Upper Bound on the Complexity of Solving Renaming

Ami Paz, Technion

Joint work with:

Hagit Attiya, Technion

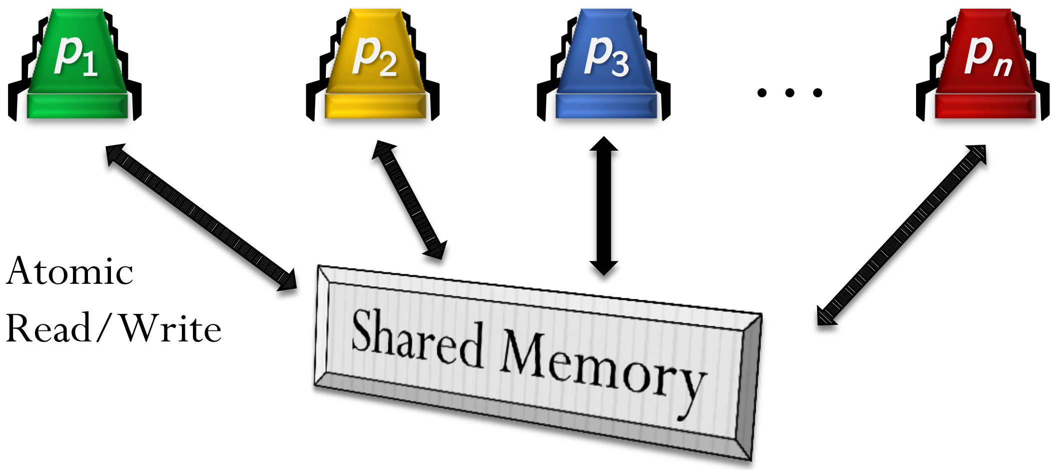
Armando Castañeda, Technion

Maurice Herlihy, Brown

PODC 2013 Best Student Paper Award

Introduction

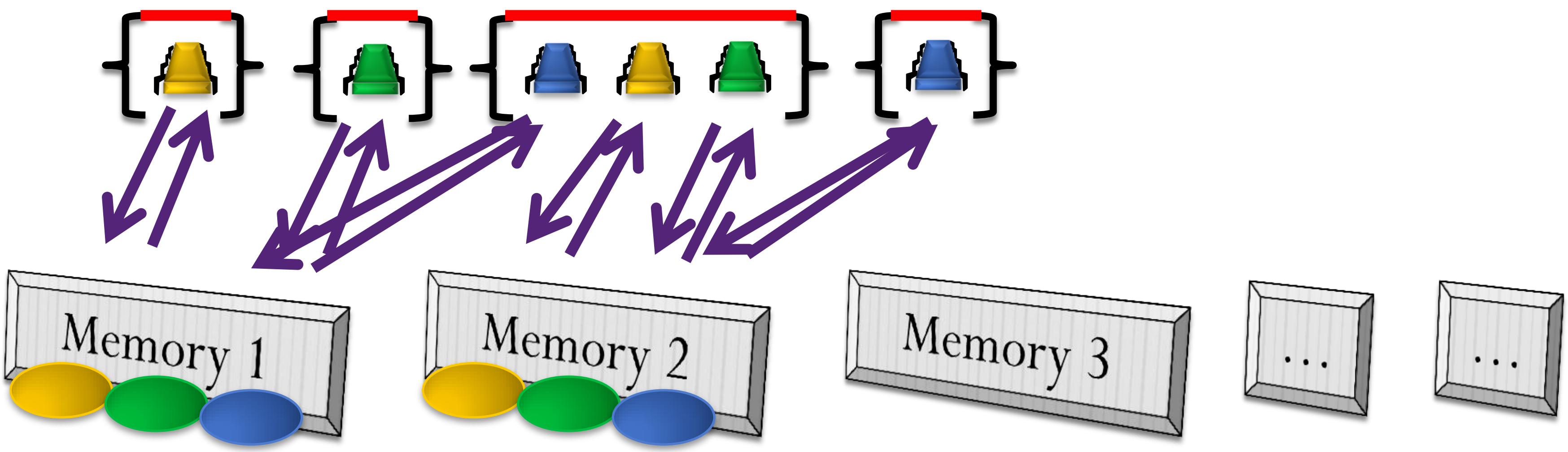
The Model



- *n* asynchronous processes.
- At most *n*–1 processes can crash.
- Wait-free algorithms: each nonfaulty process produces an output.
- Full information.

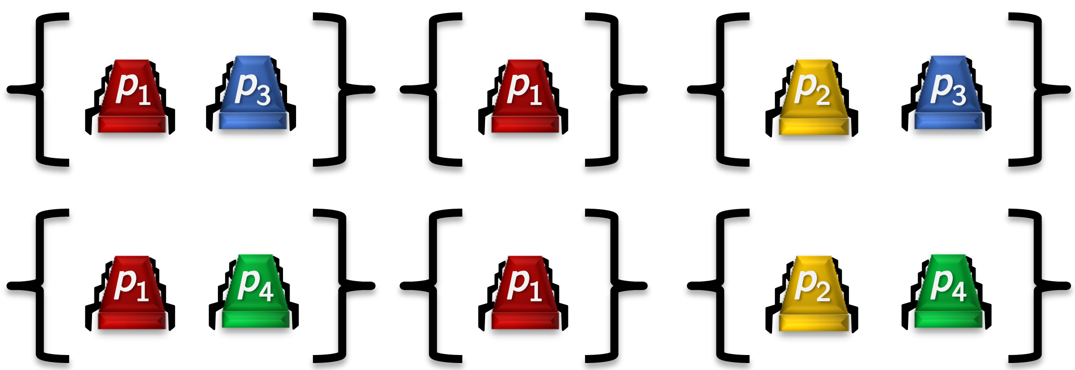
Iterated Atomic Snapshot

- Execution induced by a sequence of blocks:
  - Write together;
  - Read together.
- Fresh copy of the memory every time.
- Implemented in $O(n^2)$ overhead [Borowsky and Gafni 97].



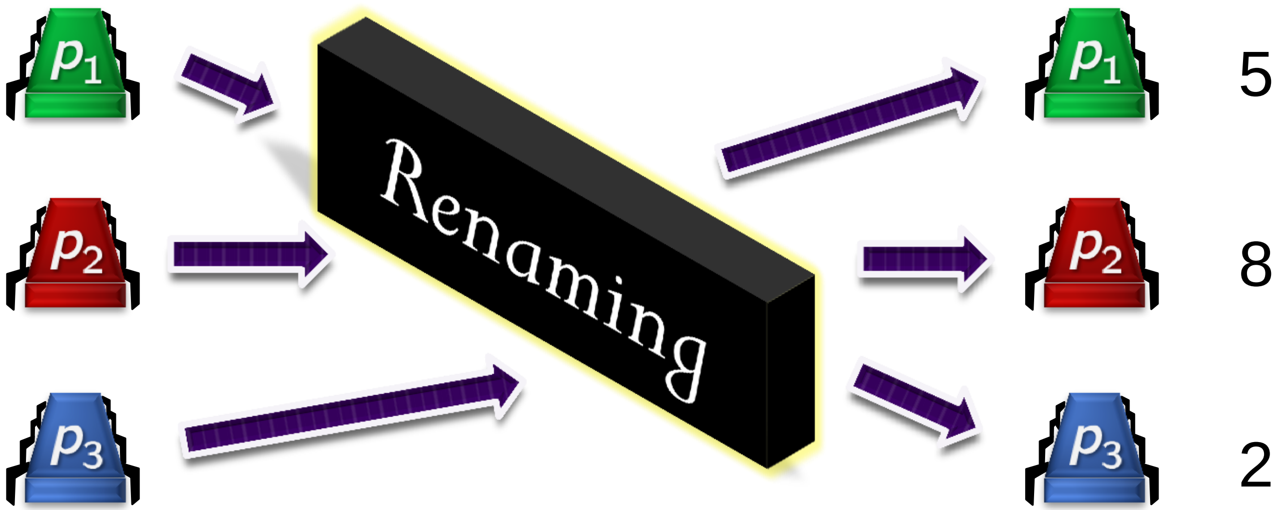
Comparison Based Algorithms

- Processes only compare their identifiers.
- Execution by P_1, P_2, P_3 looks like execution by P_1, P_2, P_4 .



M-Renaming

[Attiya et al. 90]



n processes

With identifiers

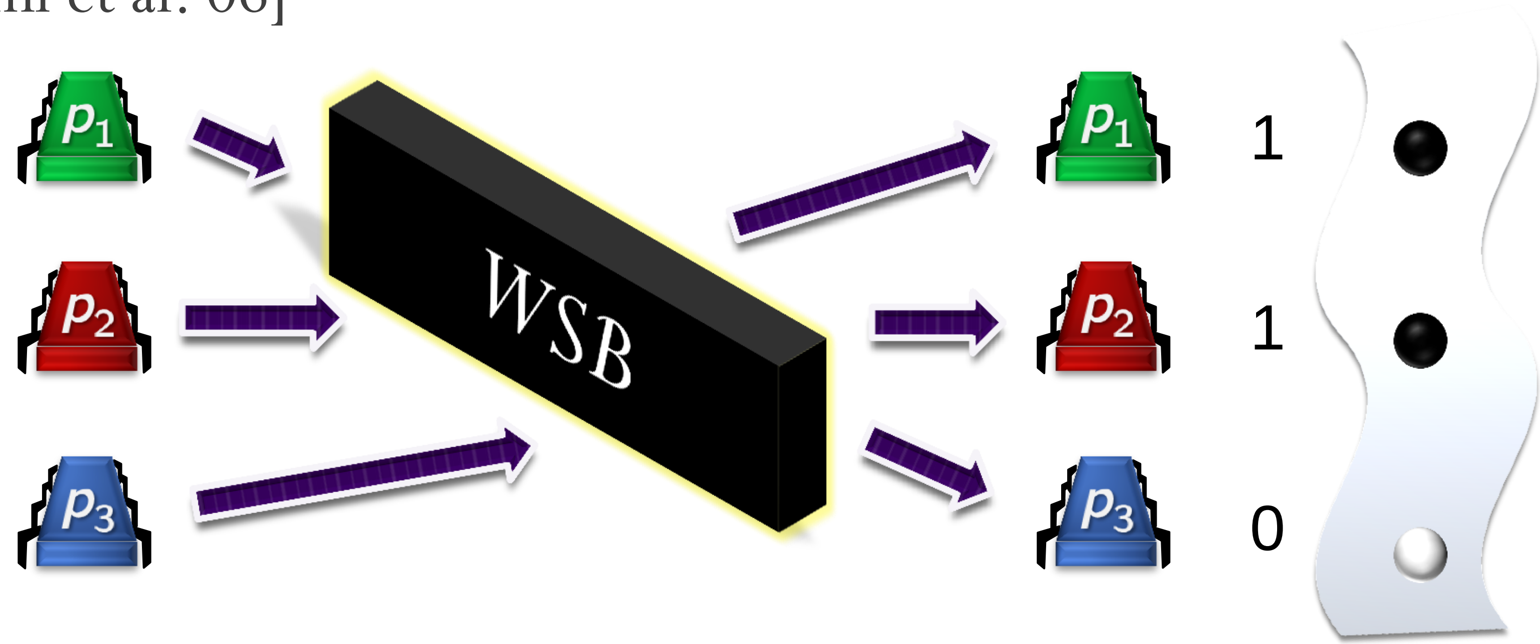
Outputs: $1, \ldots, M$

Unique values

Processes are only allowed to compare their identifiers

Weak Symmetry Breaking (WSB)

[Gafni et al. 06]



n processes

With identifiers

Outputs: 0/1

If all output: not all the same

Equivalent to $(2n\text{-}2)$-renaming

M-Renaming Bounds

M:	$1,\ldots,n$	$n+1,\ldots,2n-2$	$2n-1,\ldots$
		WSB	
	✗	✗	✓
	[Attiya et al. 90]	Several Papers	[Attiya et al. 90]

M-Renaming Bounds

[Castañeda and Rajsbaum 10]: Lower bounds are wrong.

M: / n	$1,\ldots,n$	$n+1,\ldots$	$2n\text{-}2$ WSB	$2n\text{-}1,\ldots$
Prime Power	✗	✗	✗	✓
Non Prime Power	✗	?	✓	✓

Renaming Bounds

[Castañeda and Rajsbaum 10]: Lower bounds are wrong.

- Existential proof.
- No bounds on steps complexity.

Our Results

- n-process algorithm for WSB and $(2n-2)$-renaming, when n is not a prime power.
- Bounded step complexity: $O(n^{q+5})$, where q is the largest prime power dividing n.

Topology & Distributed Computing

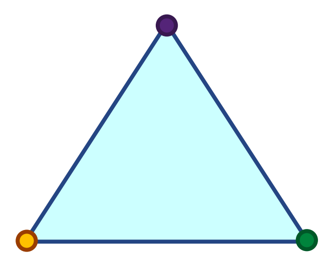
Simplexes

- Sets of objects.
- Represented as convex hulls of points.

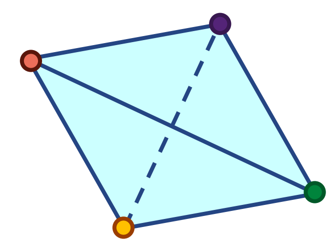
$\{x\}$

$\{x, y\}$

$\{x, y, z\}$

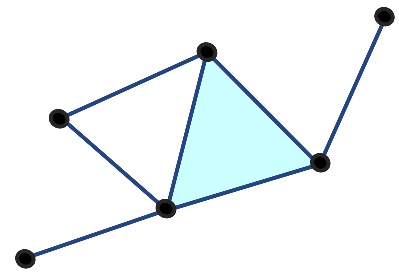
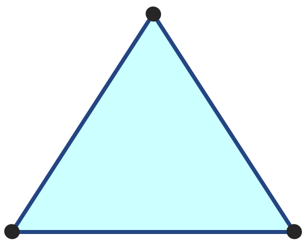


$\{x, y, z, w\}$

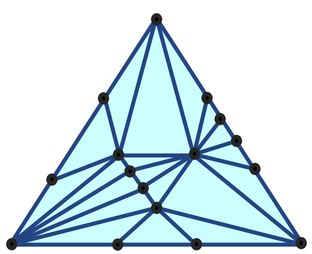


Simplicial Complexes

- "Gluings" of simplexes.

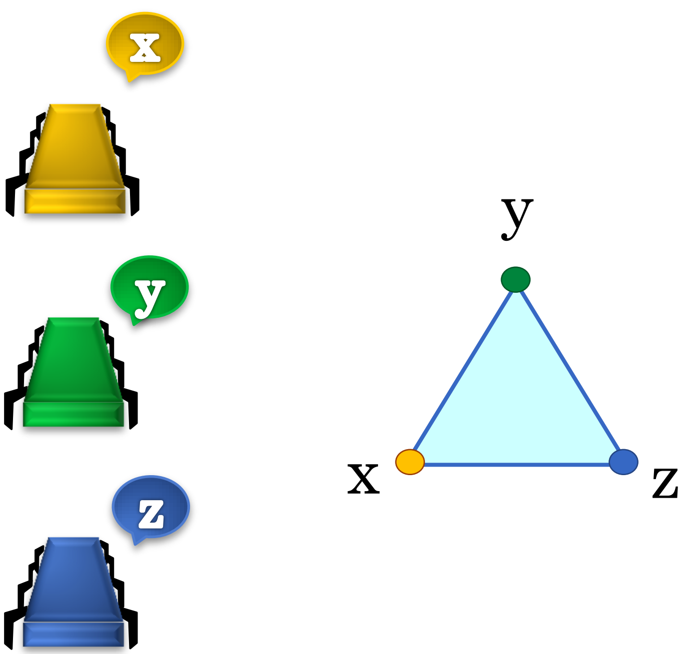
- Some complexes are called subdivisions of others.

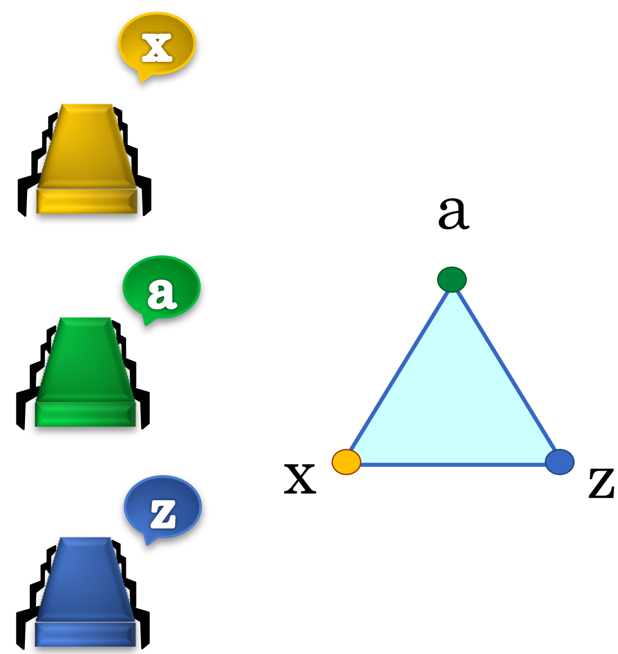



Topology & Distributed Computing

[Borowsky and Gafni 93]; [Herlihy and Shavit 93,99];
[Saks and Zaharoglou 93,00]; [Herlihy and Rajsbaum 94,00].

- Simplicial complexes represent states of the system.

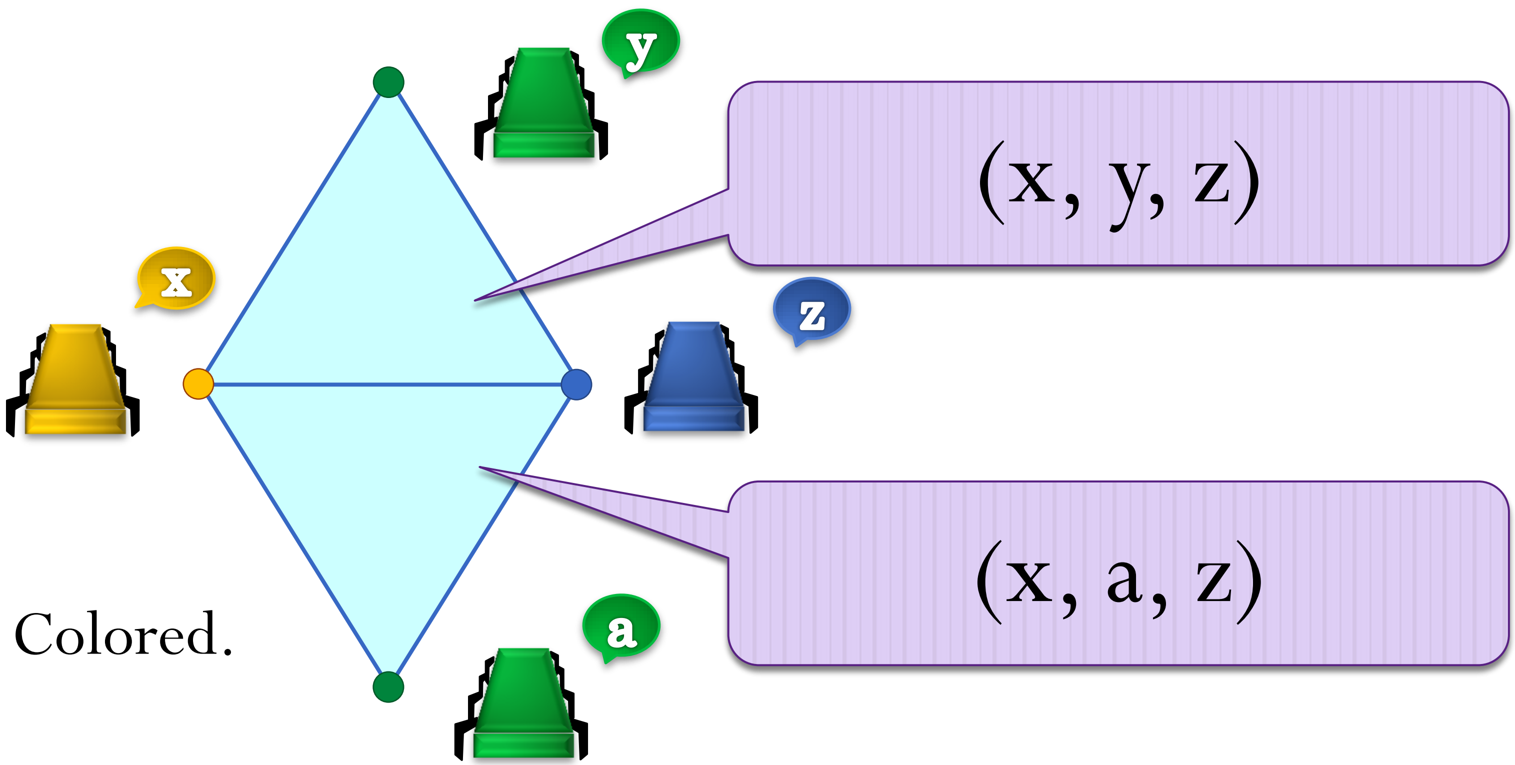




Topology & Distributed Computing

[Borowsky and Gafni 93]; [Herlihy and Shavit 93,99];
[Saks and Zaharoglou 93,00]; [Herlihy and Rajsbaum 94,00].

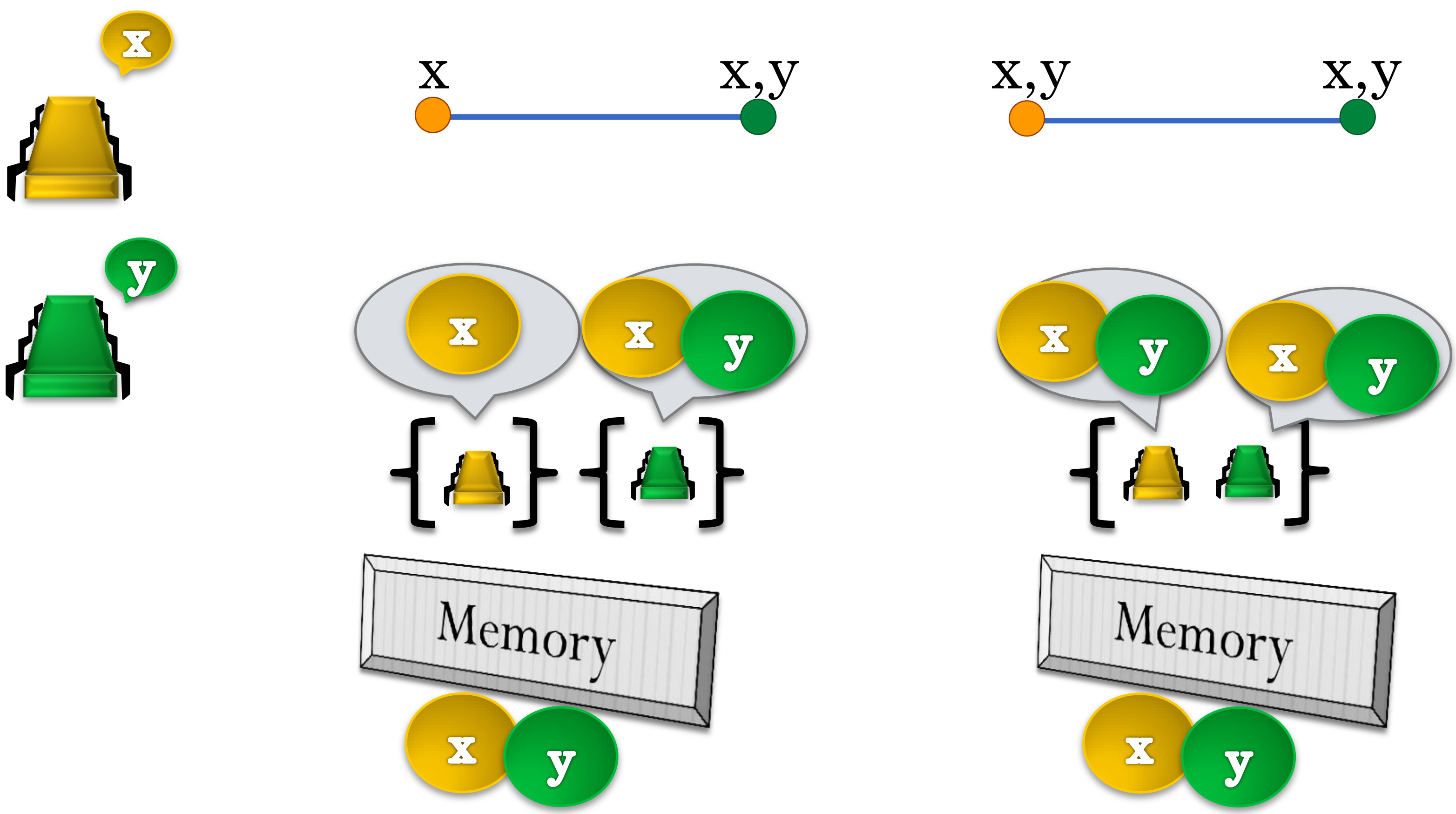
- Simplicial complexes represent states of the system.



- Colored.

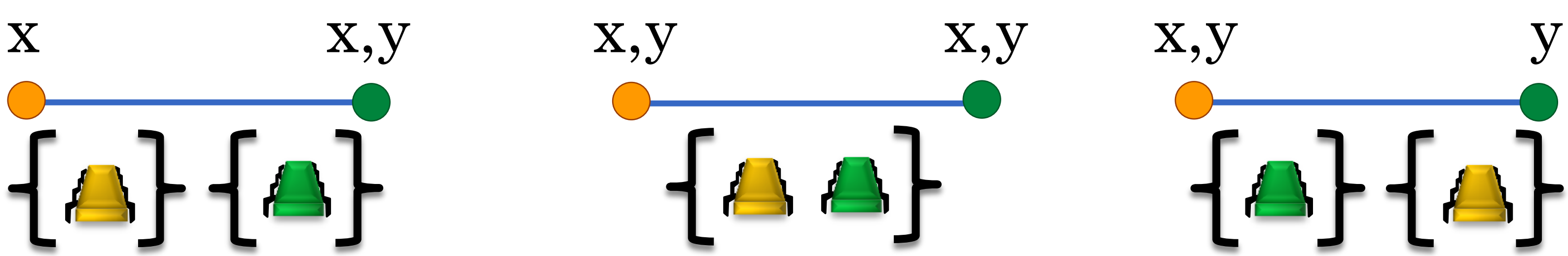
Topology & Distributed Computing

- An execution.

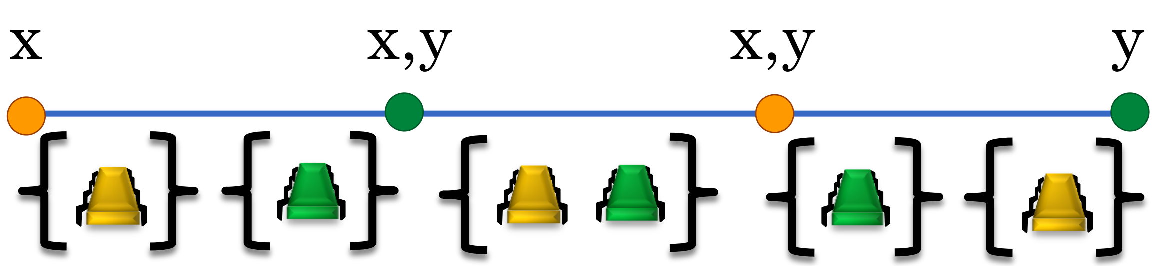


Topology & Distributed Computing

- An execution.

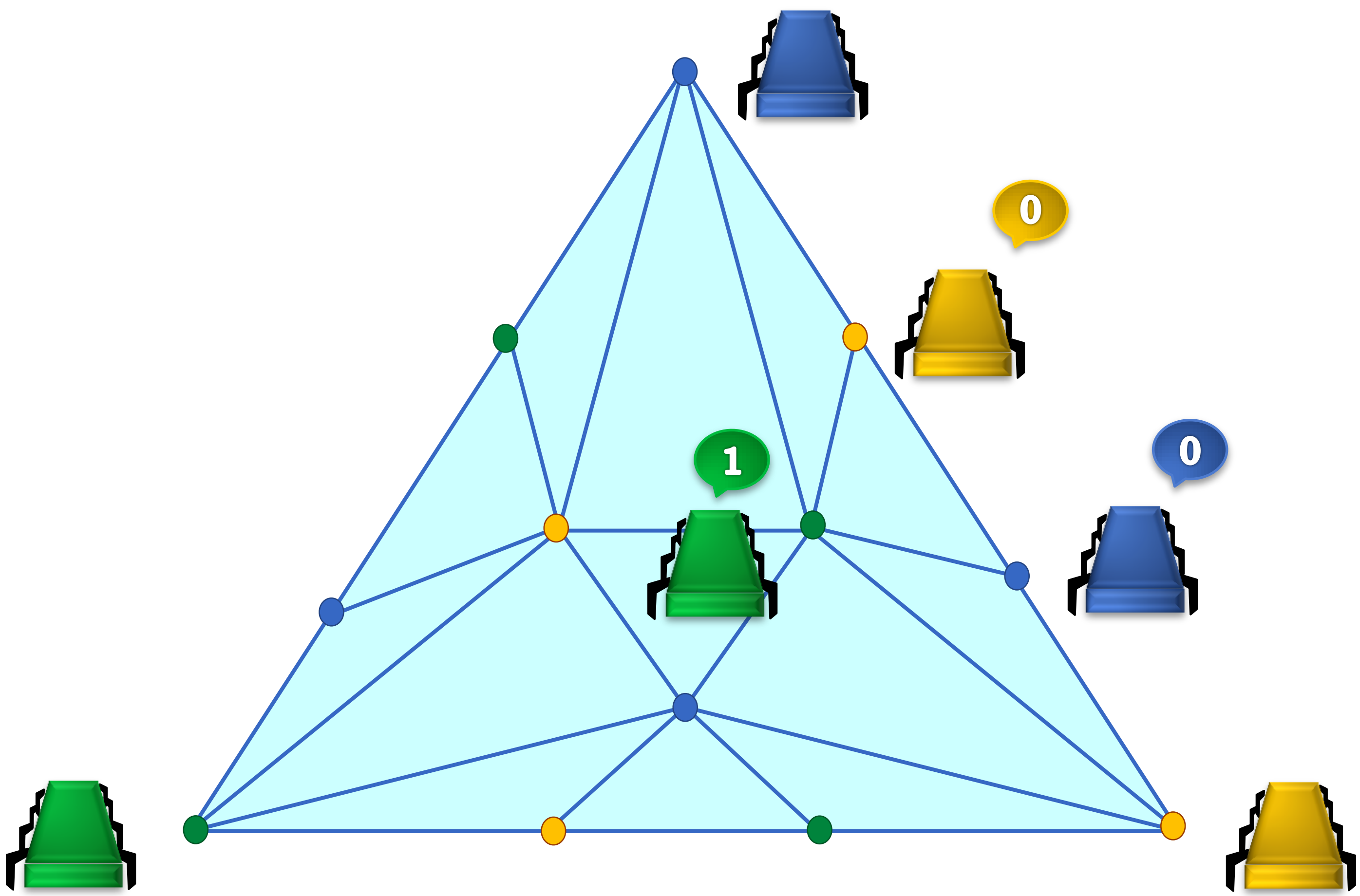


- All 1-step interleaving.

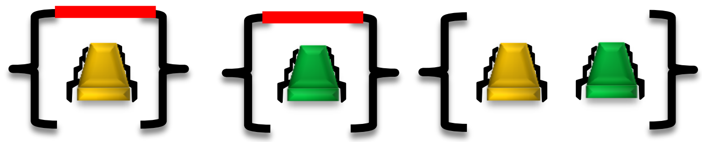


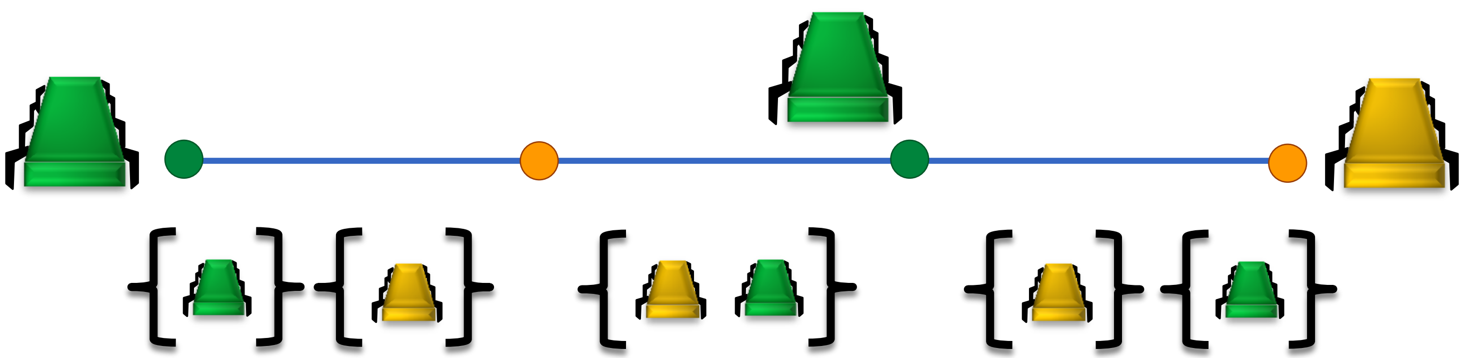
Subdivision Implies Algorithm

- Simplicial approximation: processes converge on a simplex.

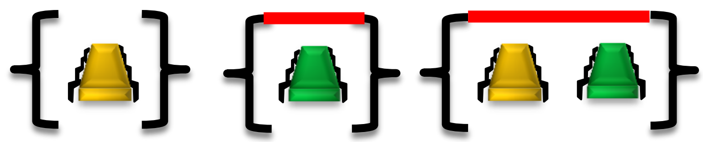


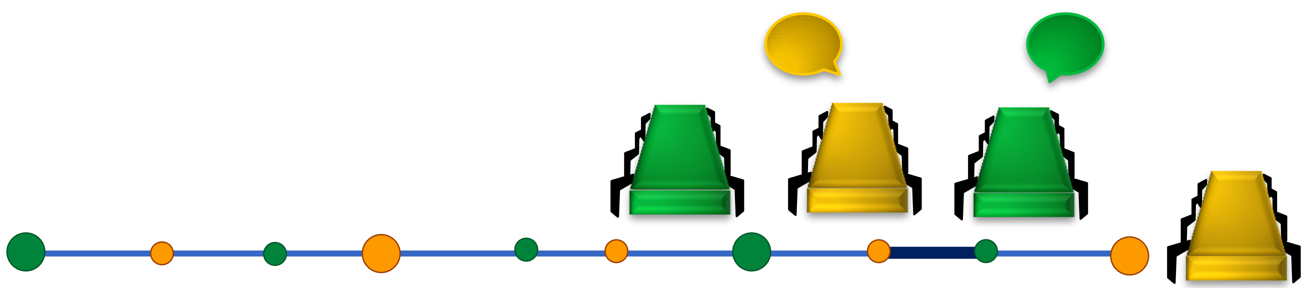
Subdivision Implies Algorithm

- Execution: 

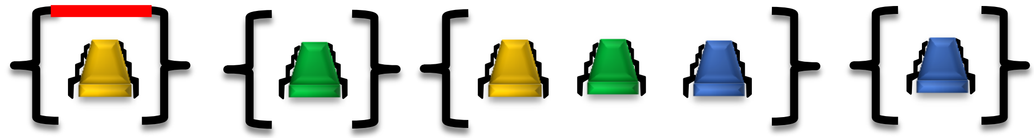


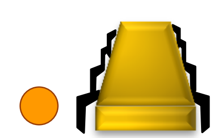
Subdivision Implies Algorithm

- Execution: 

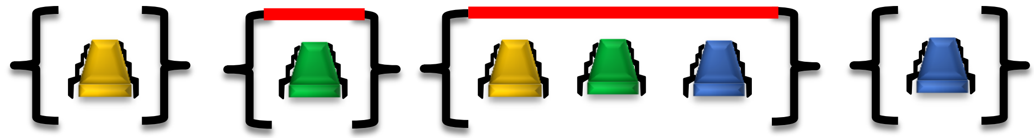


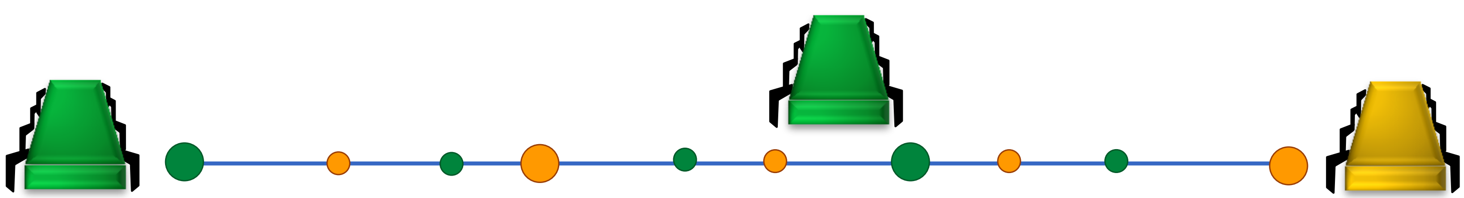
Subdivision Implies Algorithm

- Execution: 



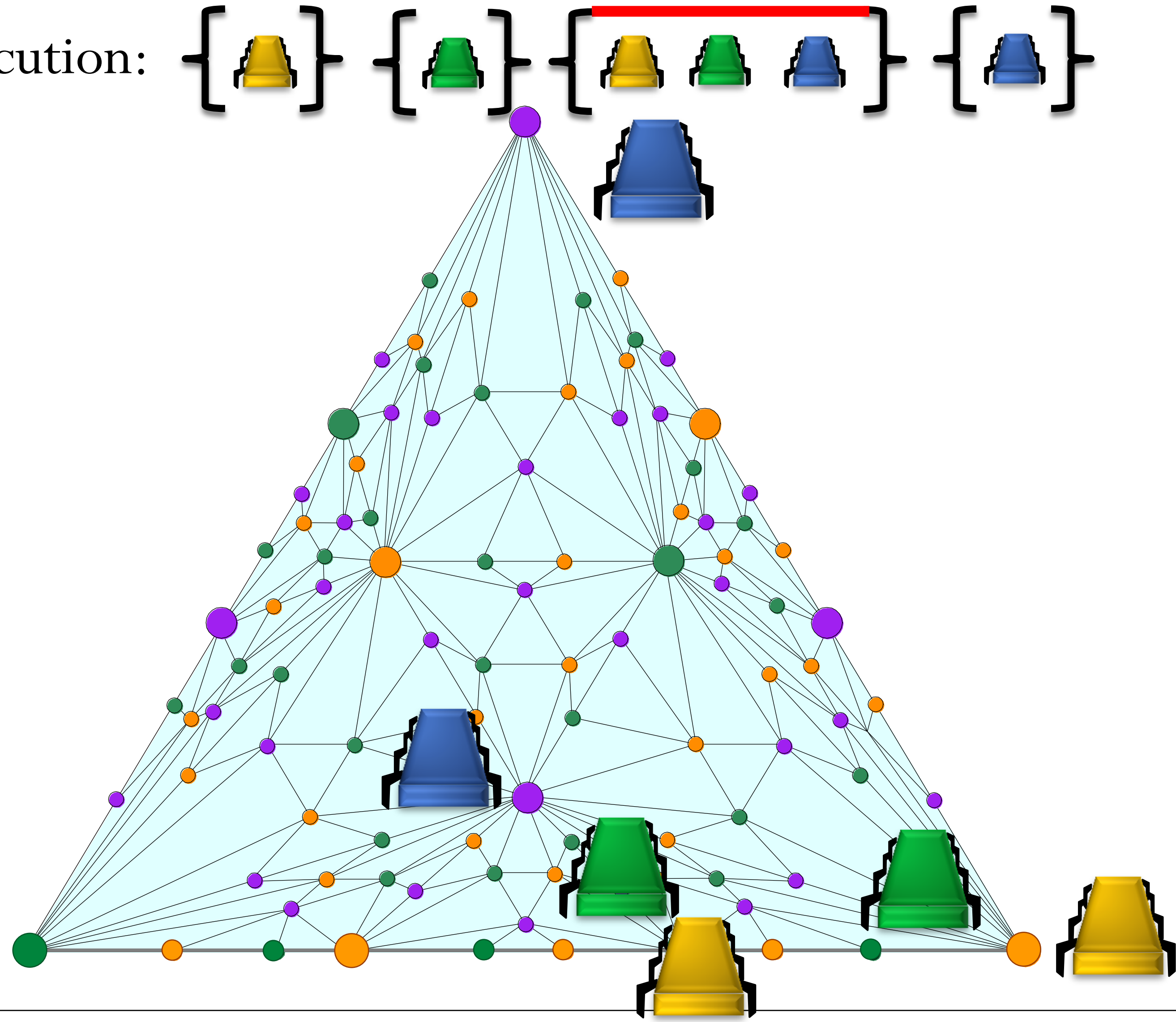
Subdivision Implies Algorithm

- Execution: 

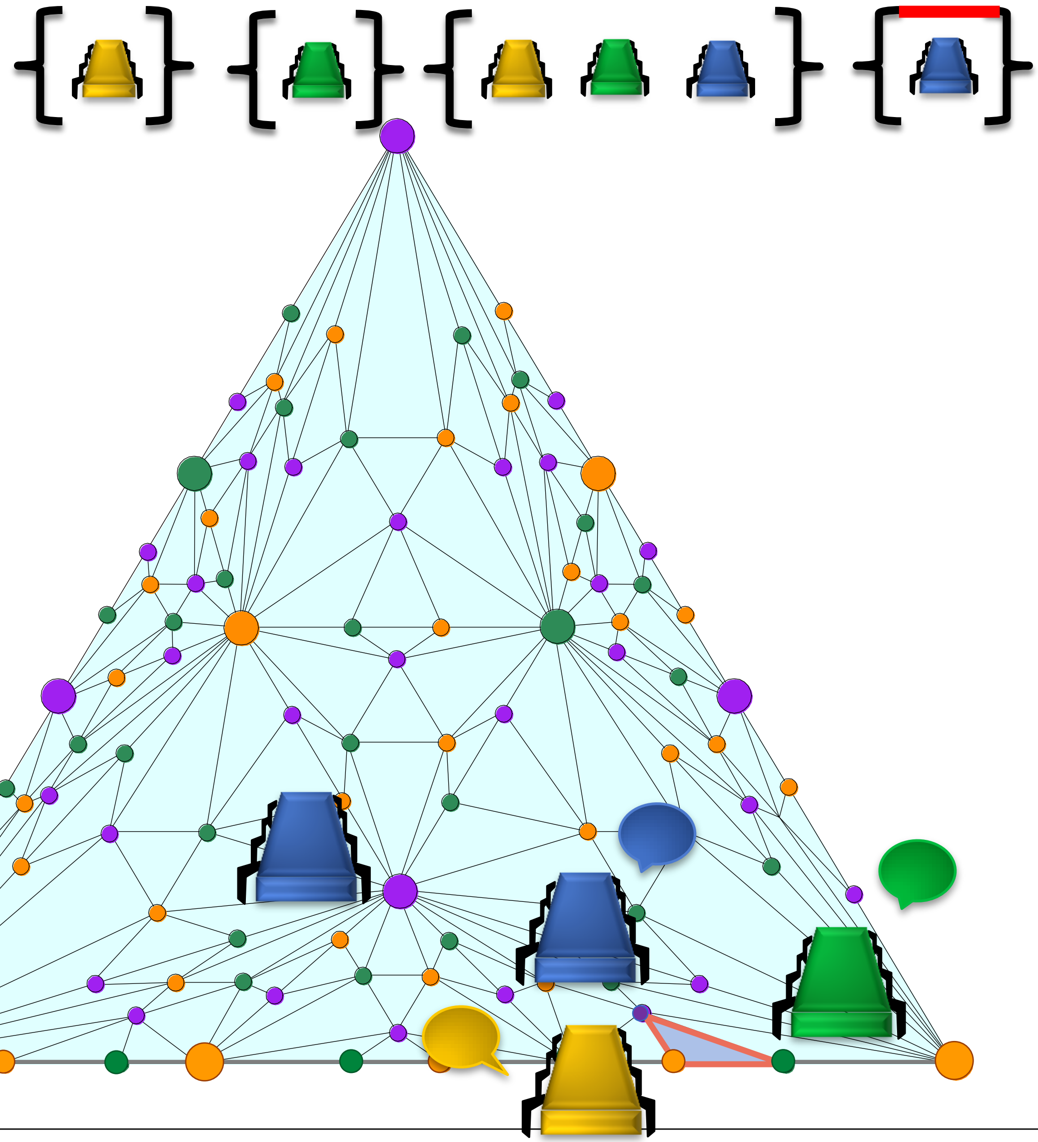


Subdivision Implies Algorithm

- Execution:

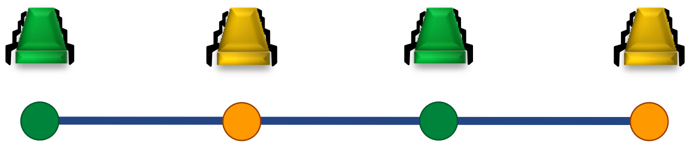


Subdivision Implies Algorithm

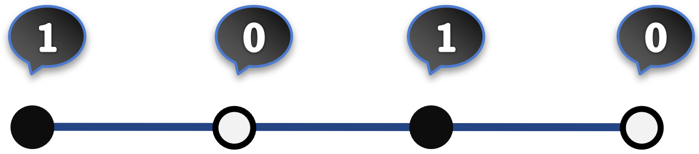
- Execution: 

Outputs

- Each vertex has double coloring:
  - Process id

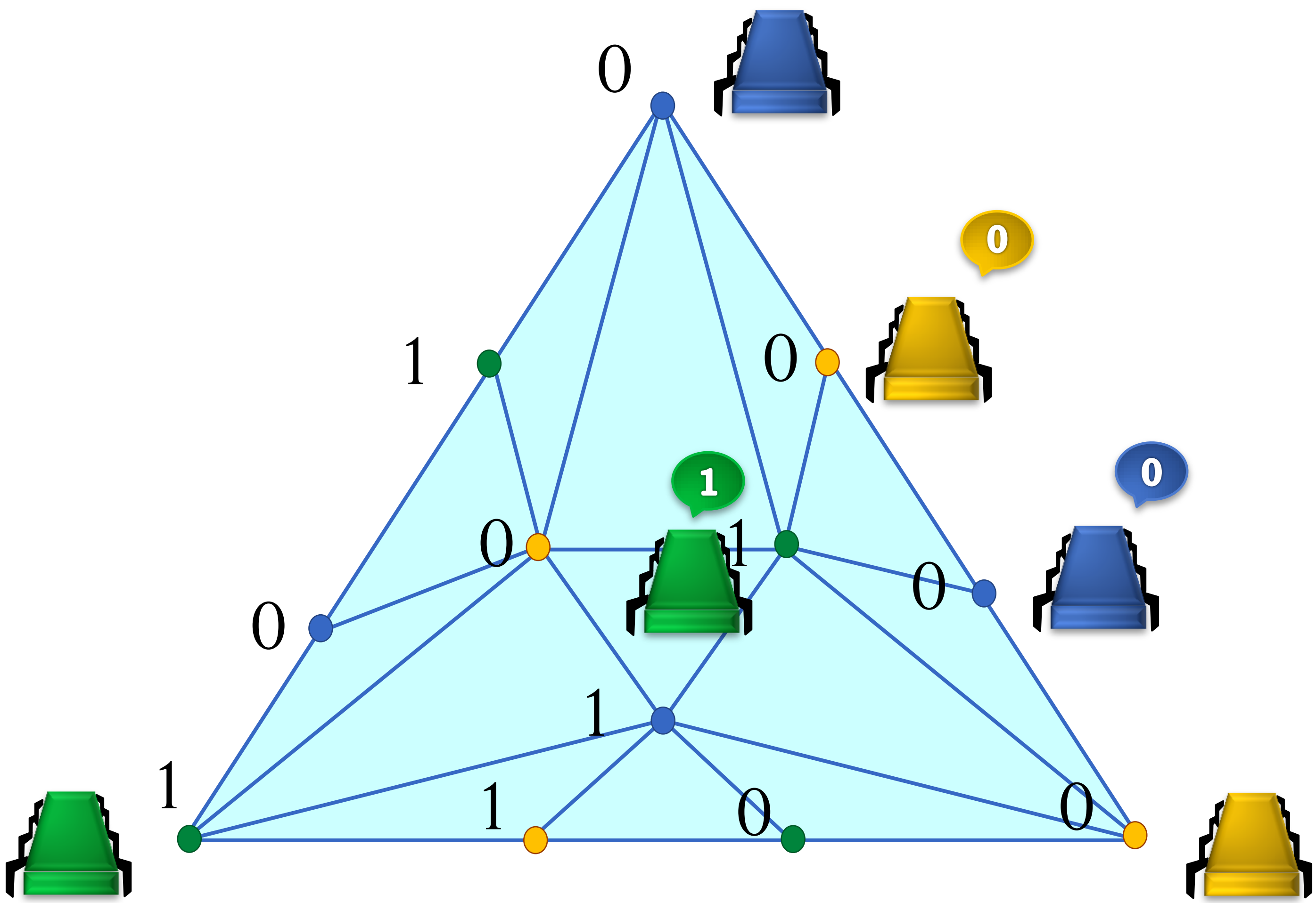


  - Output value



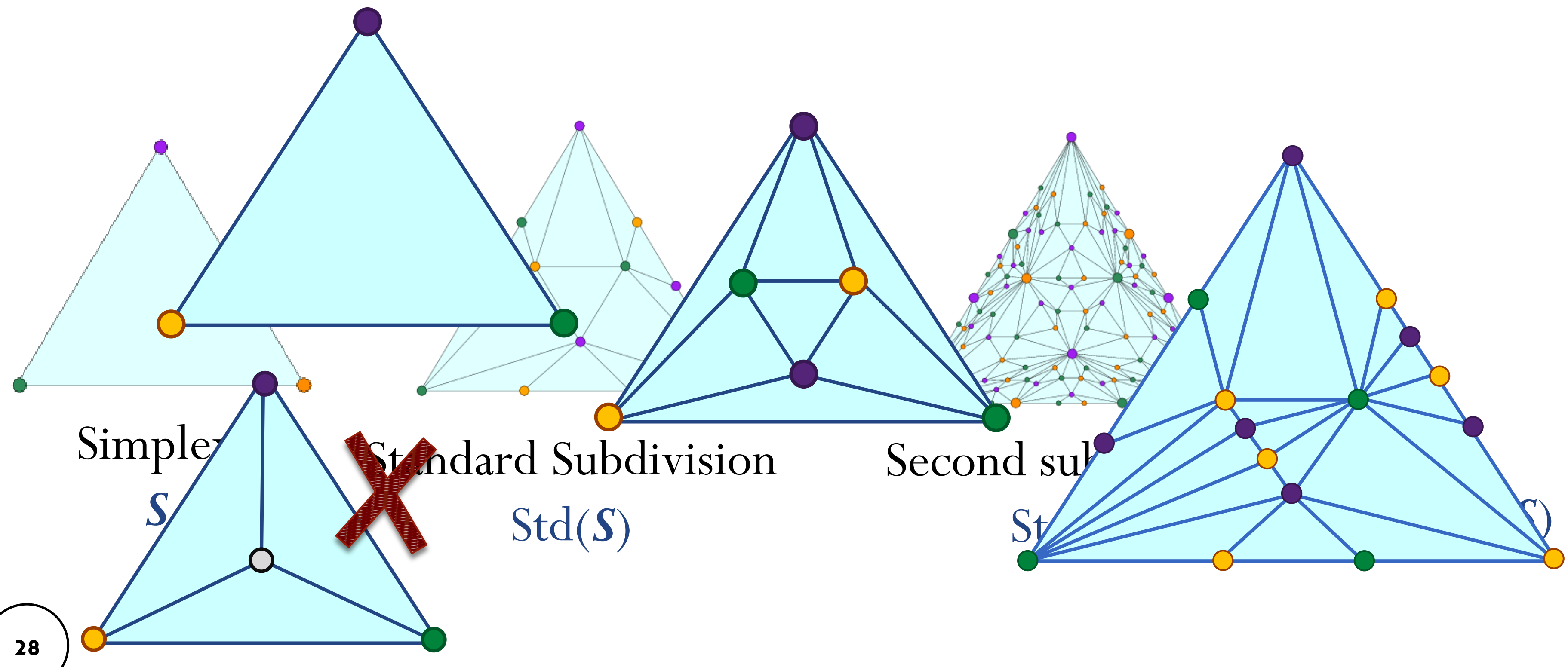
Subdivision Implies Algorithm

- Simplicial approximation



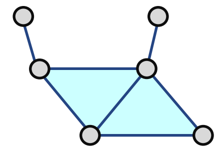
Chromatic Subdivisions

- Chromatic subdivision: can assign a process to each vertex.
- An algorithm is induced by a specific subdivision:
  - Standard chromatic subdivision.

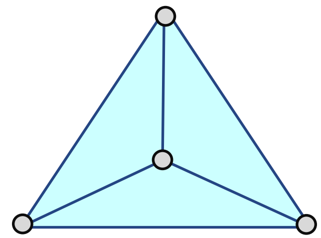


Topological Notions

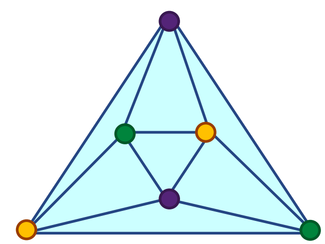
- Simplicial complex



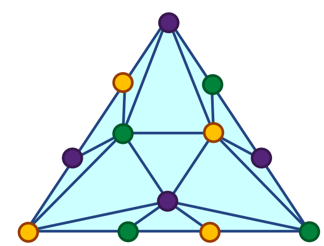
- Subdivision



- Chromatic Subdivision



- Standard chromatic Subdivision



Topology & Distributed Computing

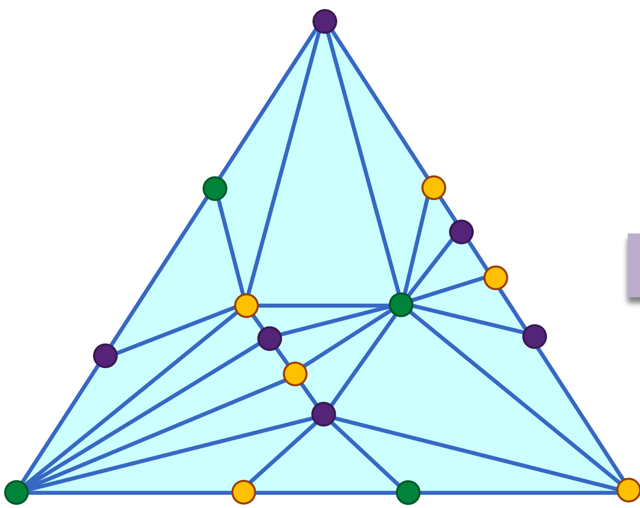
Theorem

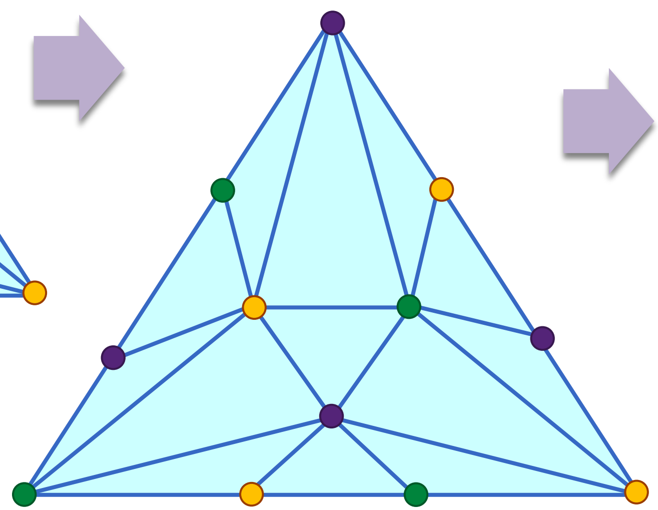
[Herlihy and Shavit 99]

Chromatic Subdivision ⟷ Distributed Algorithm

From Subdivision to Algorithm

Chromatic Subdivision → Standard Chromatic Subdivision → Distributed Algorithm

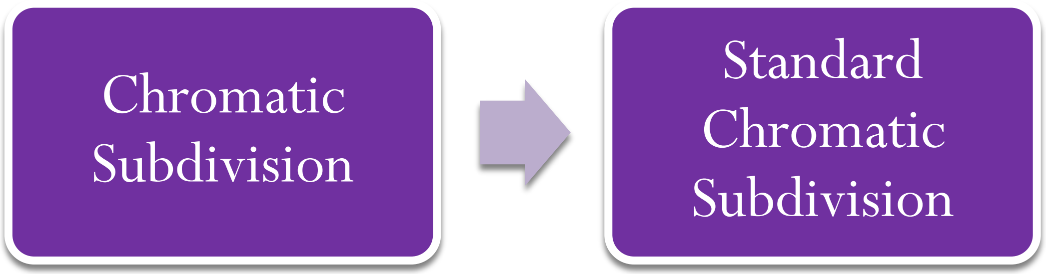




simulated $\leftarrow$ 0
Write(initialState$_i$) to R$_i$
while true do
  r $\leftarrow$ Scan (R$_0$,…,R$_{n-1}$)
  if r contains all then
    return simulated
  simulated $\leftarrow$ 1
  Execute Local A (r)
  if A returns *v* then
    return the same value *v*
  Write (r) to R
…

Colored Simplicial Approximation

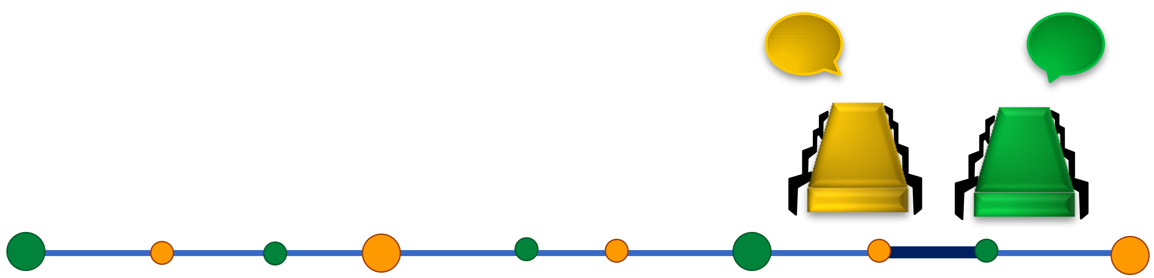
[Herlihy and Shavit 99]



- Colored simplicial approximation theorem:
  any chromatic subdivided simplex can be "approximated" by a standard chromatic subdivision $\mathrm{std}^K(S)$…
  - …for large enough K.
- Yields no bound on K.

Subdivision Implies Algorithm

Standard Chromatic Subdivision → Distributed Algorithm



Step complexity = Number of subdivisions

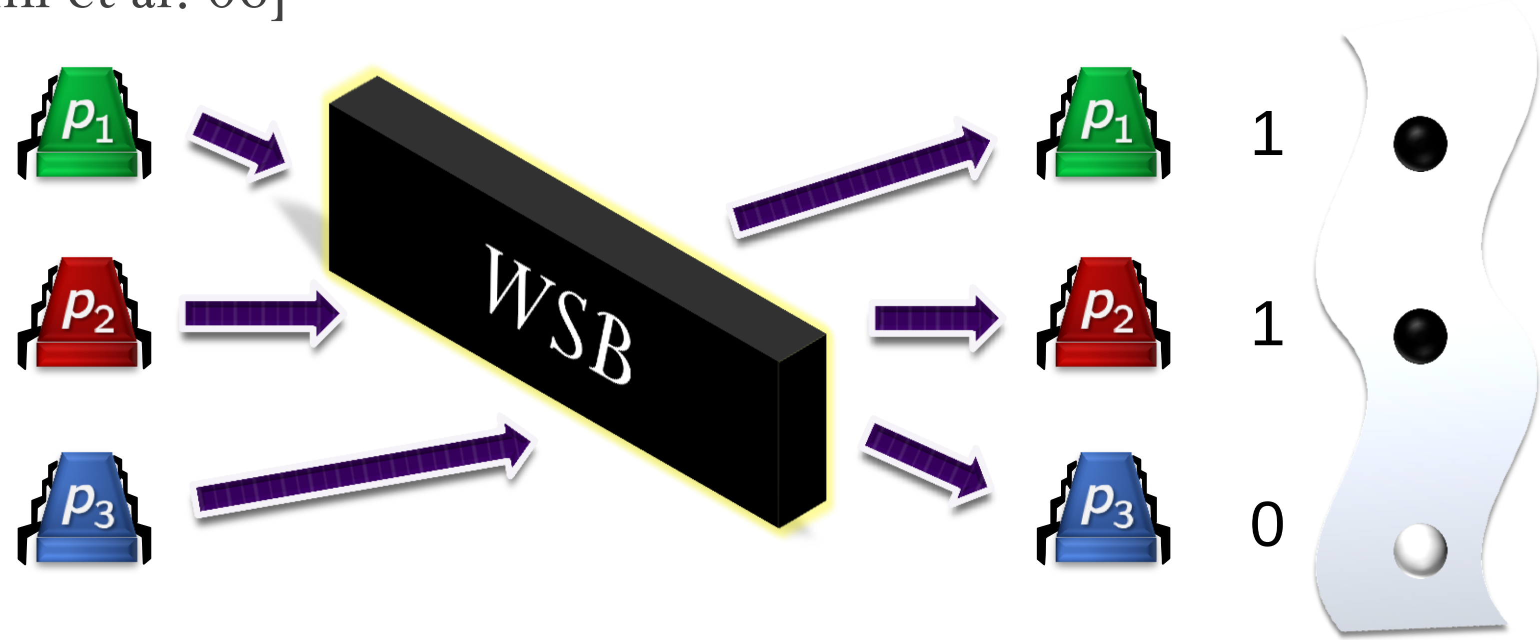
- We count subdivisions, to get the step complexity.

Solving WSB

Properties of the desired solution

Recall: WSB

[Gafni et al. 06]



n Processes

With identifiers

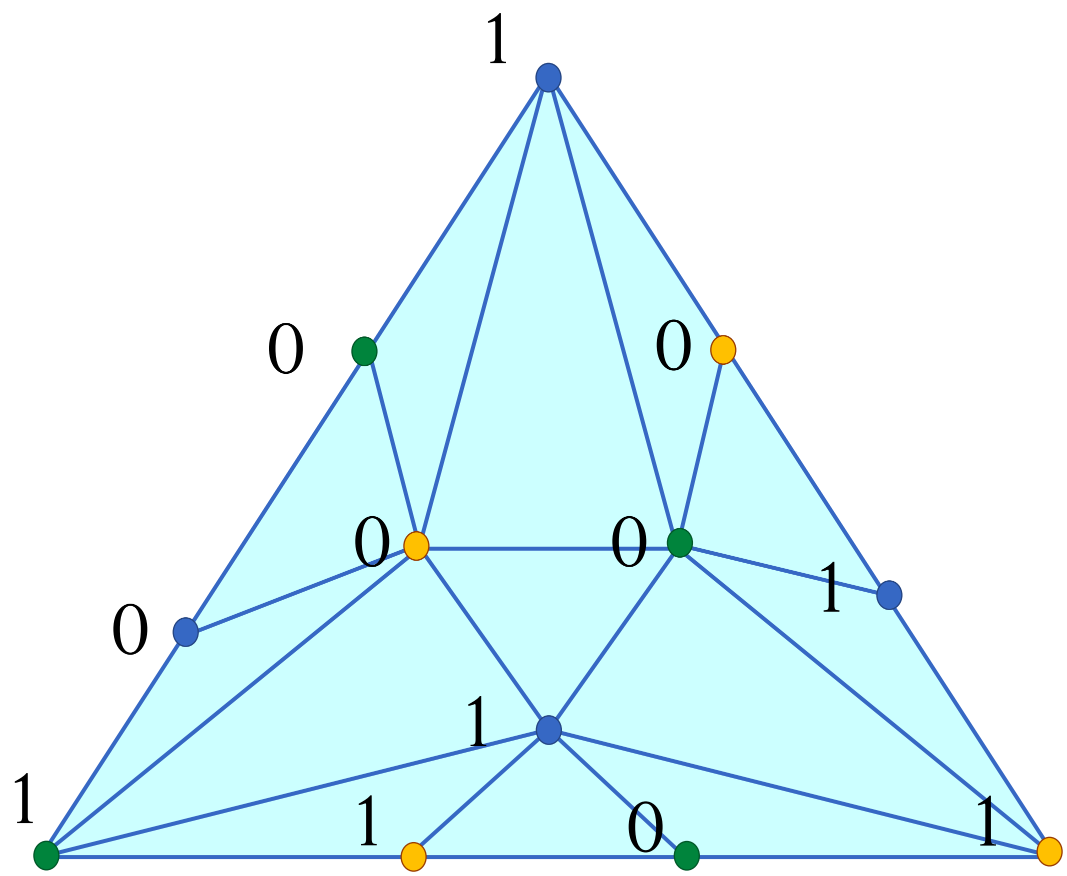
Outputs: 0/1

If all output: not all the same

Processes are only allowed to compare their identifiers

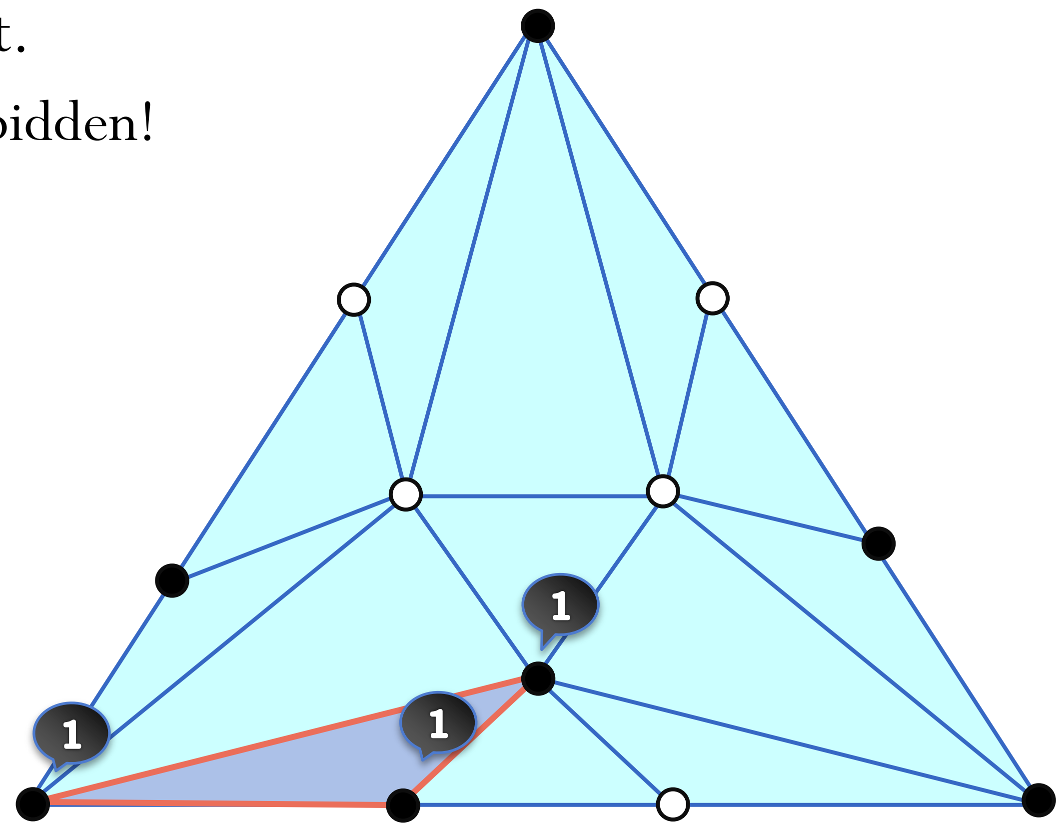
Binary Outputs

- All output values are binary.



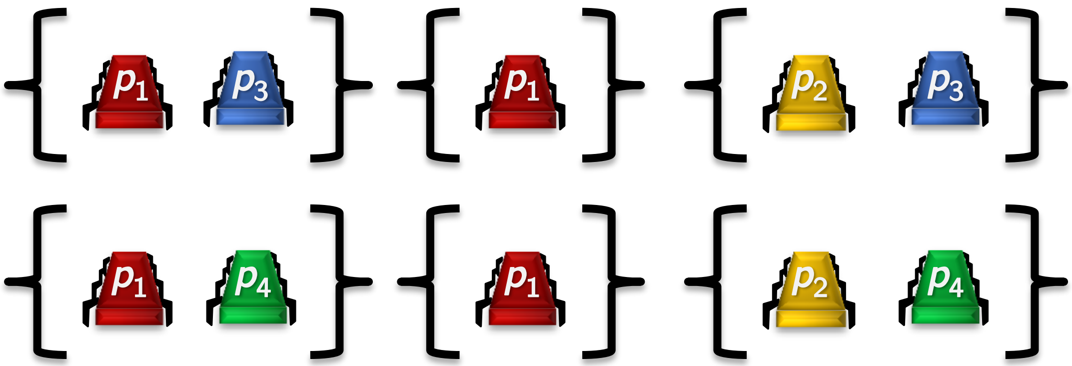
Monochromatic Simplexes

- Represent executions with a single output.
  - Forbidden!



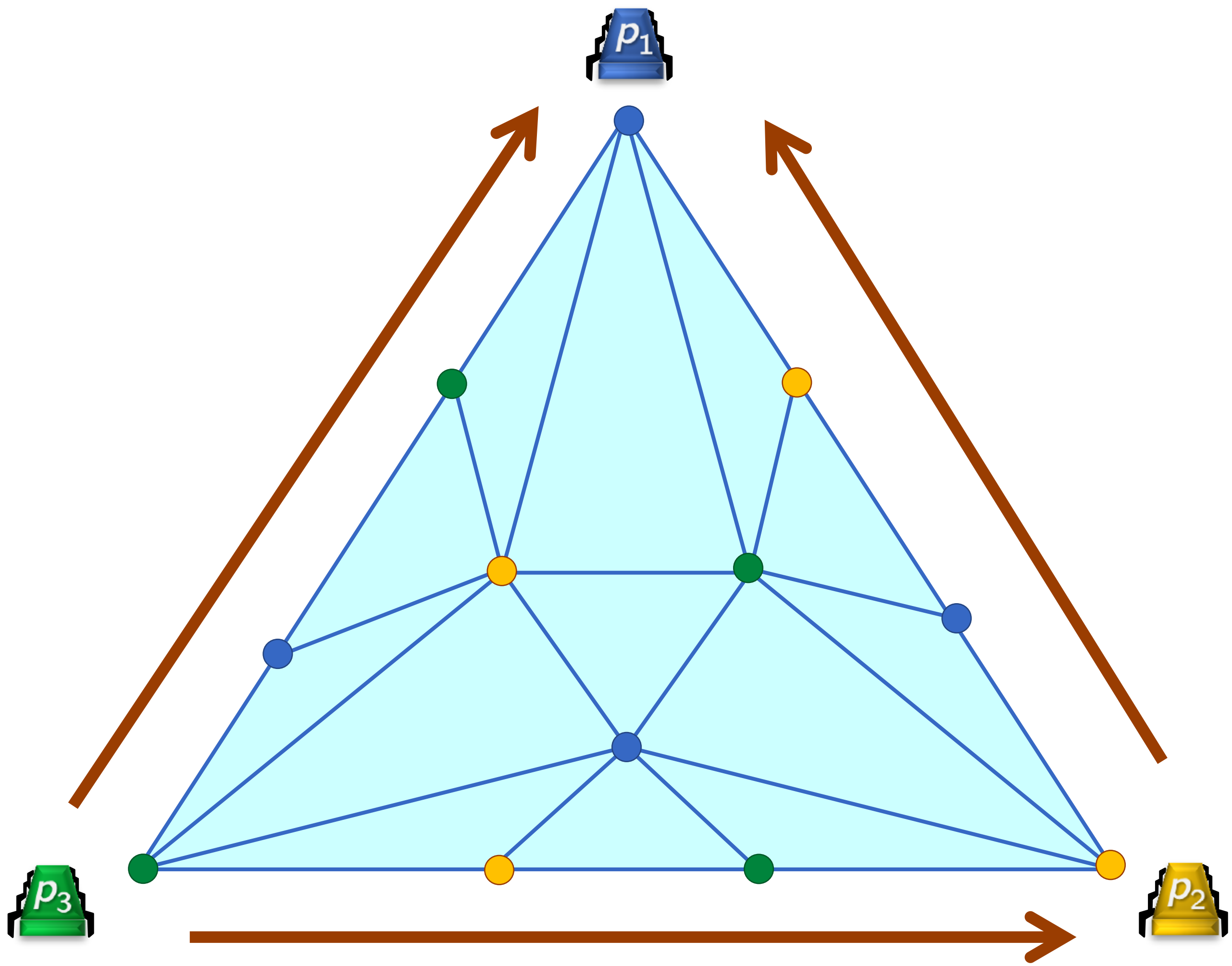
Comparison Based Algorithms

- Processes only compare their values.
- Execution by P_1, P_2, P_3 looks like execution by P_1, P_2, P_4 .

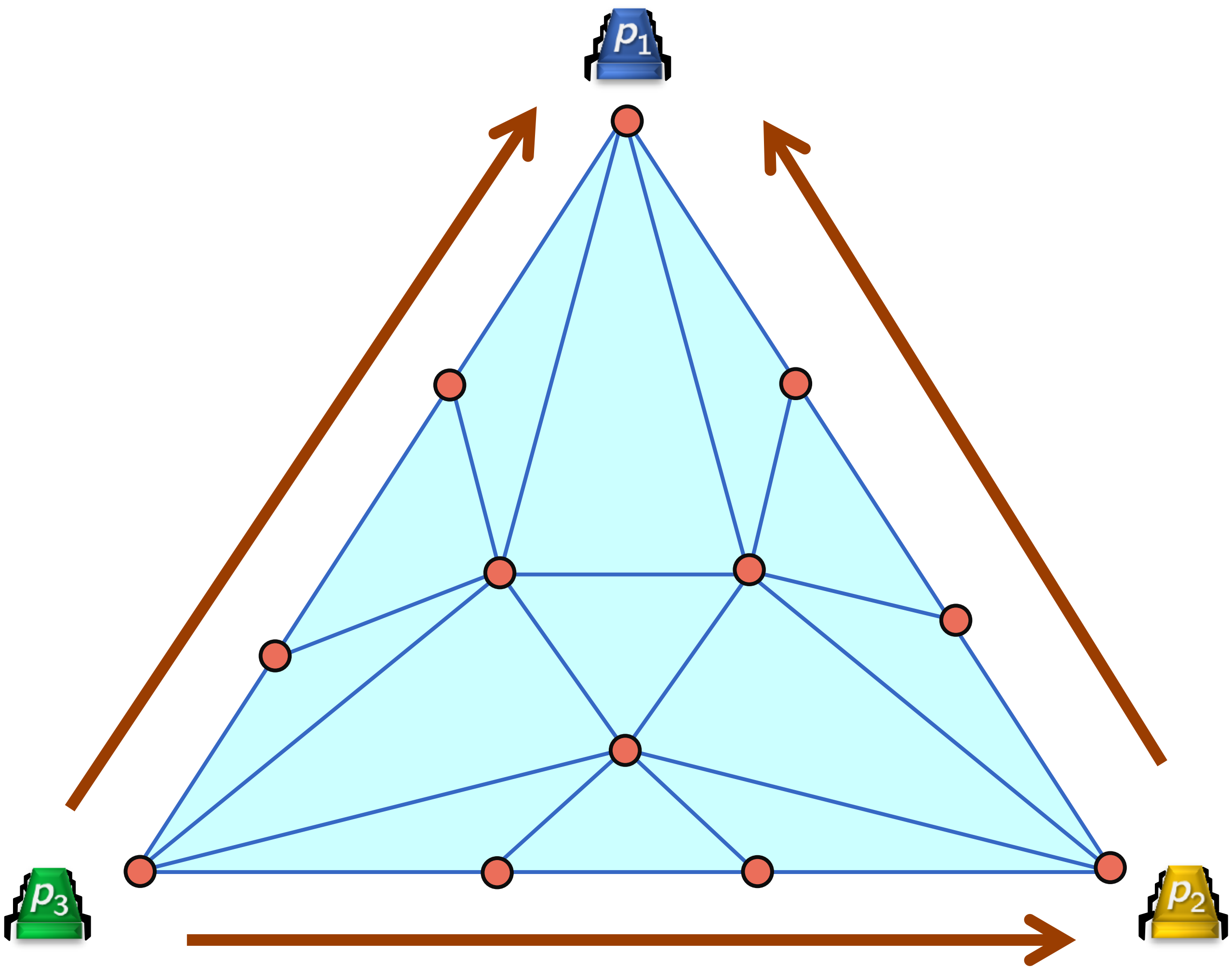


- Topology: implies symmetry on the boundary.

Who is Bigger?



Symmetric Output Coloring

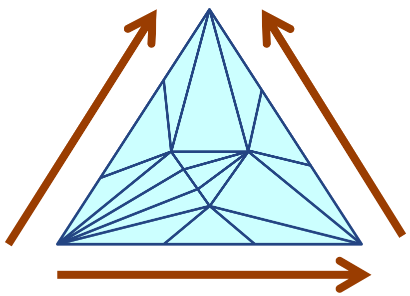


Three Steps to Solution

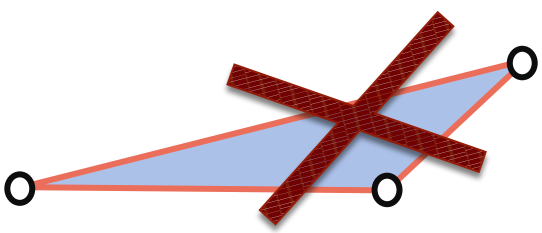
Our Goal

Construct a subdivided simplex & coloring, s.t.:

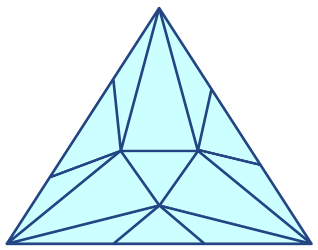
- Symmetric coloring on the boundry.



- Without monochromatic simplexes.



- Standerd chromatic subdivision.



Three Step Plan

- Step 1: find a symmetric subdivision
  with only good monochromatic simplexes.
- Step 2: eliminate mono. simplexes,
  while preserving symmetry.
- Step 3: get a mapping from standard subdivision,
  yielding a WSB coloring and algorithm.

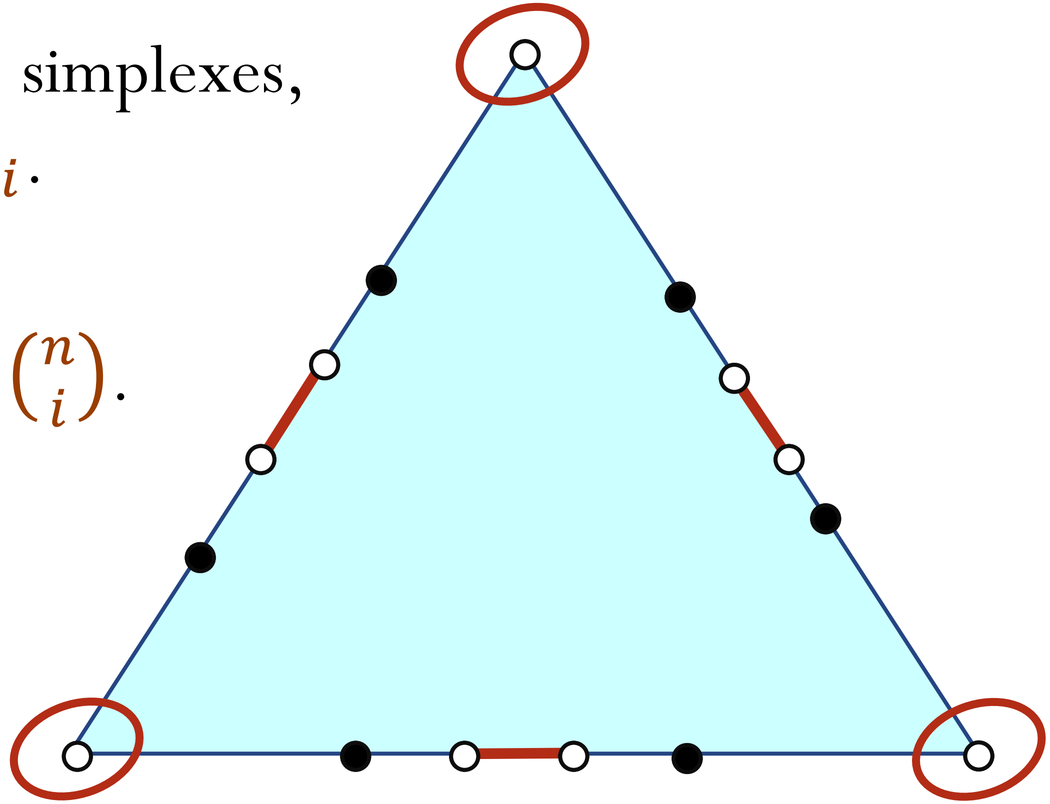
Step One: Symmetric Boundary

1. Create Boundary

- Start by creating a symmetric boundary.

- Each i-face is subdivided and colored:
  - Create k_i 0-mono. simplexes, for some integer k_i.

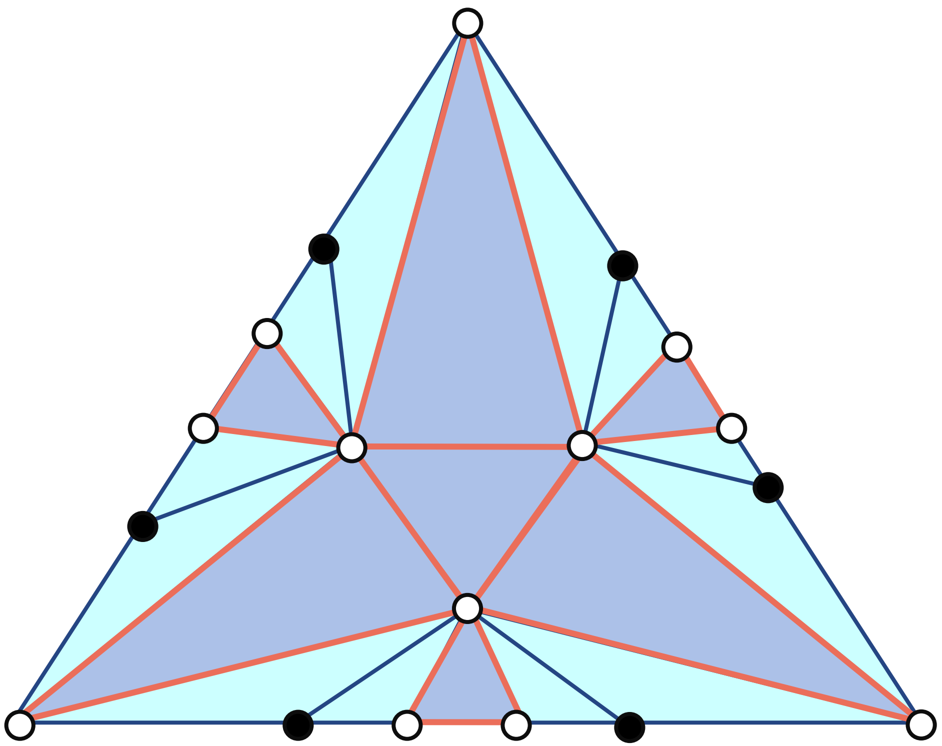
- Number of i-faces $= \binom{n}{i}$.



1. Fill in the Interior

- Add internal 0-mono. simplex.
- More 0-mono. simplexes are created.
- Total number of mono.:

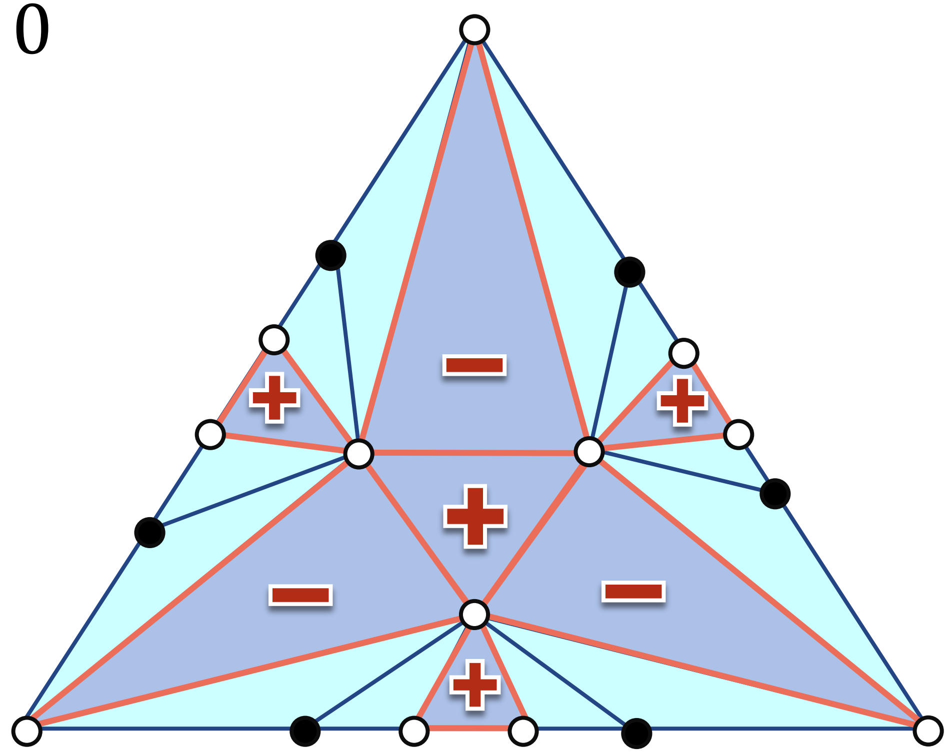
$$1 + \sum_{i=1}^{n-1} \binom{n}{i} k_i$$



1. Counting Mono. Simplexes

- Each k_i has a sign.
- We want:

$$1+\sum_{i=1}^{n-1}\binom{n}{i}k_i=0$$



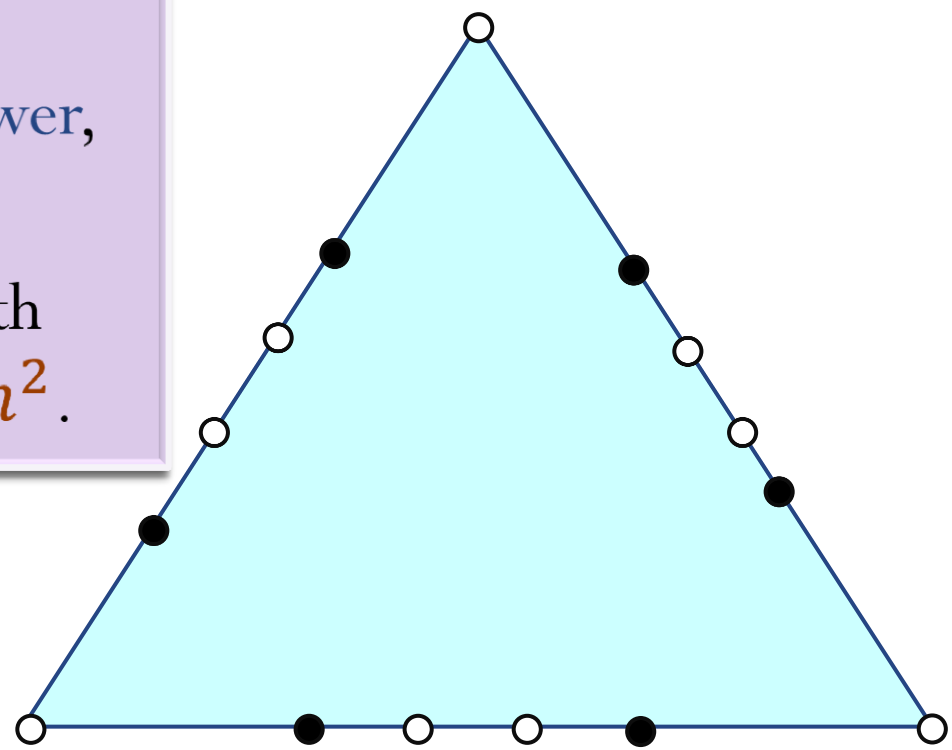
1. Creating the Boundary

- We want: $1 + \sum \binom{n}{i} k_i = 0$.
- Subdivide boundaries simultaneously.

Lemma:

- If n is not a prime power, such k_is exist.
- There is a solution with small values: $|k_i| < n^2$.

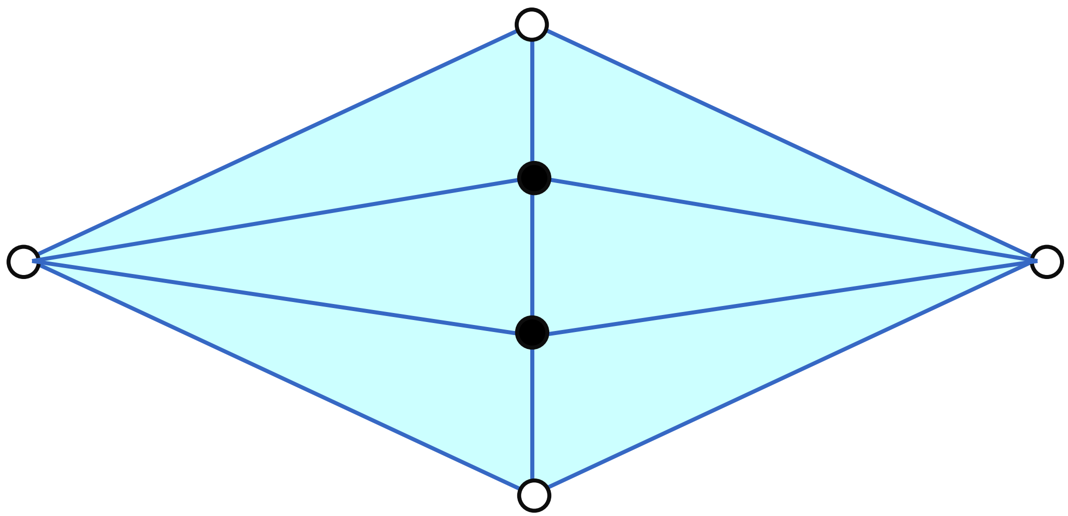
- $O(1)$ subdivisions.



Step Two:
Eliminating Mono. Simplexes

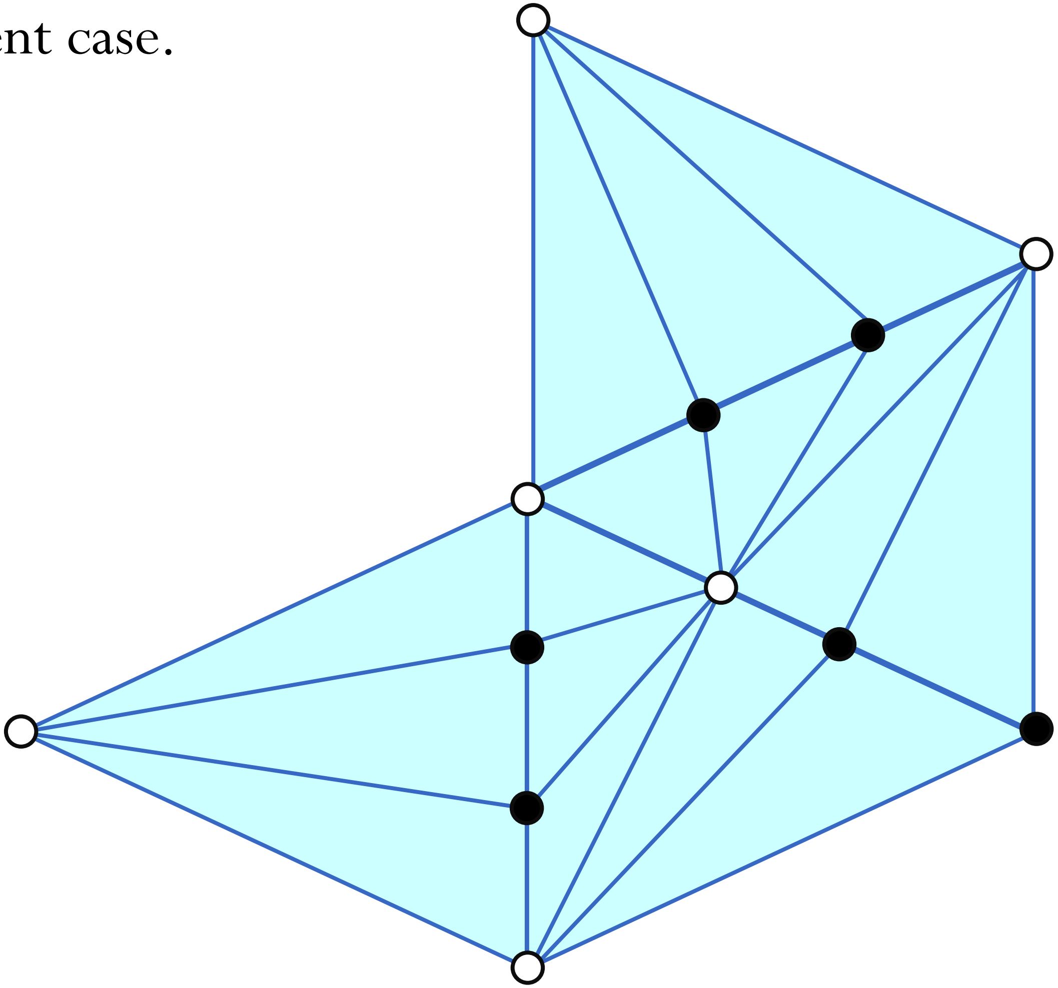
Eliminating Monochromatic Simplexes

- Use subdivisions to eliminate monochromatic simplexes.
- While preserving symmetry on the boundary.
  - Adjacent case.



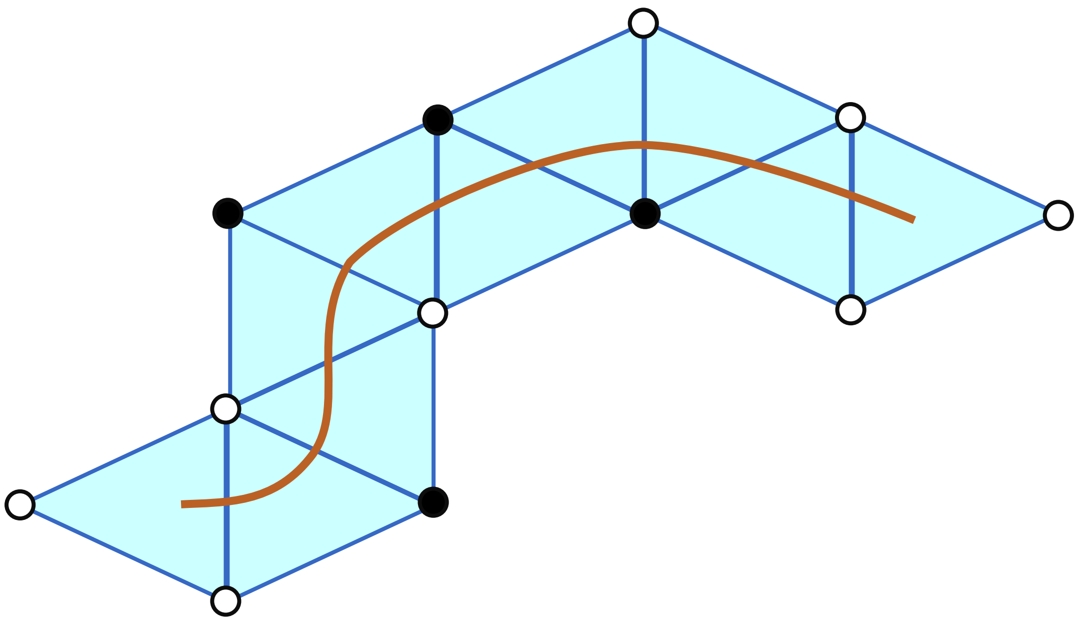
Eliminating Monochromatic Simplexes

- Use subdivisions to eliminate monochromatic simplexes.
  - Non Adjacent case.



Eliminating Monochromatic Simplexes

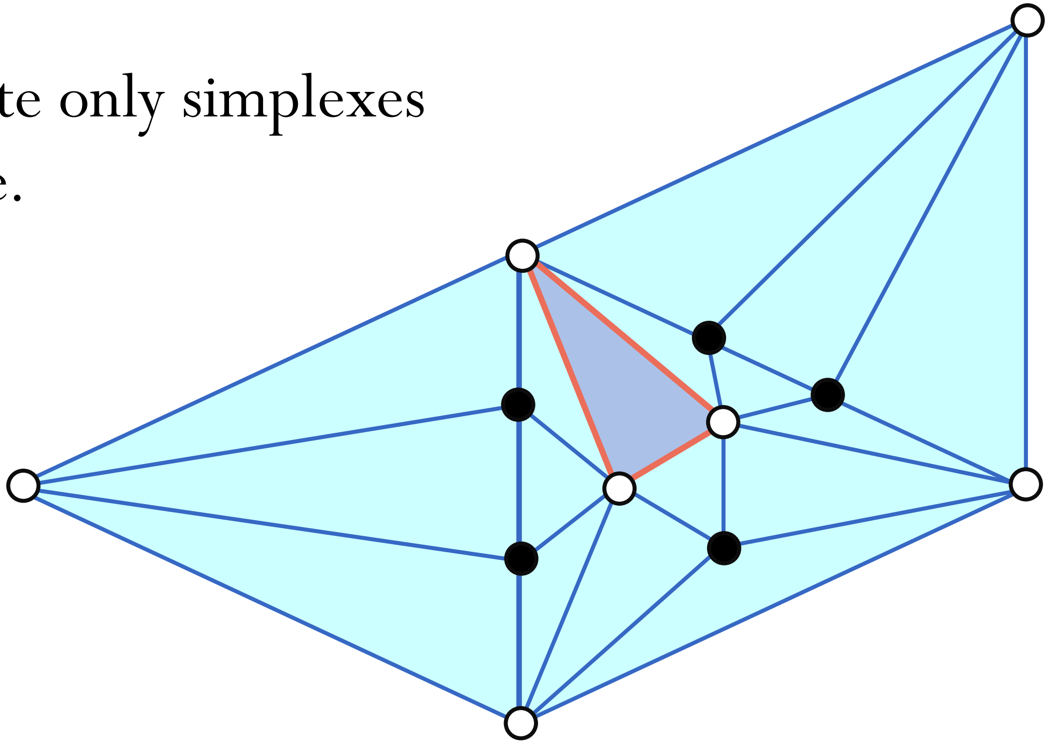
- We can use subdivisions to eliminate monochromatic simplexes.
- Similar constructions for longer paths.
- $O(\ell)$ subdivisions for ℓ-length path.



Odd Paths

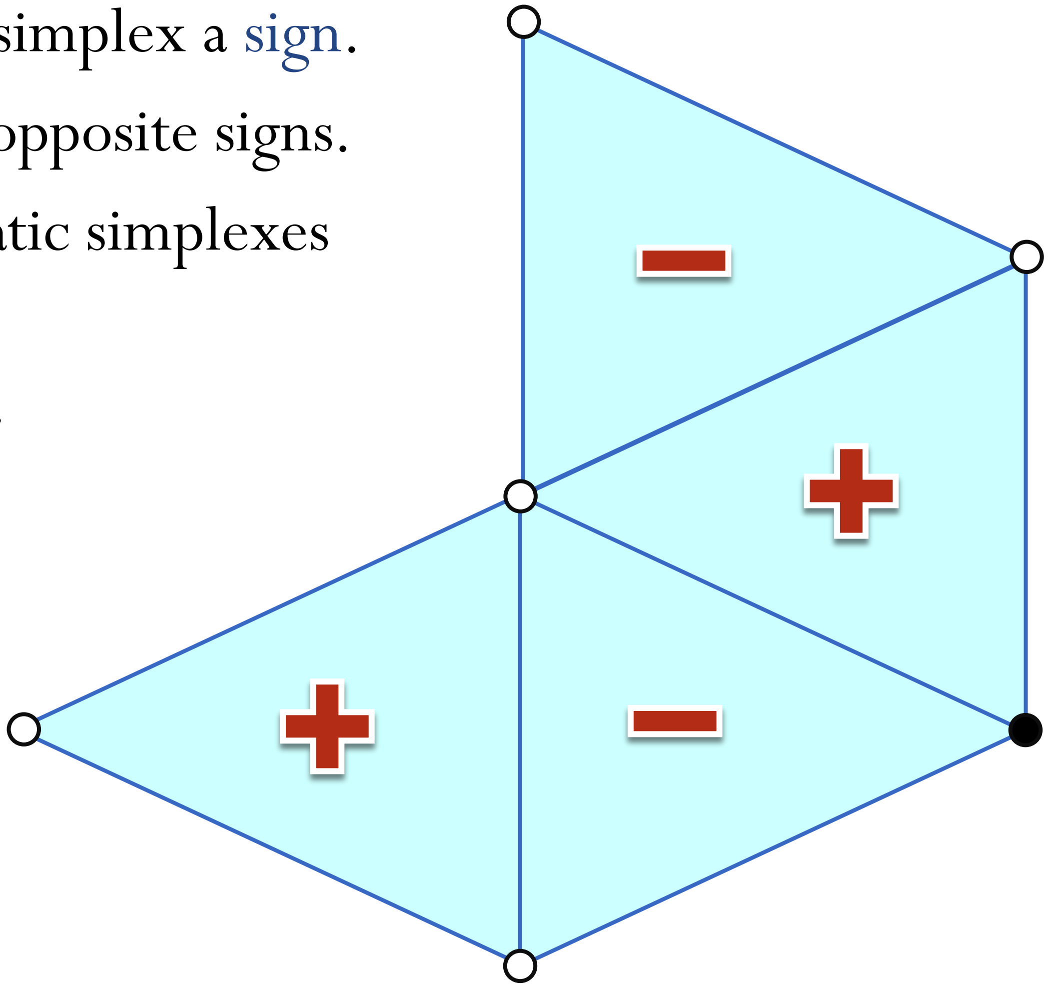
- Eliminate odd length paths?
  - Impossible!

- We can eliminate only simplexes of even distance.



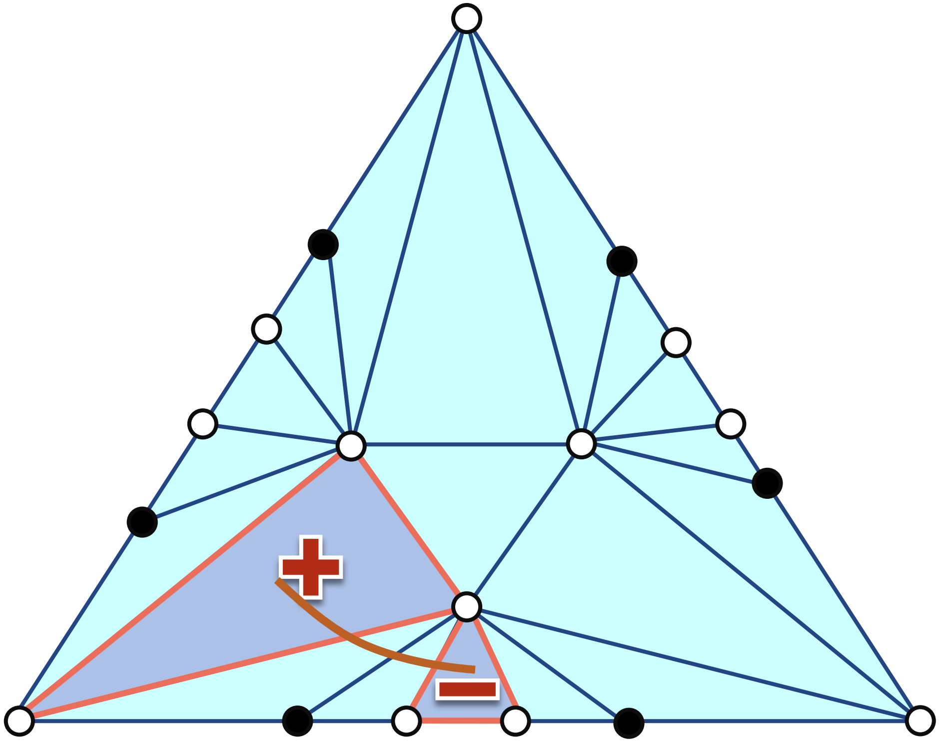
Signs

- Give each maximal simplex a sign.
- Can eliminate only opposite signs.
- Count monochromatic simplexes by their sign.
    - This is an invariant.



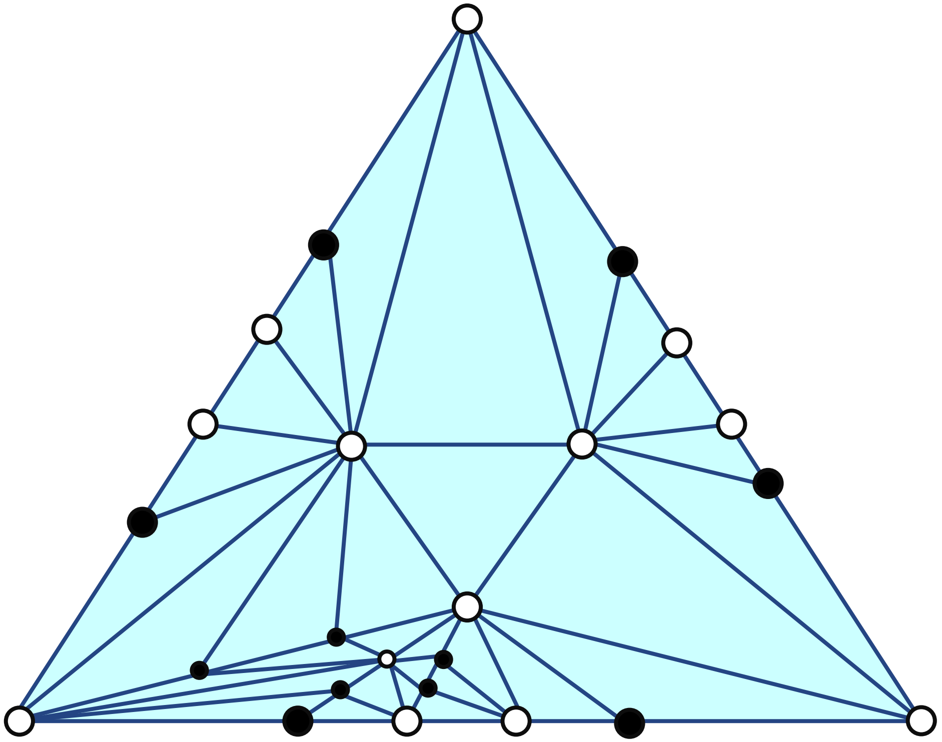
2. Create Path

- Choose mono. simplexes of opposite signs.
- Find a connecting path.



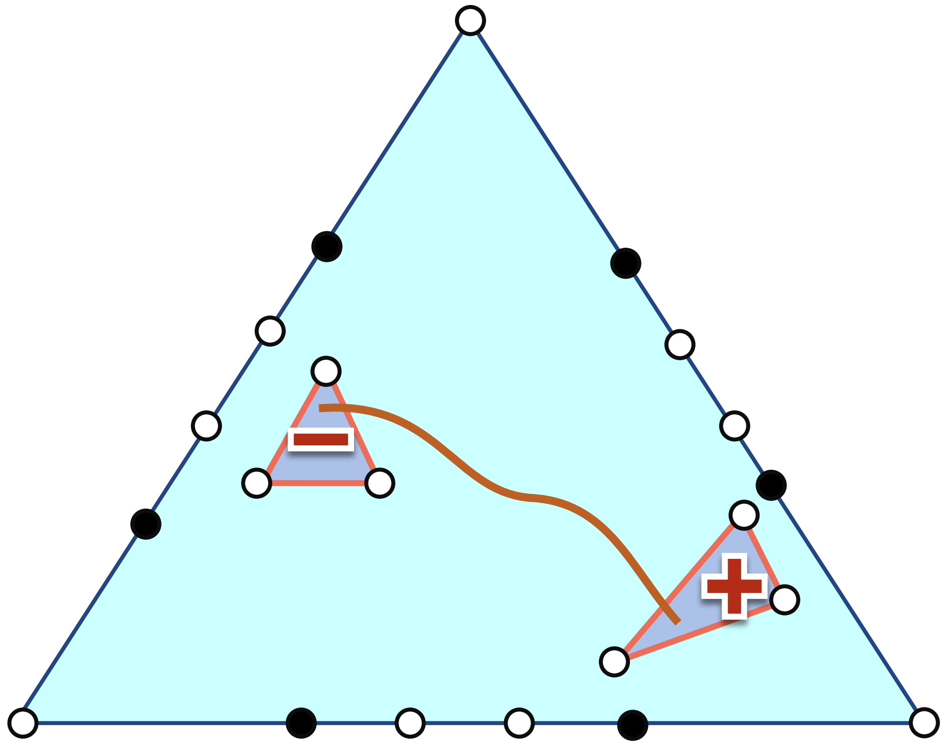
2. Eliminate

- Choose mono. simplexes of opposite signs.
- Find a connecting path
- Eliminate.



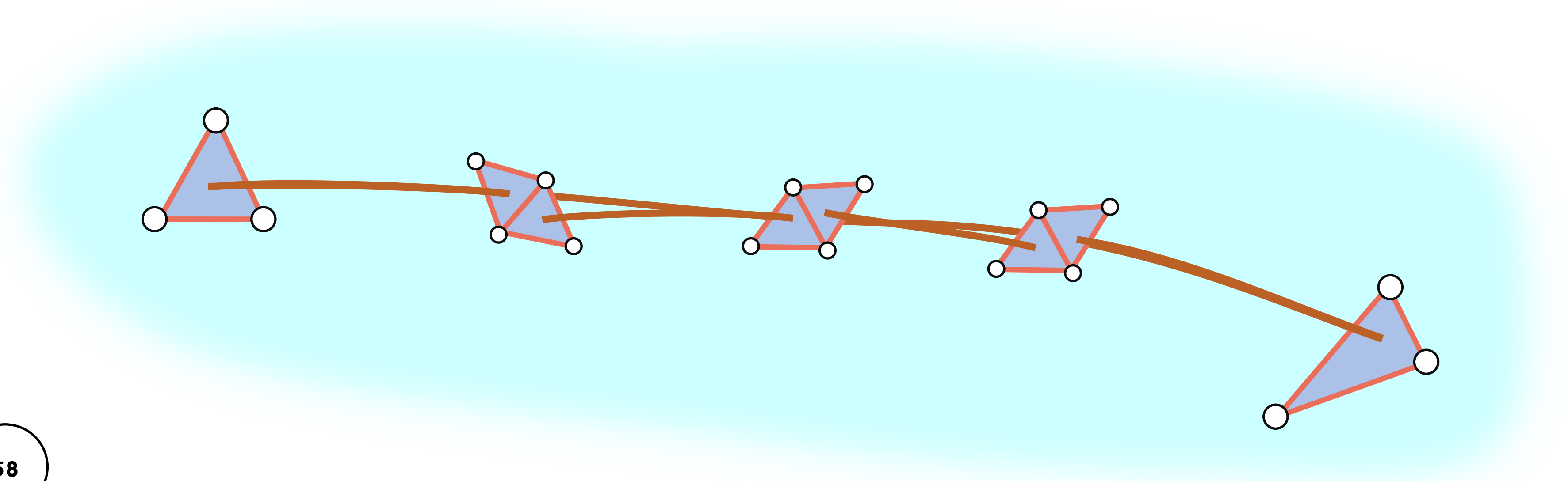
2. Longer Paths

- Path between simplexes of opposite signs.
- The longer the path, more subdivisions are needed.



2. Longer Paths

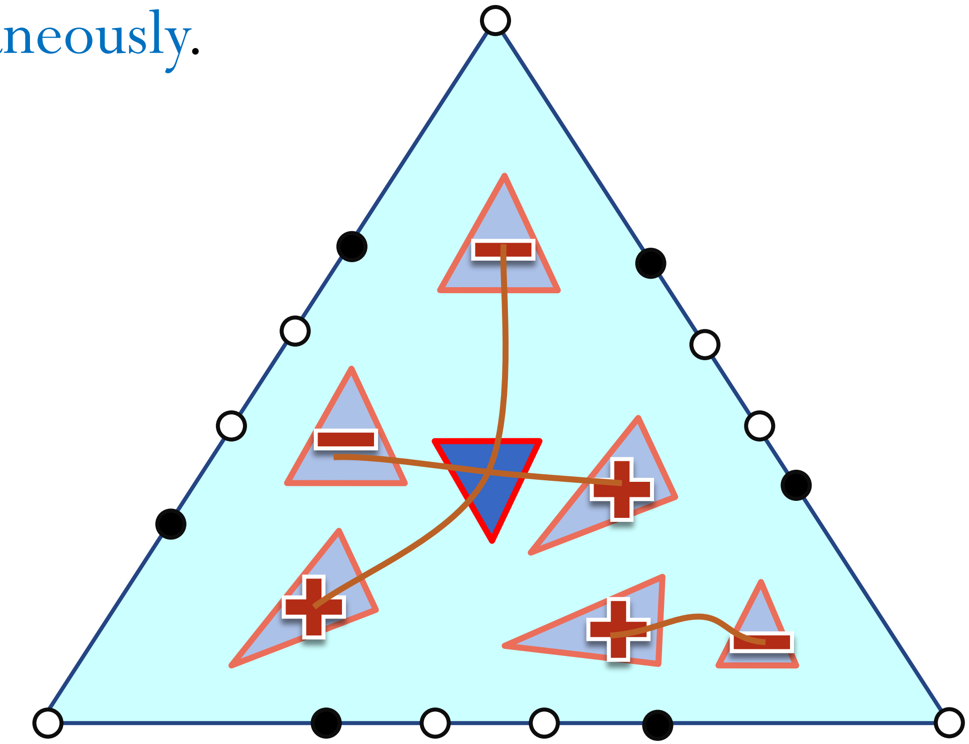
- Path between simplexes of opposite signs.
- The longer the path, more subdivisions are needed.
- Solution:
  - Break into short paths.
  - Many n-length paths, subdivided simultaneously in $O(n)$.



2. Eliminate Paths

- Match all simplexes in pairs.
- Eliminate pairs.

- Cannot be done simultaneously.

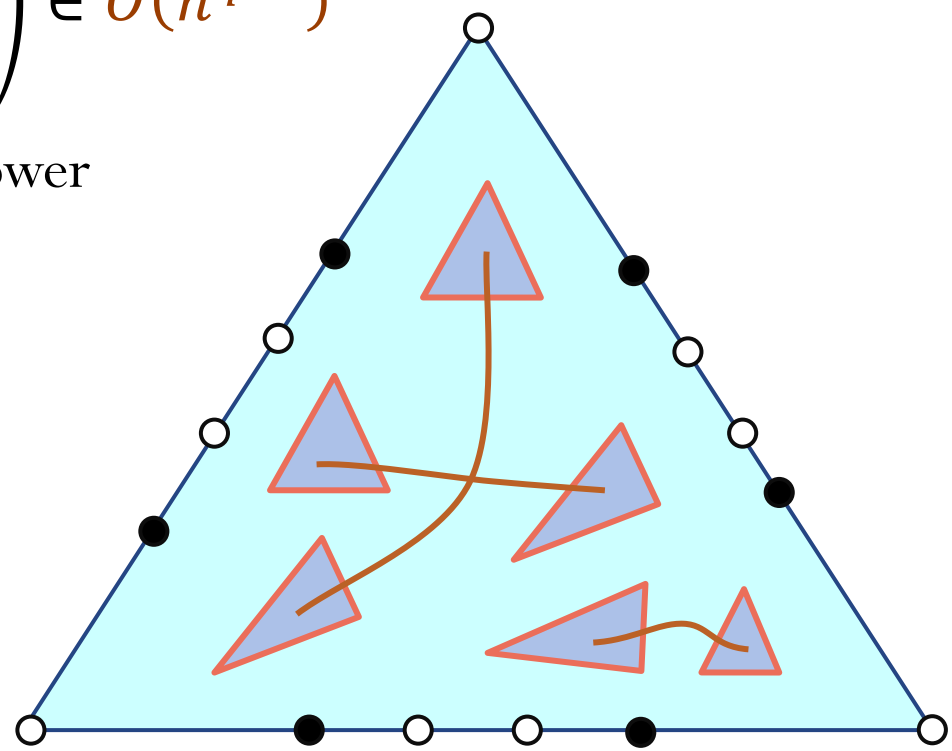


2. Number of paths

- Number of paths:
  - Half the number of mono. simplexes:

$$\frac{1}{2}\left(1+\sum_{i=1}^{n-1}\binom{n}{i}|k_i|\right)\in O(n^{q+2})$$

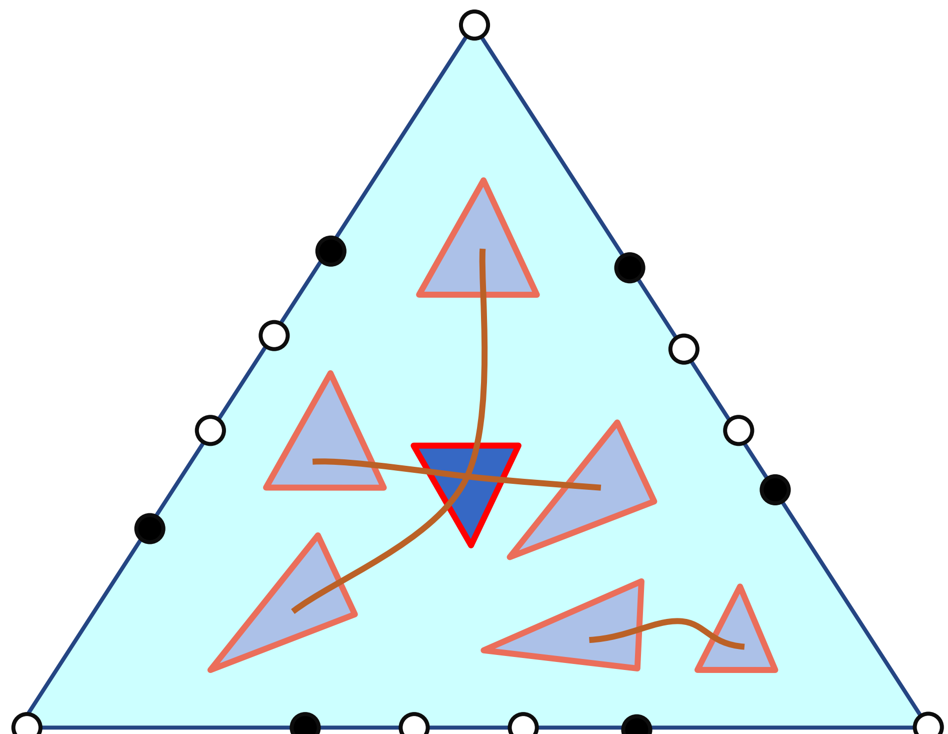
  - q is the largest prime power dividing n.



2. Number of Subdivisions

- The "expensive" part:
  - A simplex shared by many paths is subdivided many times.

- $O(n^{q+3})$ subdivisions.

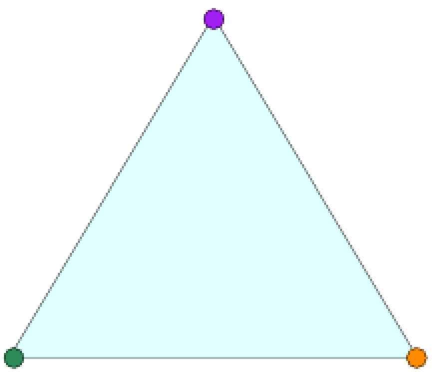


?

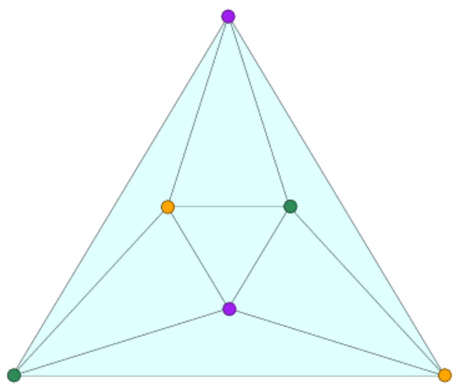
Possible solution: finding disjoint paths.

Step Three: The Output Map

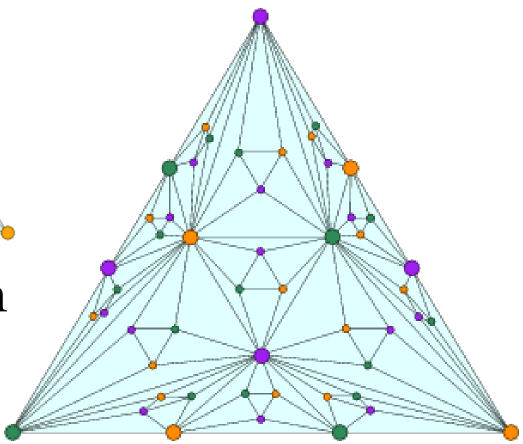
Cone Subdivision



Simplex ***S***



Cone subdivision



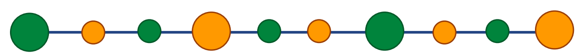
Second cone subdivision

… *L*-cone subdivision

Cone Subdivision

Simplex ***S***

Cone subdivision



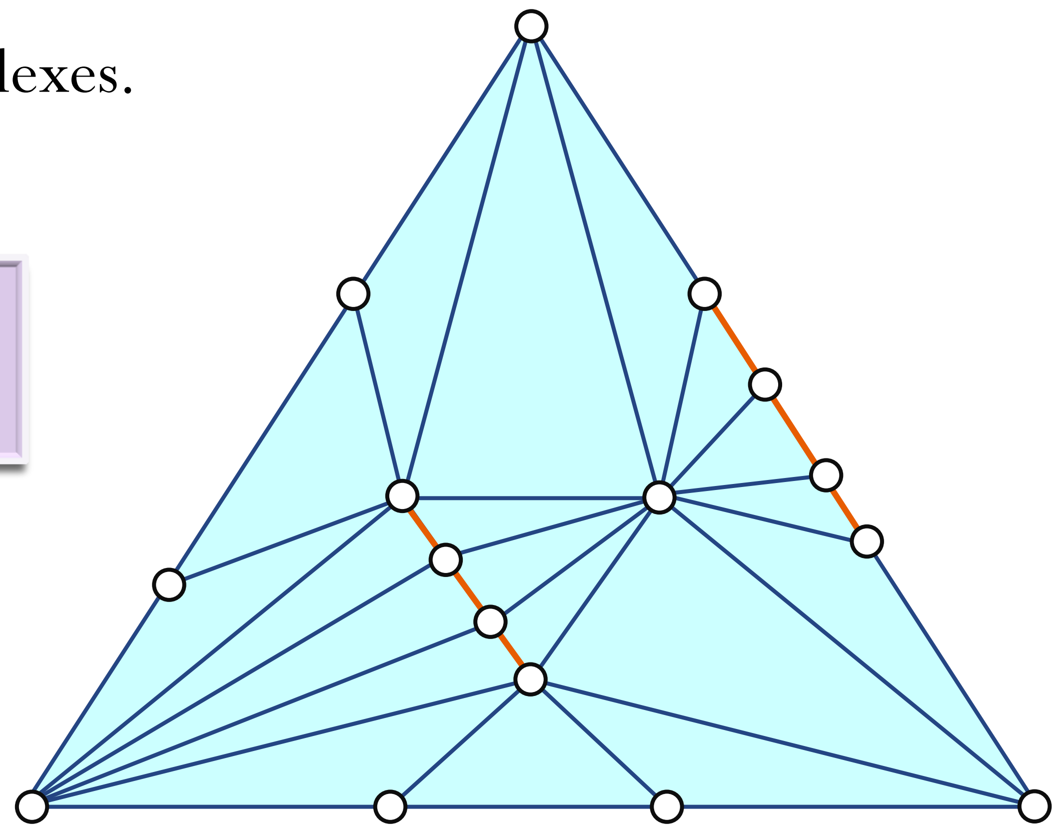
Second cone subdivision

$\cdots$ *L*-cone subdivision

Constructing Subdivisions

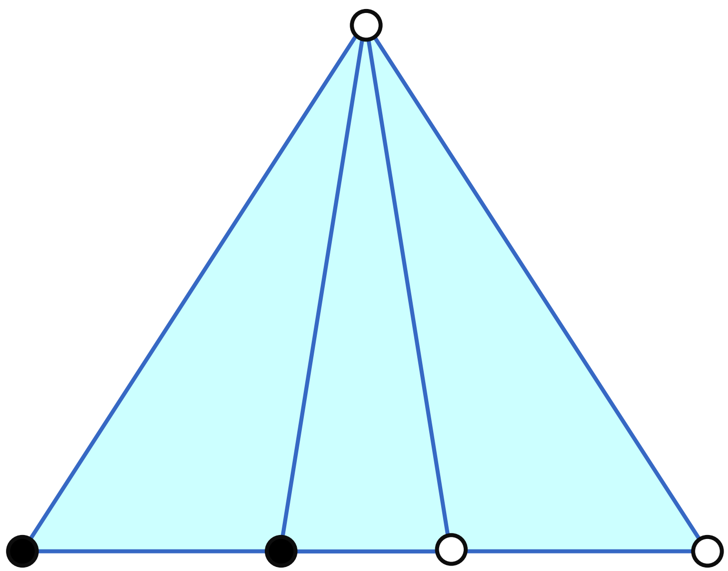
1. Pick simplexes and an integer L.
2. L-cone (in parallel) these simplexes.
3. Extend to all simplexes.

These are the subdivisions we used



3. Cone Subdivisions

- We use cone subdivisions.
- How to derive an algorithm?

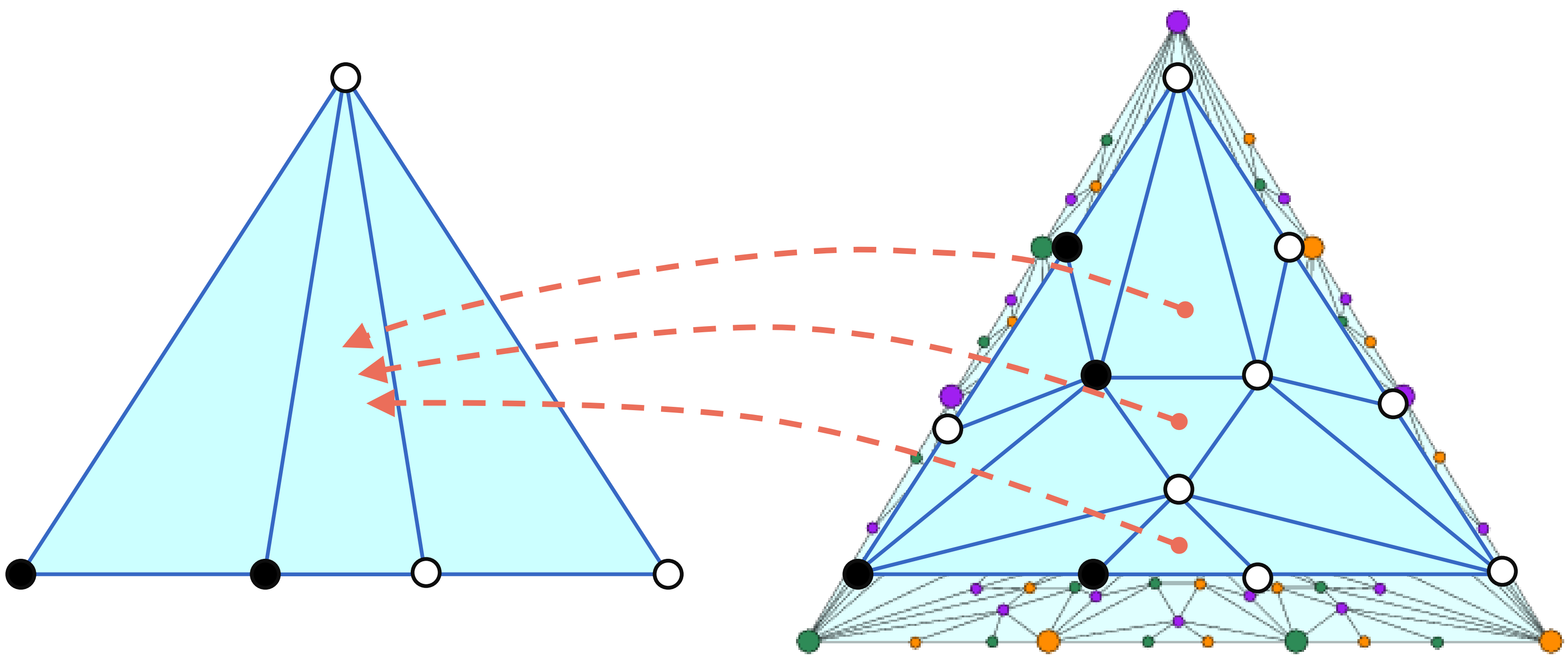


?

simulated $\leftarrow$ 0
Write(initialState$_i$) to R$_i$
while true do
  r $\leftarrow$ Scan (R$_0$,…,R$_{n-1}$)
  if r contains all then
    return simulated
  simulated $\leftarrow$ 1
  Execute Local A (r)
  if A returns *v* then
    return the same value *v*
  Write (r) to R
…

Cone Subdivisions

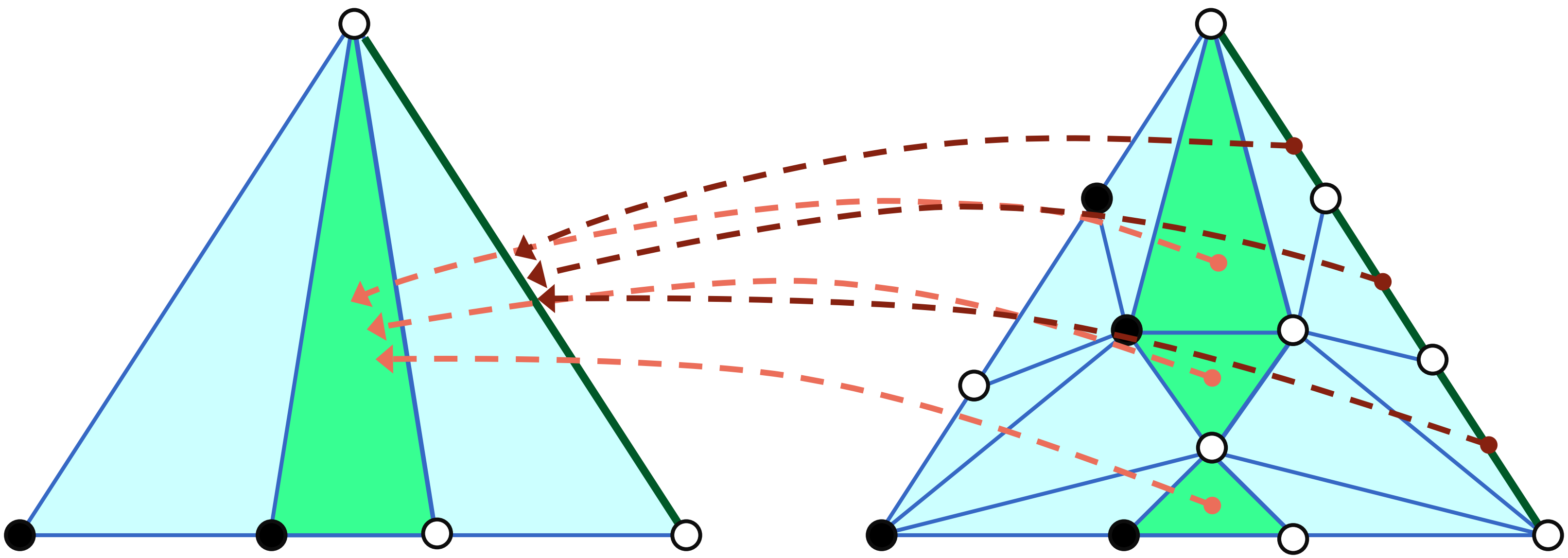
- Use cone subdivisions,
  than map standard subdivision to them.



- Without using simplicial approximation!

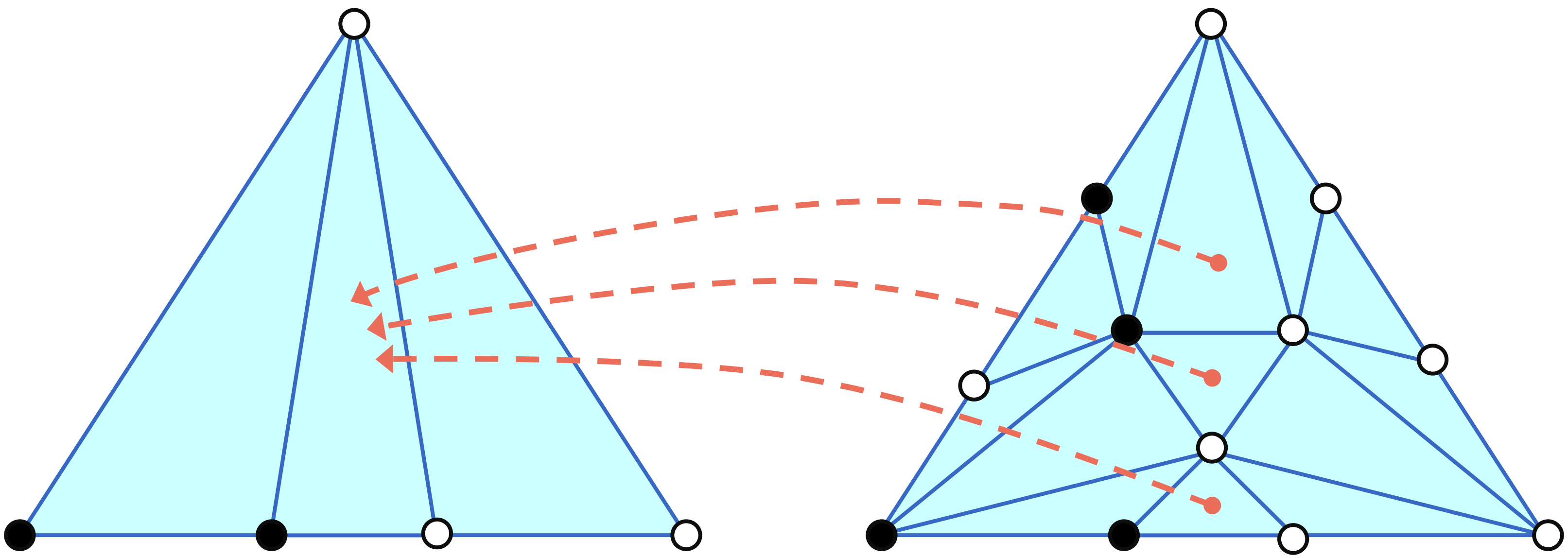
3. Mapping

- Solution:
  - Map standard chromatic subdivisions to cone subdivisions.
  - “Pull back” coloring accordingly.



3. Mapping

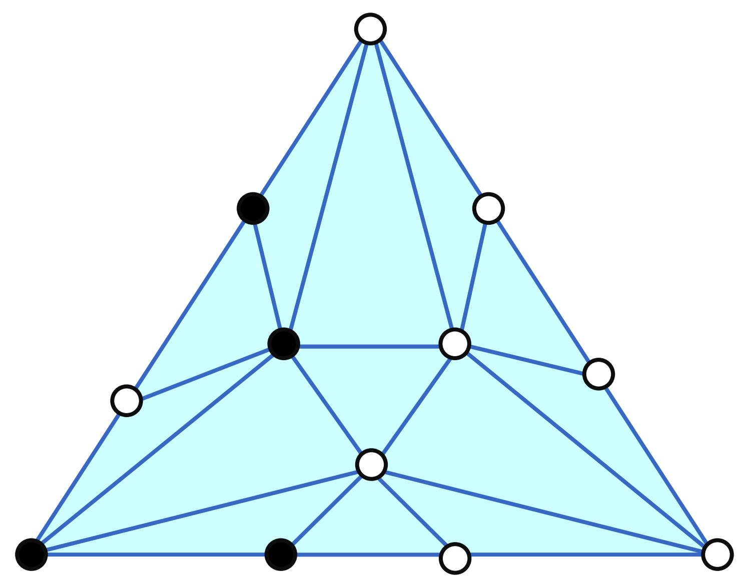
- Properties:
  - Map simplexes to simplexes.
  - Preserve process identifiers.
  - Preserve the structure of the subdivision.



3. Mapping

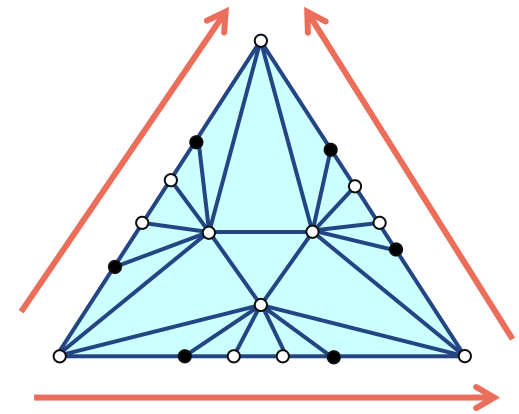
- From a standard chromatic subdivision, we derive an algorithm.

```
simulated ← 0
Write(initialState_i) to R_i
while true do
   r ← Scan (R_0,…,R_{n−1})
  if  r contains all then
        return simulated
  simulated ← 1
  Execute Local A (r)
  if A returns v then
       return the same value v
  Write ( r) to R
…
```

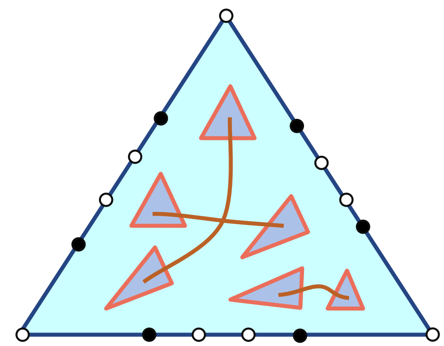


Wrap Up

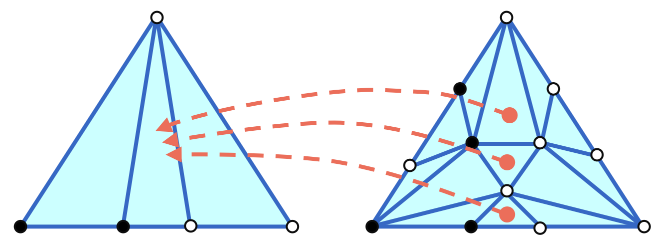
- Step 1: symmetric subdivision,
  with 0 mono. simplexes by sign.
  - $O(1)$ subdivisions.



- Step 2: eliminate mono. simplexes,
  while preserving symmetry.
  - $O(n^{q+3})$ subdivisions.



- Step 3: mapping from standard subdivision.
  - No subdivisions.



Total: $O(n^{q+3})$ subdivisions.

Main Results

- Upper bound on the complexity of solving WSB and $(2n\text{-}2)$-renaming.
  - Not just existence.
- Explicit mapping of standard chromatic subdivision to cone subdivision.
  - “We do not discuss Lebesgue numbers in a polite company” [M. P. Herlihy].
- Improved path-elimination procedure.
  - Do not depend on the length of the path.

Open Questions

- Non-intersecting matching paths.
- Intuitive WSB algorithm.
- $(2n\text{-}3)$-renaming and below.
- Colored computability theorem with bounds.