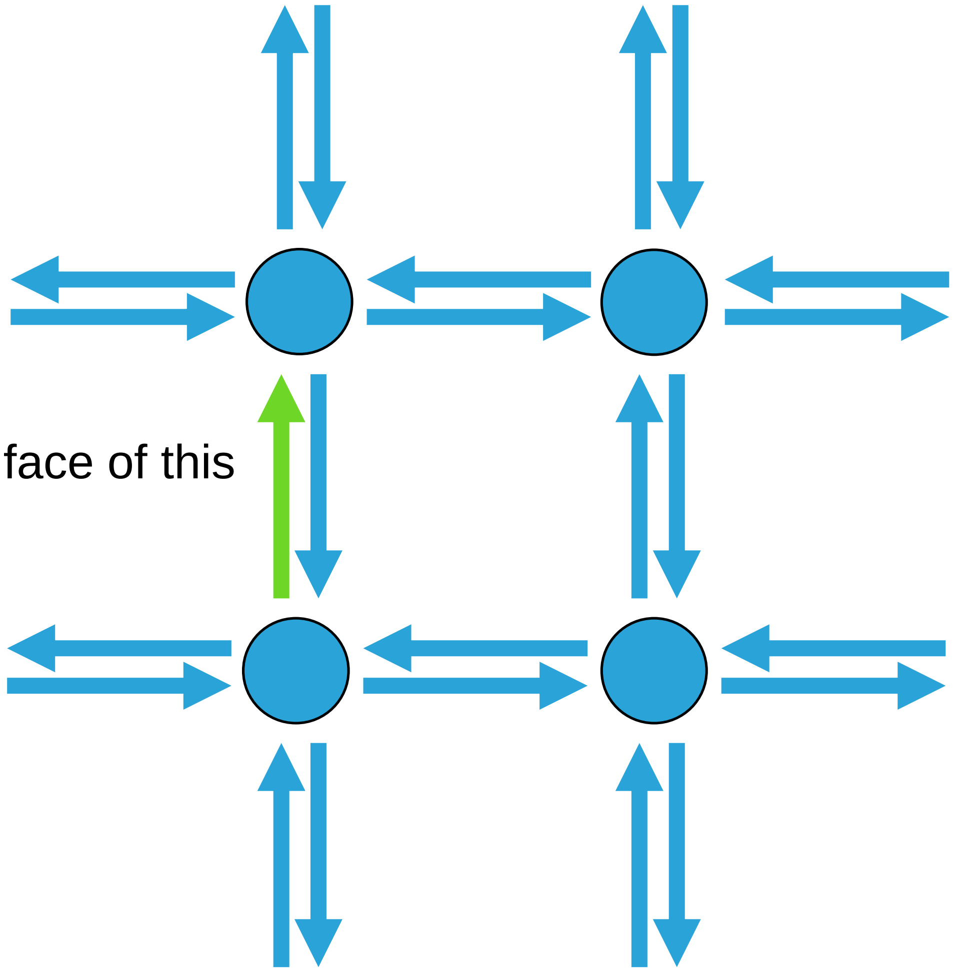
- When contracting darts double edges and self-direct-loops are created. By removing these darts the pyramid is easier to read and faster to compute.

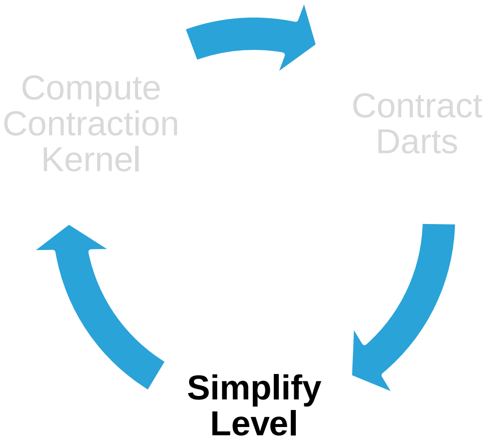


CP Creation

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

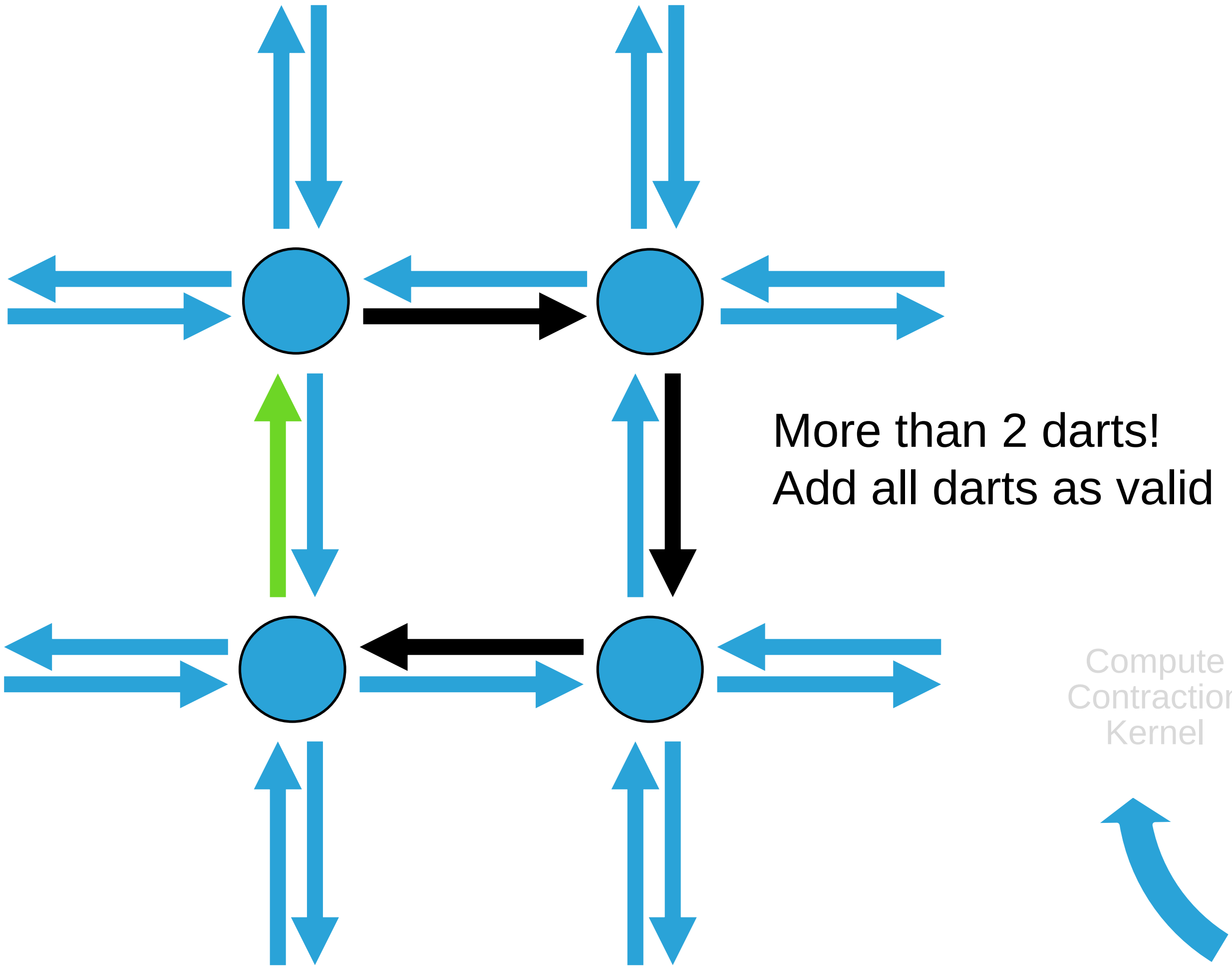
Get face of this dart

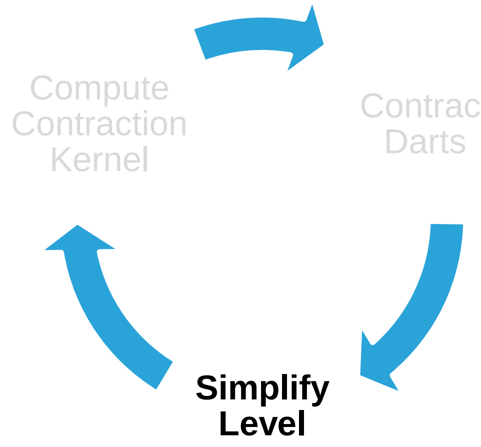




CP Creation

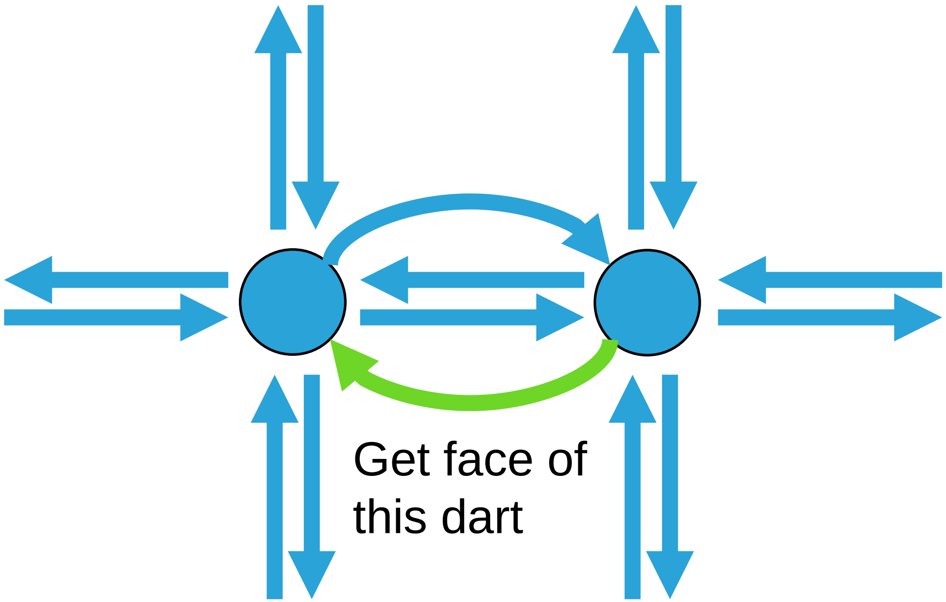
- Check if the **face of a dart** has less than 2 darts:

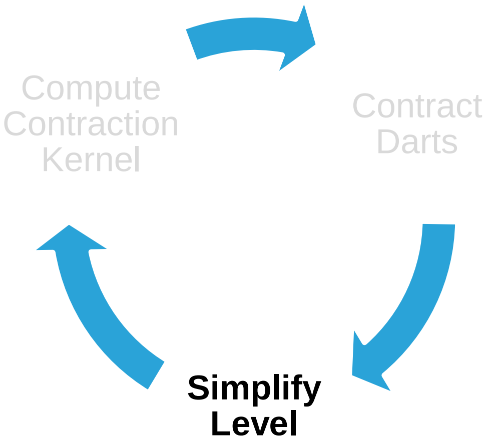




CP Creation

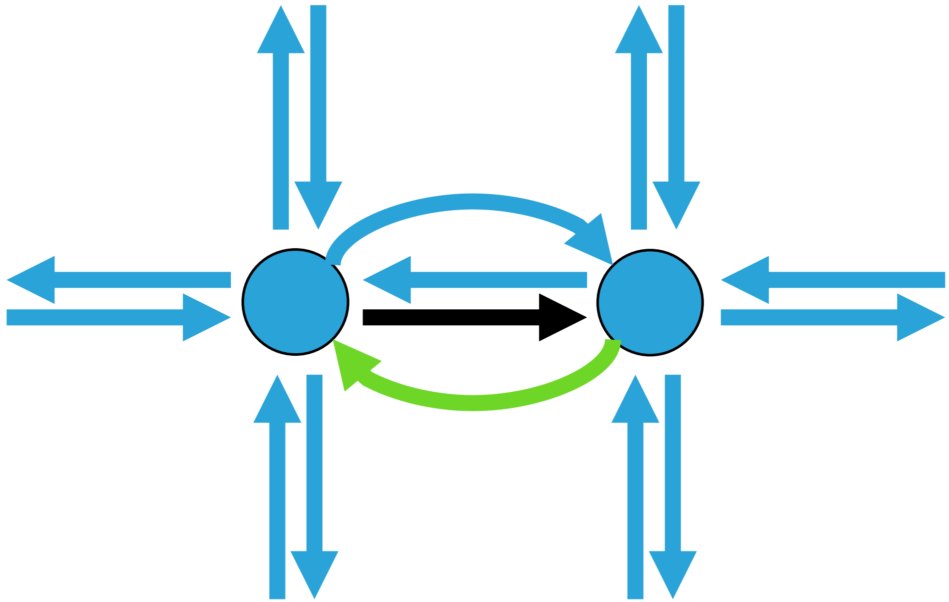
- Check if the **face of a dart** has less than 2 darts:



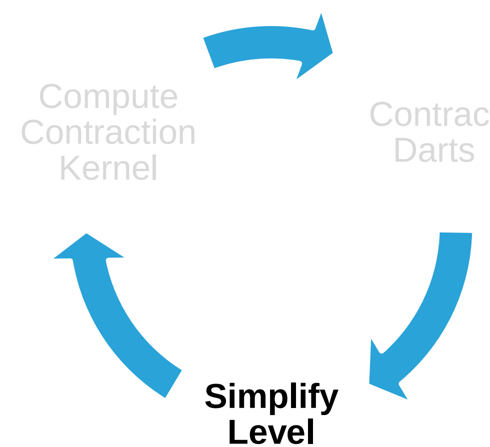


CP Creation

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

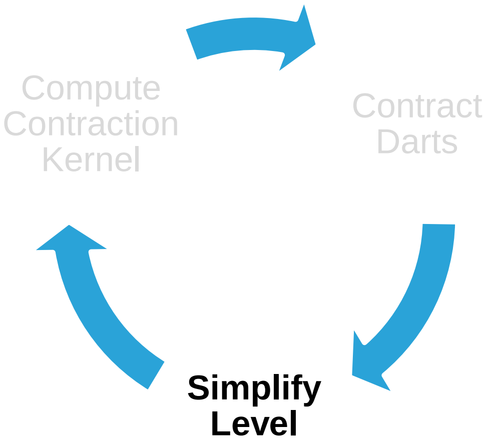


Number of darts in face is 2
→ can be removed!



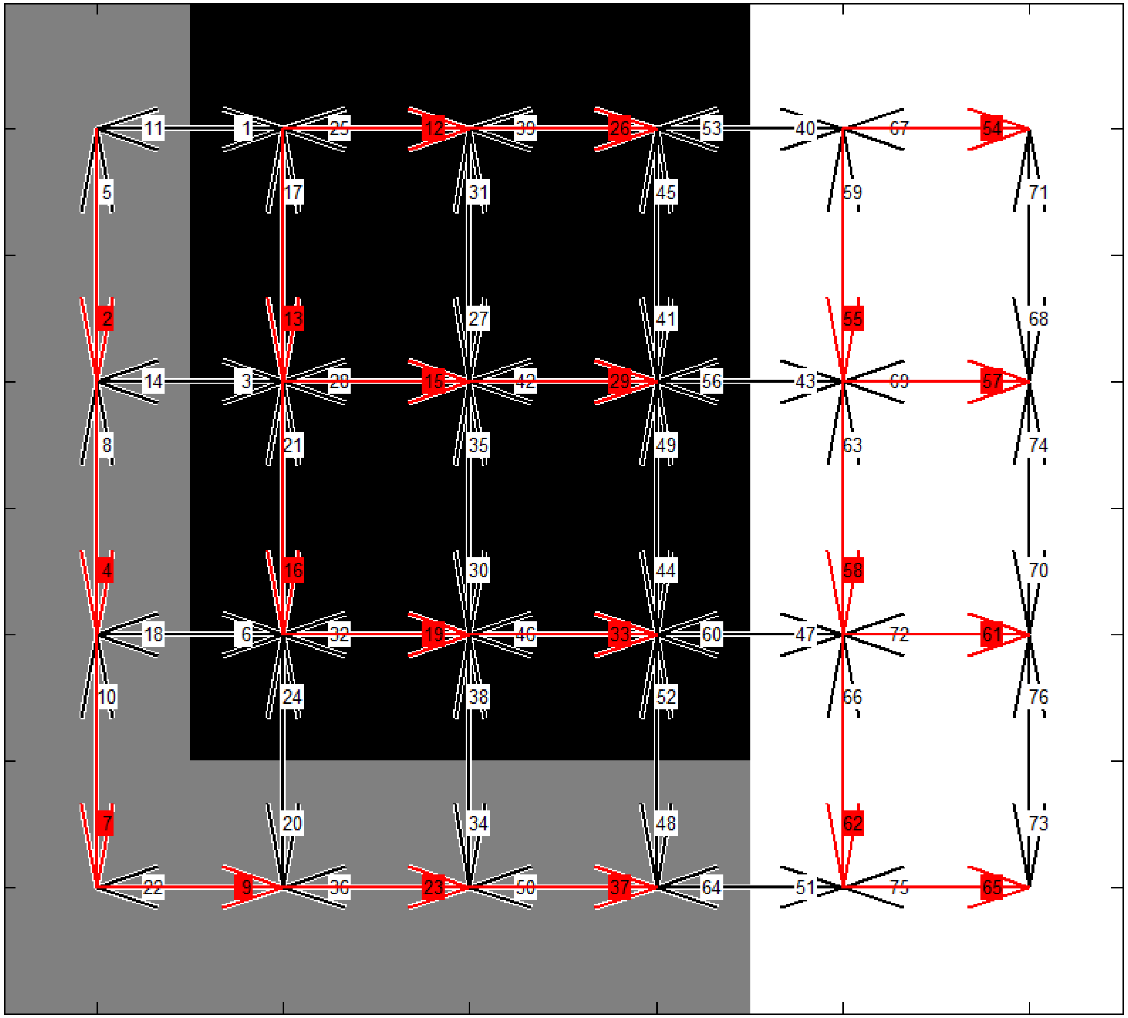
CP Creation

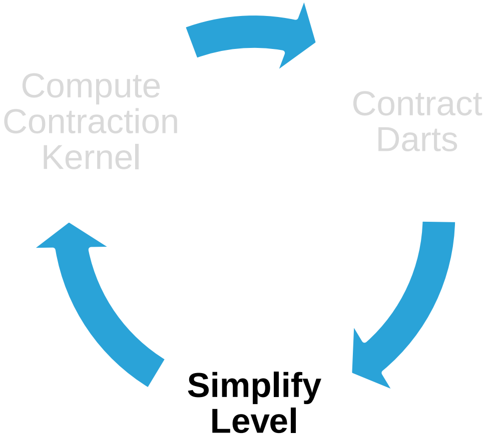
- Special cases for removal:
  - Self-Direct-Loops (2 cases)



CP Creation

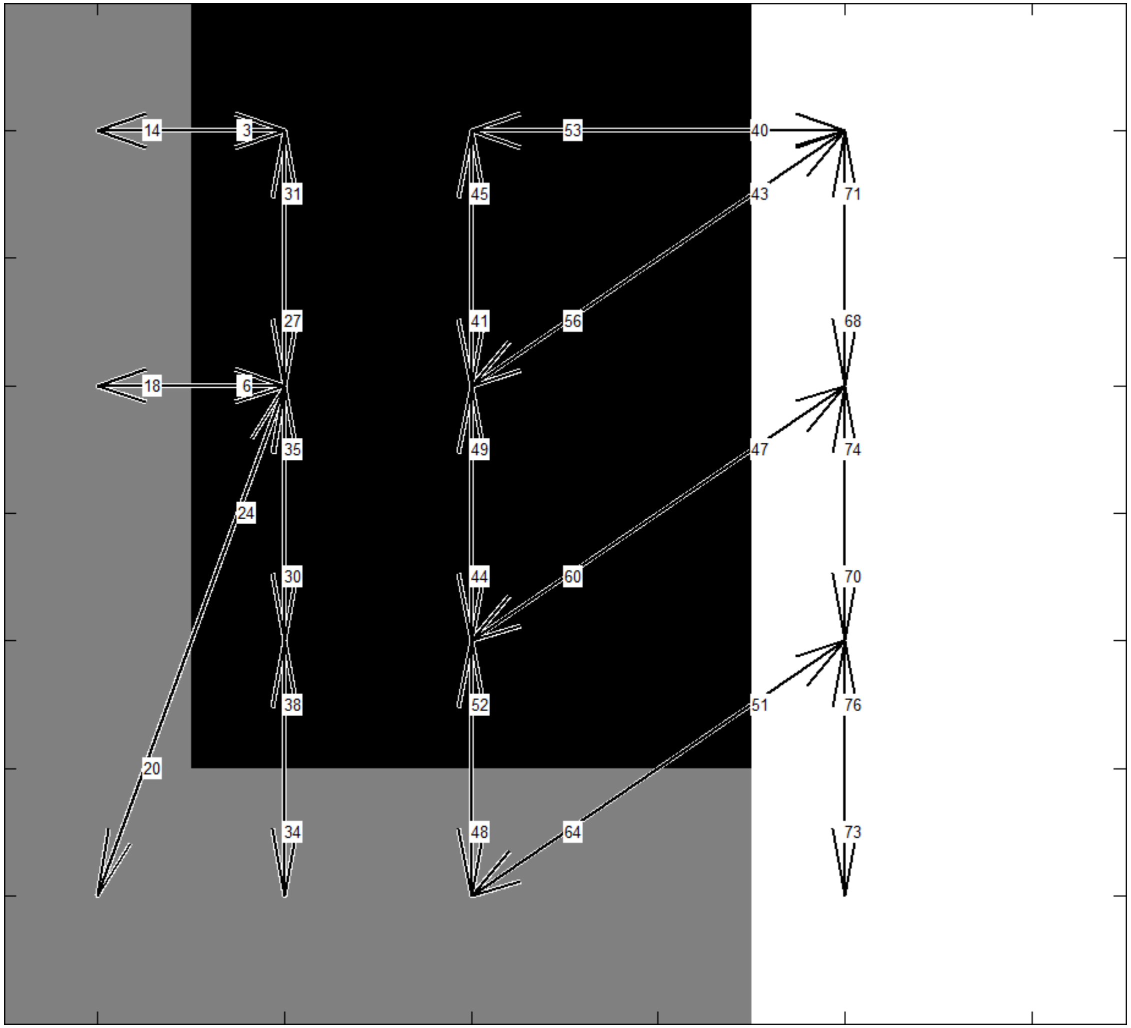
- Example: contraction kernel (before contraction)

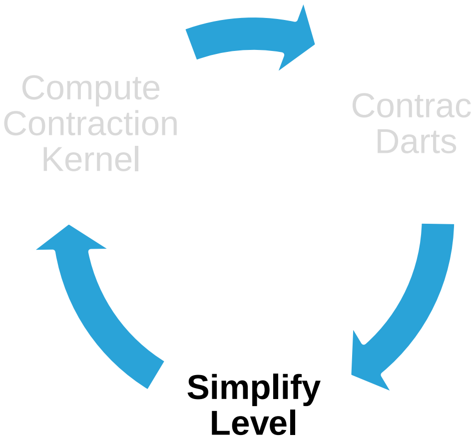




CP Creation

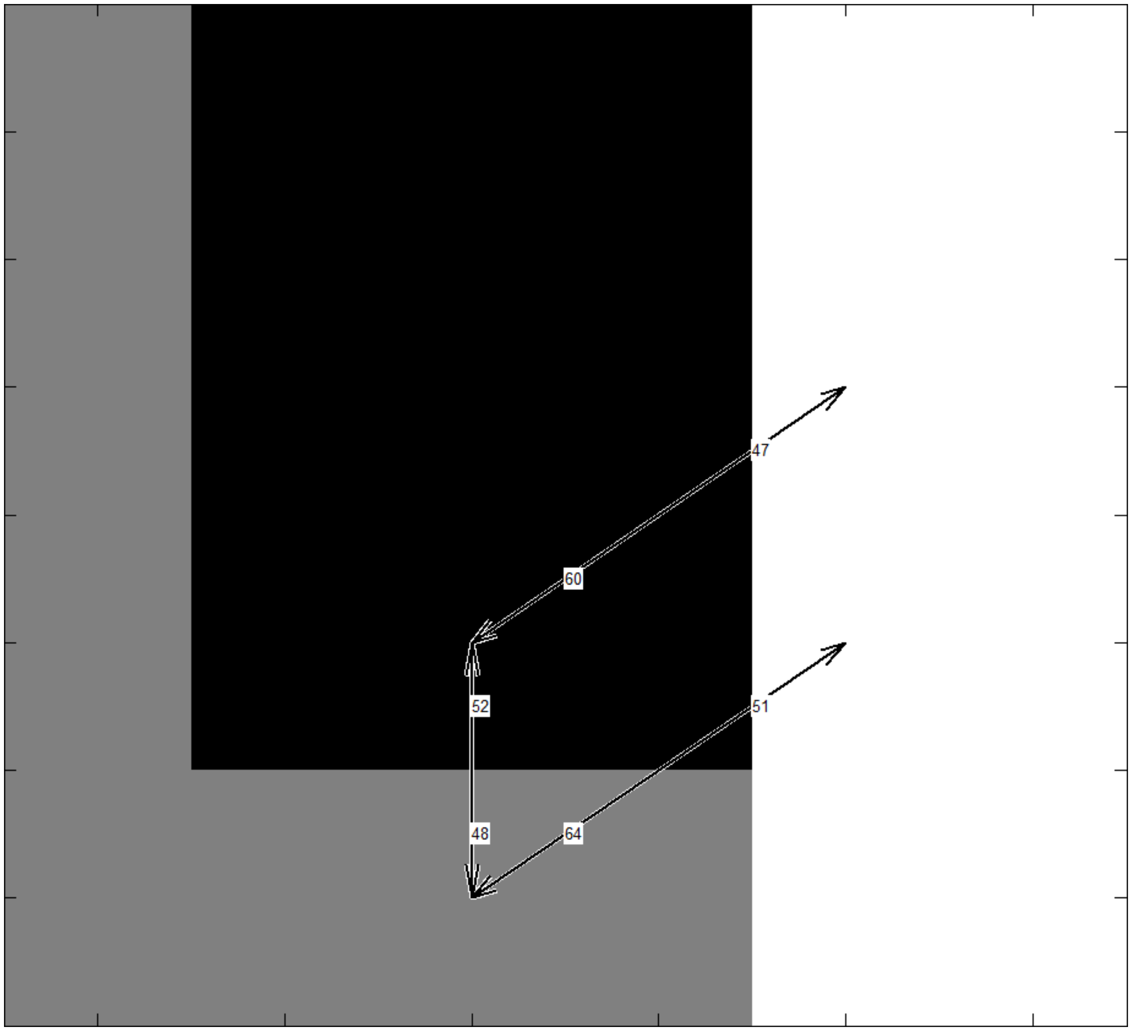
- Example: after contraction

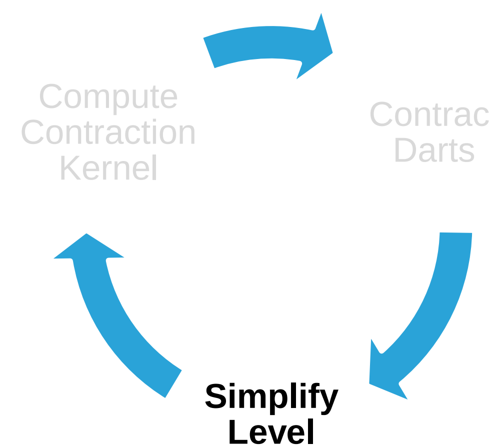




CP Creation

- Example: after simplification





Development Phases

Phase 1:
Existing Work

Phase 2:
Meet the Maps

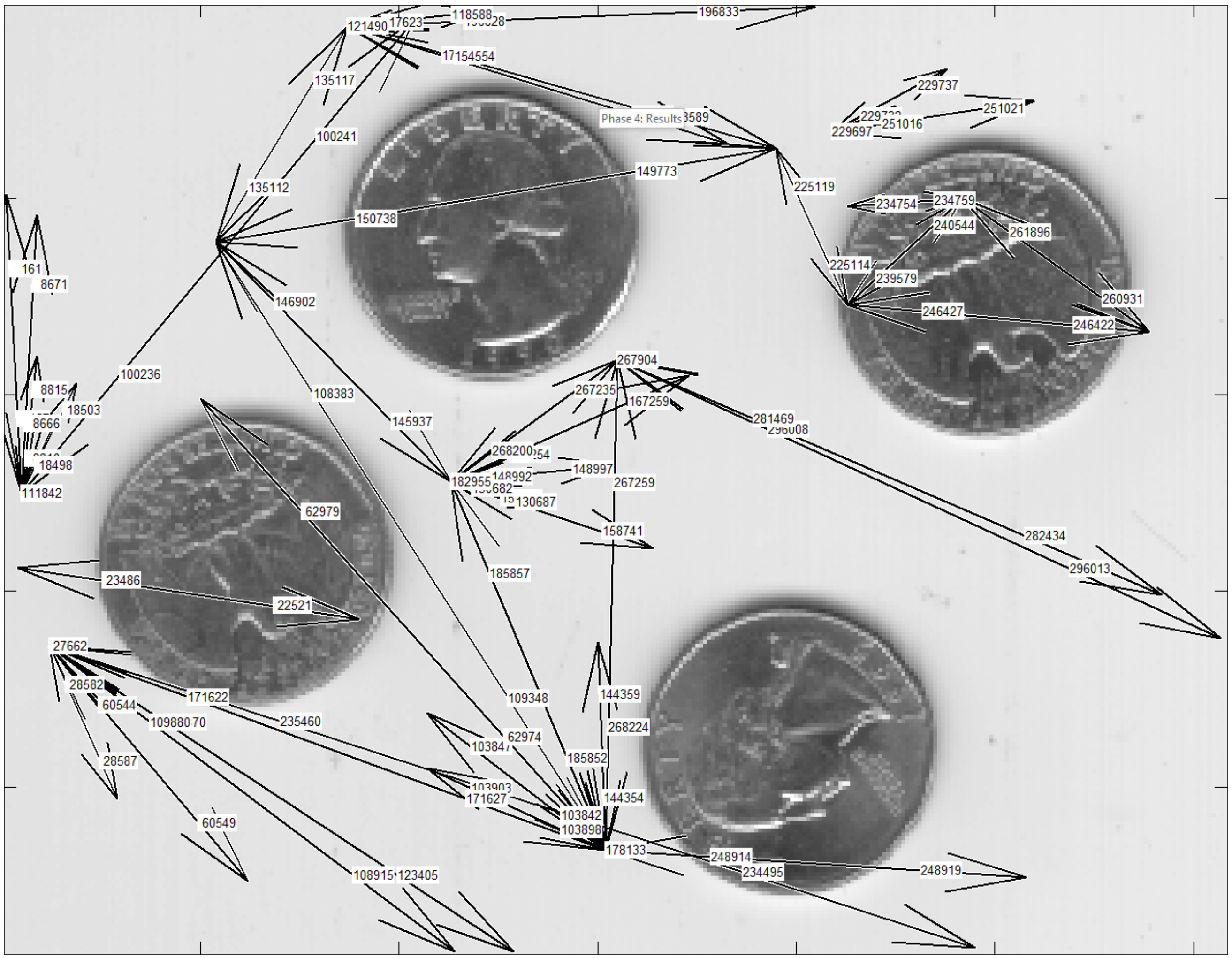
Phase 3:
Implementation

Phase 4:
Results

Phase 4: Results

- DEMO

- Some costly examples:



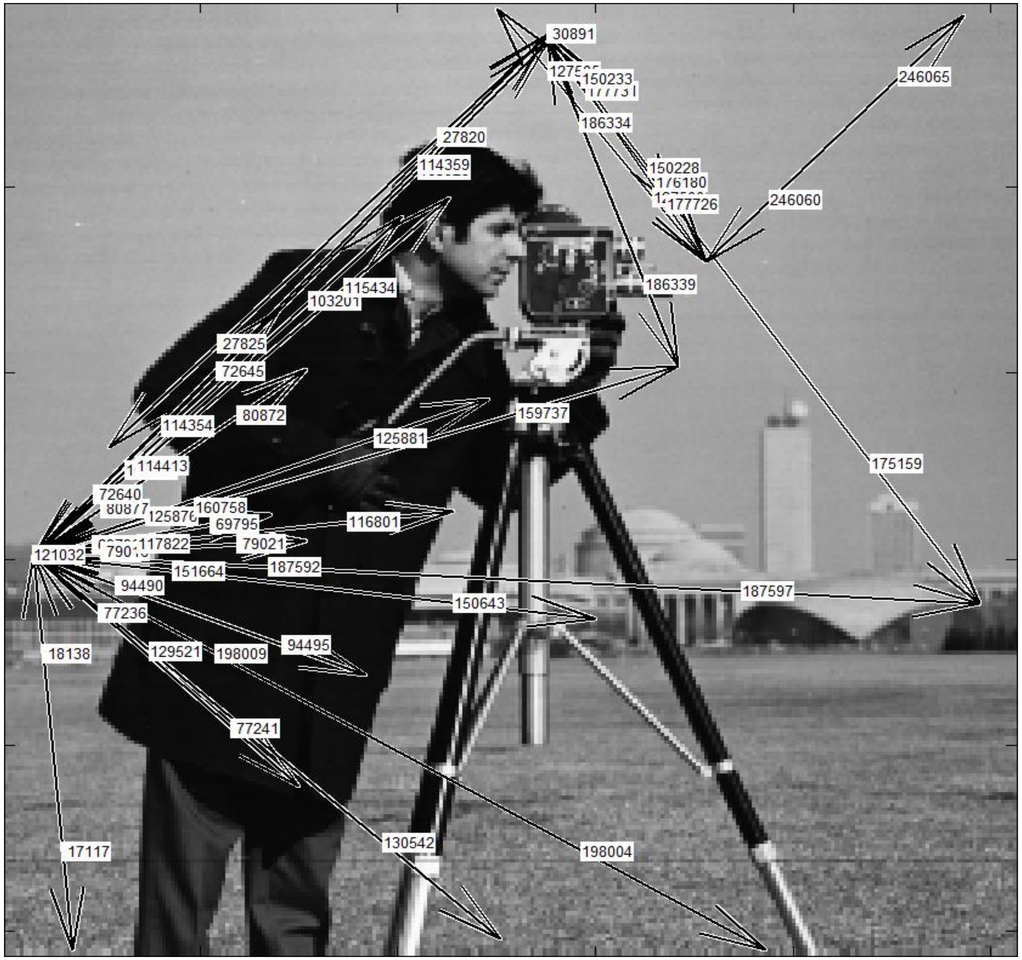
Phase 4: Results

- Some costly examples:

```
Computed dart values in t = 0.0081839
Build the first level in t = 0.034072

Pyramid level 1:
Computing contraction darts in t = 3.1576
Contracting darts in t = 1.5676
Simplify darts in t = 99.6362

Pyramid level 2:
Computing contraction darts in t = 0.52598
Contracting darts in t = 0.22664
Simplify darts in t = 7.4213

Pyramid level 3:
Computing contraction darts in t = 0.14502
Contracting darts in t = 0.052272
Simplify darts in t = 0.90517

Pyramid level 4:
Computing contraction darts in t = 0.04065
Contracting darts in t = 0.013697
Simplify darts in t = 0.13381

Pyramid level 5:
Computing contraction darts in t = 0.020479
Contracting darts in t = 0.0055467
Simplify darts in t = 0.029576

Pyramid level 6:
Computing contraction darts in t = 0.0089889
Contracting darts in t = 0.0038467
Simplify darts in t = 0.010096

Pyramid level 7:
Computing contraction darts in t = 0.03025
Contracting darts in t = 0.0028141
Simplify darts in t = 0.0046334

Pyramid level 8:
Computing contraction darts in t = 0.021293
Contracting darts in t = 0.00013621
Simplify darts in t = 0.0022108
```

Phase 4: Results

- Some costly examples:



Phase 4: Results

- Some costly examples:

```
Computed dart values in t = 0.0040965
Build the first level in t = 0.028408

Pyramid level 1:
Computing contraction darts in t = 2.3352
Contracting darts in t = 1.0566
Simplify darts in t = 73.4445

Pyramid level 2:
Computing contraction darts in t = 0.66482
Contracting darts in t = 0.29912
Simplify darts in t = 12.5519

Pyramid level 3:
Computing contraction darts in t = 0.17884
Contracting darts in t = 0.077297
Simplify darts in t = 1.668

Pyramid level 4:
Computing contraction darts in t = 0.054448
Contracting darts in t = 0.021359
Simplify darts in t = 0.24825

Pyramid level 5:
Computing contraction darts in t = 0.016426
Contracting darts in t = 0.0073146
Simplify darts in t = 0.04384

Pyramid level 6:
Computing contraction darts in t = 0.010979
Contracting darts in t = 0.0039908
Simplify darts in t = 0.013091

Pyramid level 7:
Computing contraction darts in t = 0.008718
Contracting darts in t = 0.003313
Simplify darts in t = 0.004469

Pyramid level 8:
Computing contraction darts in t = 0.015805
Contracting darts in t = 0.0001234
Simplify darts in t = 0.0020848
```

Literatur

- [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.