Distributed computing and topology:

A tiny introduction

Sergio Rajsbaum
Instituto de Matemáticas, UNAM

October 12, 2021
Ecole Polytechnique

Many kinds of distributed systems

Many kinds of distributed systems

- Mutlicore, various shared-memory systems

Many kinds of distributed systems

- Mutlicore, various shared-memory systems
- Internet

Many kinds of distributed systems

- Mutlicore, various shared-memory systems
- Internet
- Wireless and mobile

Many kinds of distributed systems

- Mutlicore, various shared-memory systems
- Internet
- Wireless and mobile
- cloud computing,

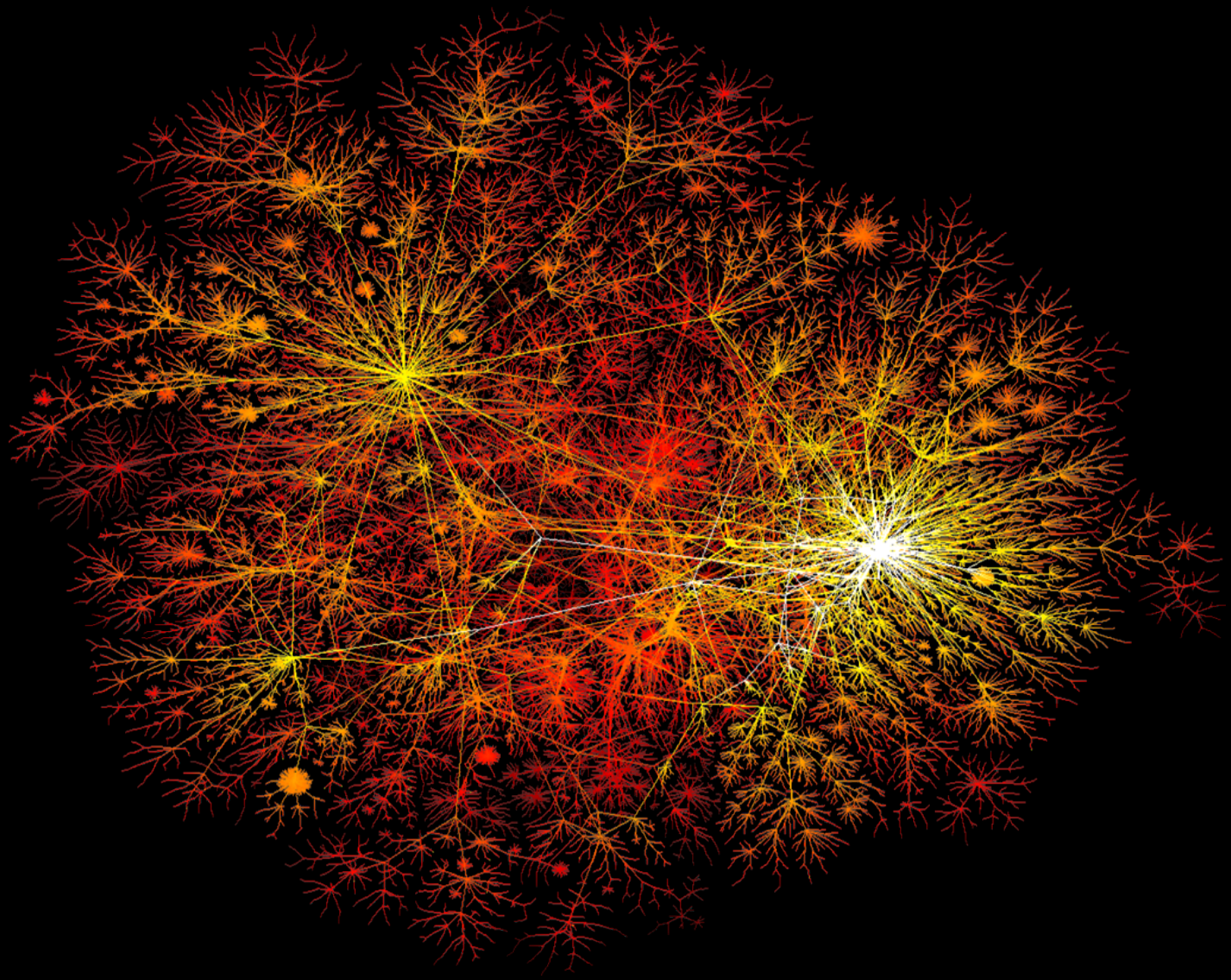
Many kinds of distributed systems

- Mutlicore, various shared-memory systems
- Internet
- Wireless and mobile
- cloud computing,
- Robots,

Many kinds of distributed systems

- Mutlicore, various shared-memory systems
- Internet
- Wireless and mobile
- cloud computing,
- Robots,
- etc.

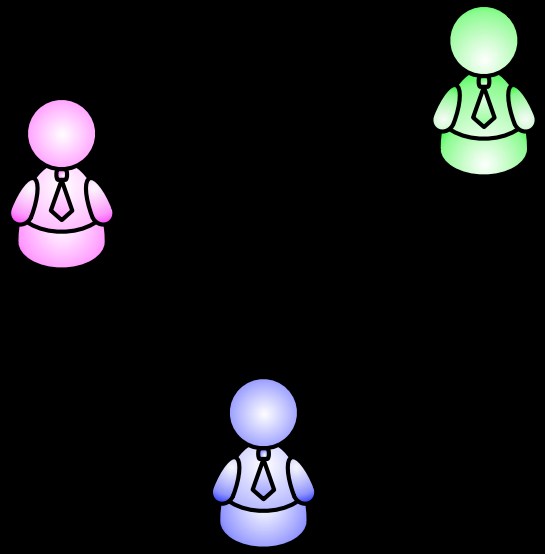
what would a theory of distributed computing be?



Ingredients of distributed computing

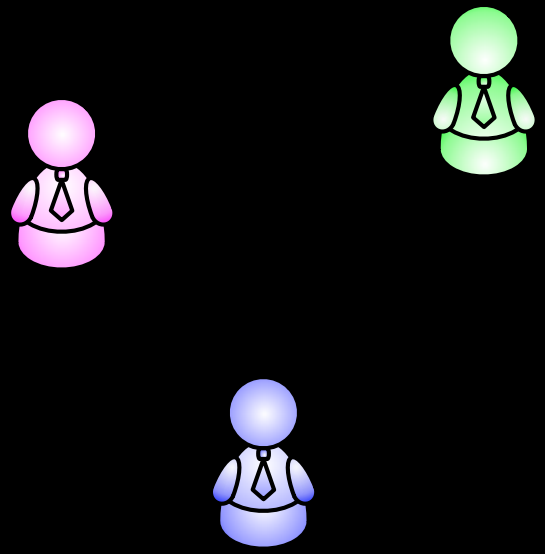
Ingredients of distributed computing

- Processes



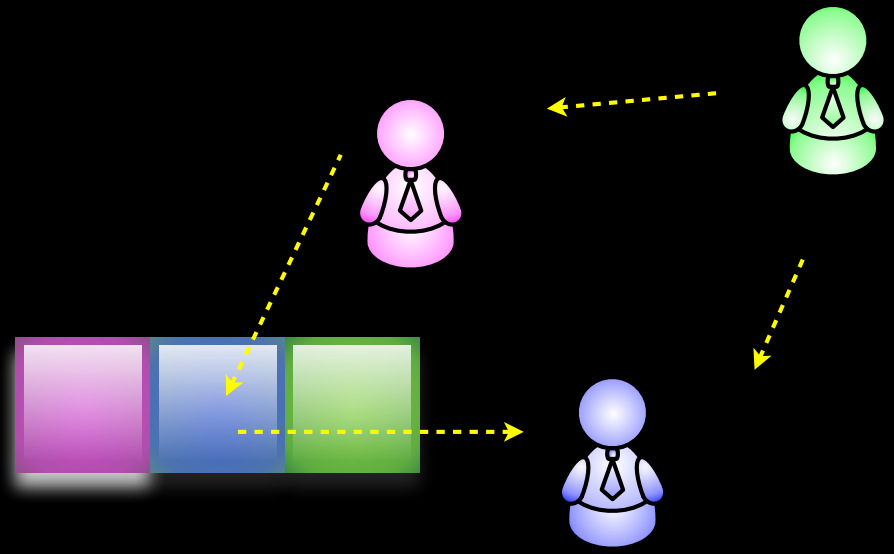
Ingredients of distributed computing

- Processes
- Communication



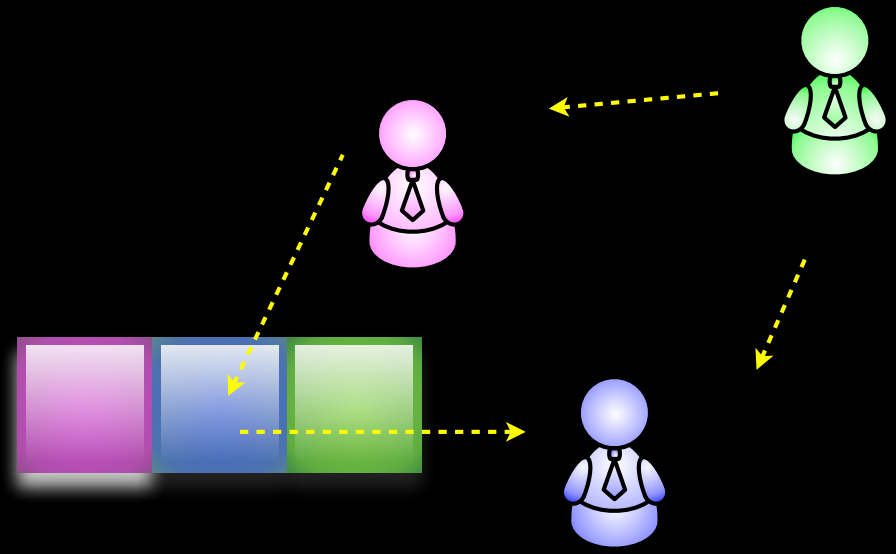
Ingredients of distributed computing

- Processes
- Communication



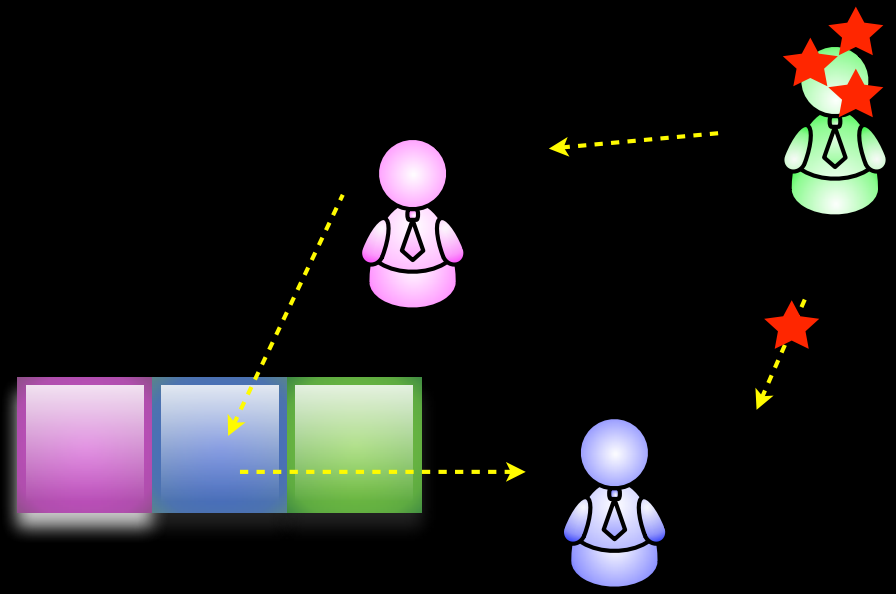
Ingredients of distributed computing

- Processes
- Communication
- Timing



Ingredients of distributed computing

- Processes
- Communication
- Timing
- Failures



Problems

- In seq., functions: one input, one output
- In dist., we consider *tasks*: distributed inputs/outputs, represented as simplexes
- Interested mainly in coordination, local computation power disregarded
- see some examples...

Agreement tasks

Agreement tasks

- consensus: agree on 1 value

Agreement tasks

- consensus: agree on 1 value
- *k*-set agreement: on at most *k* values

Agreement tasks

- consensus: agree on 1 value
- *k*-set agreement: on at most *k* values
- approximate agreement: close to each other, within epsilon

Agreement tasks

- consensus: agree on 1 value
- *k*-set agreement: on at most *k* values
- approximate agreement: close to each other, within epsilon
- Etc

Disagreement tasks

Disagreement tasks

- Leader election: one of the participants

Disagreement tasks

- Leader election: one of the participants
- Binary WSB: not all decide the same value

Disagreement tasks

- Leader election: one of the participants
- Binary WSB: not all decide the same value
- Renaming: on a small name space

Disagreement tasks

- Leader election: one of the participants
- Binary WSB: not all decide the same value
- Renaming: on a small name space
- Etc

Everything is solvable with perfect communication

Everything is solvable with perfect communication

- Exchange inputs, compute locally

Everything is solvable with perfect communication

- Exchange inputs, compute locally
- Even consensus is solvable!

Everything is solvable with perfect communication

- Exchange inputs, compute locally
- Even consensus is solvable!
- Individual machines go beyond Turing

Everything is solvable with perfect communication

- Exchange inputs, compute locally
- Even consensus is solvable!
- Individual machines go beyond Turing
- Ie, we are interested in distributed aspects of computability: uncertainty due to delays and failures

Why some weaker than others?

Why some weaker than others?

- Intuitively, a weaker model has more interleavings

Why some weaker than others?

- Intuitively, a weaker model has more interleavings
- In this sense, stronger models are contained in weaker, intuitively

Stronger objects

Stronger objects

- Over the years many stronger communication objects proposed

Stronger objects

- Over the years many stronger communication objects proposed
- Test&Set, compare&swap, etc

Stronger objects

- Over the years many stronger communication objects proposed
- Test&Set, compare&swap, etc
- And each one with different power:

Stronger objects

- Over the years many stronger communication objects proposed
- Test&Set, compare&swap, etc
- And each one with different power:
- More powerful are more expensive to build

what do we expect of a “universal” model?

what do we expect of a “universal” model?

what do we expect of a "universal" model?

- We want to be able to study (most) distributed computing issues, and then extrapolate to other models

what do we expect of a "universal" model?

- We want to be able to study (most) distributed computing issues, and then extrapolate to other models
- Not the strongest! The strongest can solve everything- no failures, no asynchrony, no communication problems-- no DC issues!

A universal model should

A universal model should

- Be the weakest! Then all DC issues appear; stronger models are contained in it

A universal model should

- Be the weakest! Then all DC issues appear; stronger models are contained in it
- (But not too weak. There are details we want to avoid, or study separately; e.g. routing, partitions, etc)

A “universal” model defined by properties

A “universal” model defined by properties

- Asynchronous- all interleavings

A “universal” model defined by properties

- Asynchronous- all interleavings
- read/write or message passing- basic way of communicating information, direct all-to-all

A "universal" model defined by properties

- Asynchronous- all interleavings
- read/write or message passing- basic way of communicating information, direct all-to-all
- any number of processes can crash: *wait-free*

A "universal" model defined by properties

- Asynchronous- all interleavings
- read/write or message passing- basic way of communicating information, direct all-to-all
- any number of processes can crash: *wait-free*
- crash failures only (more severe have not been studied as much)

Example of a one-round "universal" model

Example of a one-round "universal" model

- a shared array

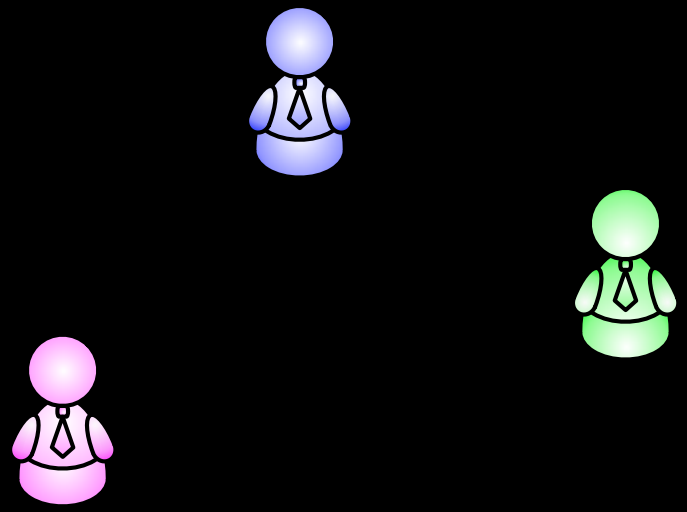
Example of a one-round "universal" model

- a shared array
- Write in your location,

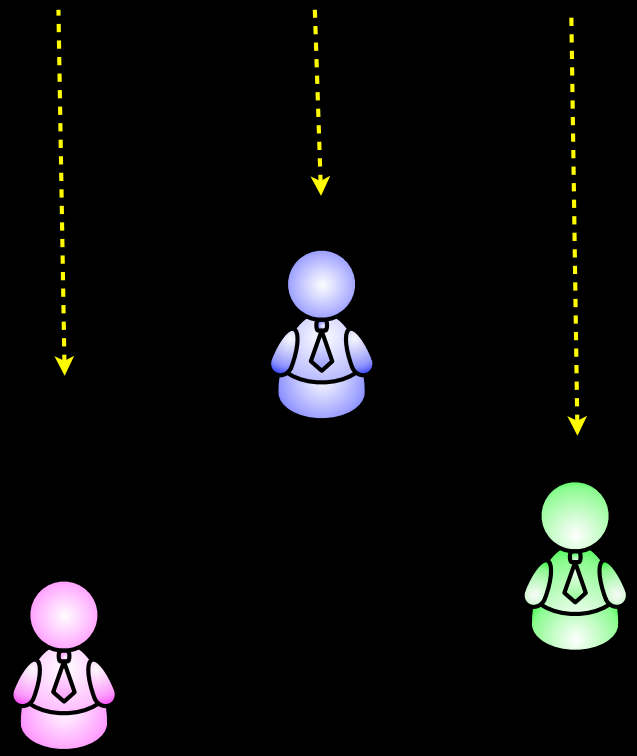
Example of a one-round "universal" model

- a shared array
- Write in your location,
- read all array in an atomic snapshot

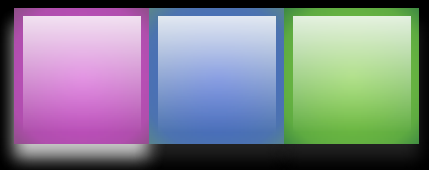
n Processes

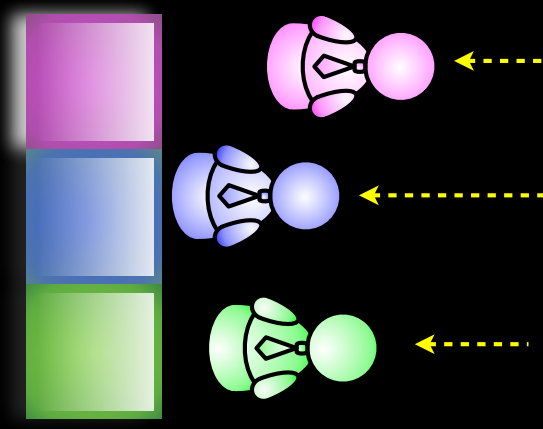


asynchronous

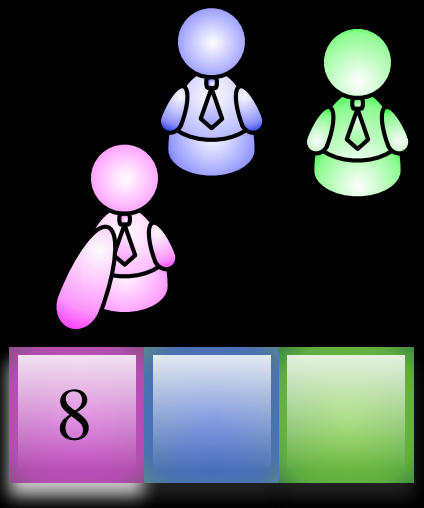


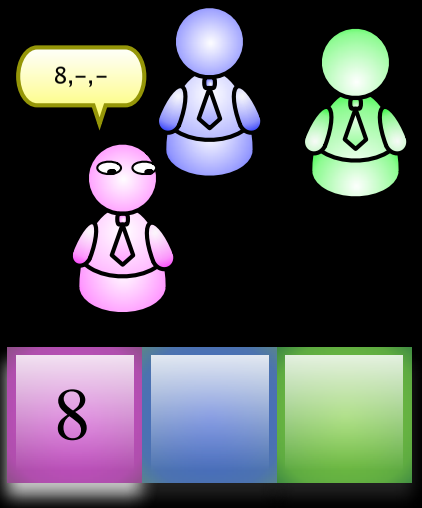
read/write
shared
array

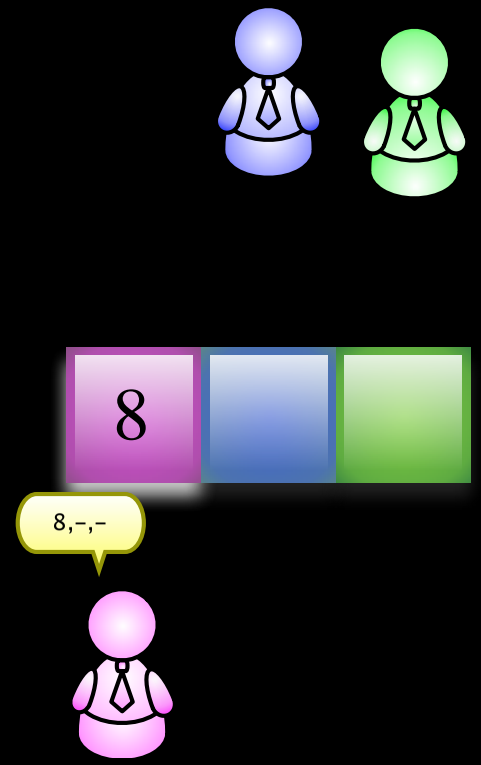


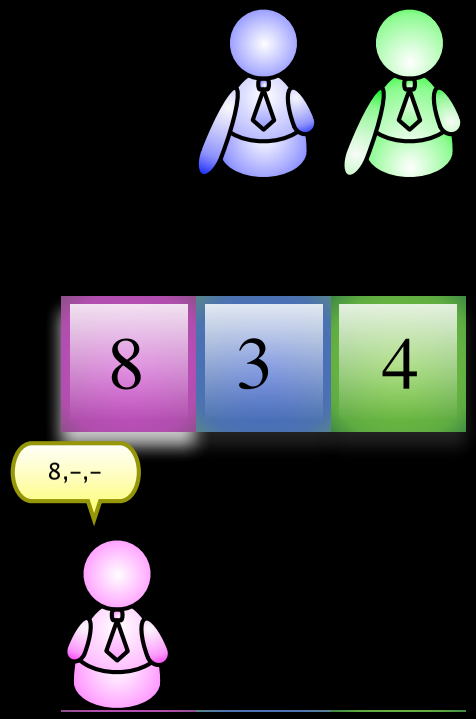


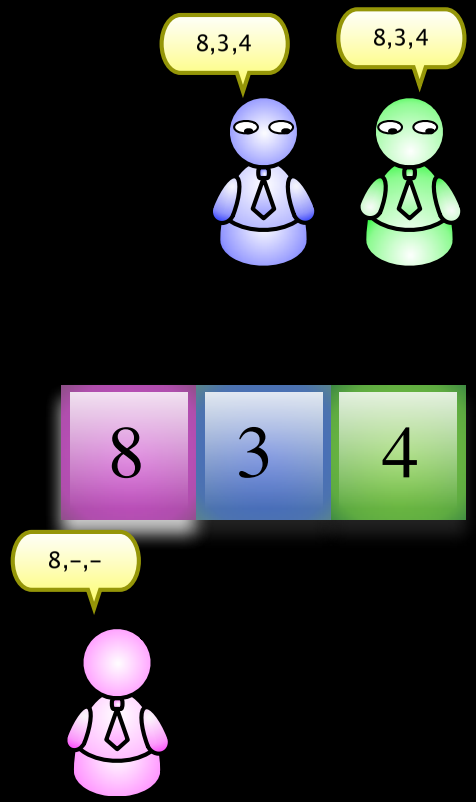
write, then read

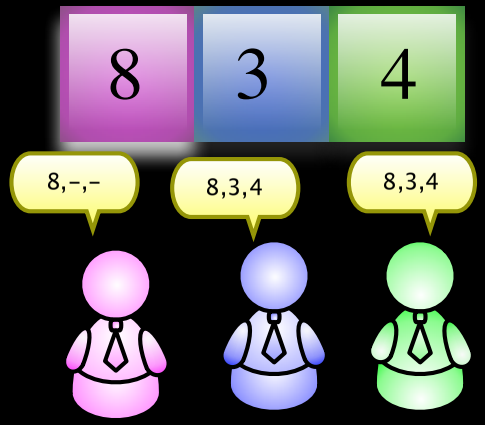



8,–,–
8


8
8,–,–


8
3
4
8,–,–

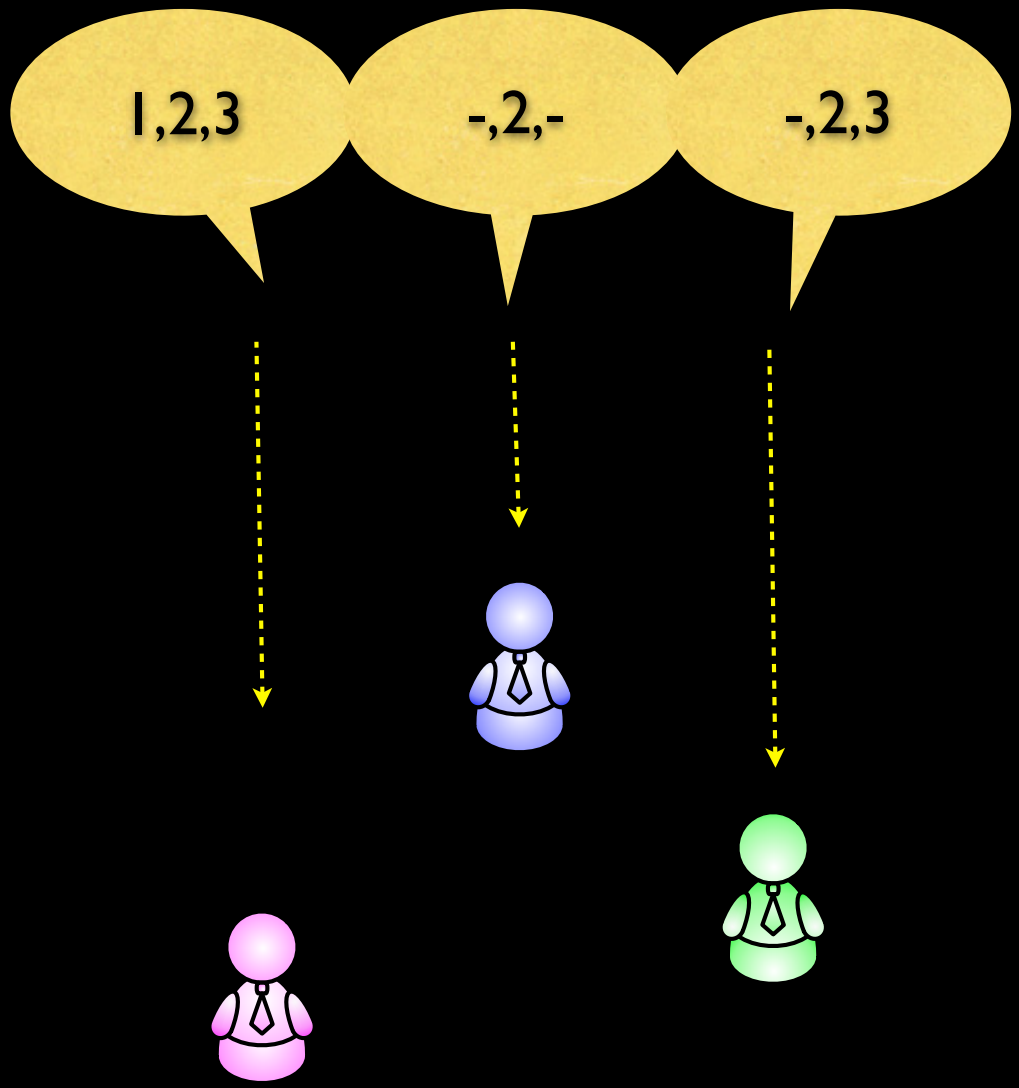

8,3,4
8,3,4
8
3
4
8,–,–


8
3
4
8,–,–
8,3,4
8,3,4

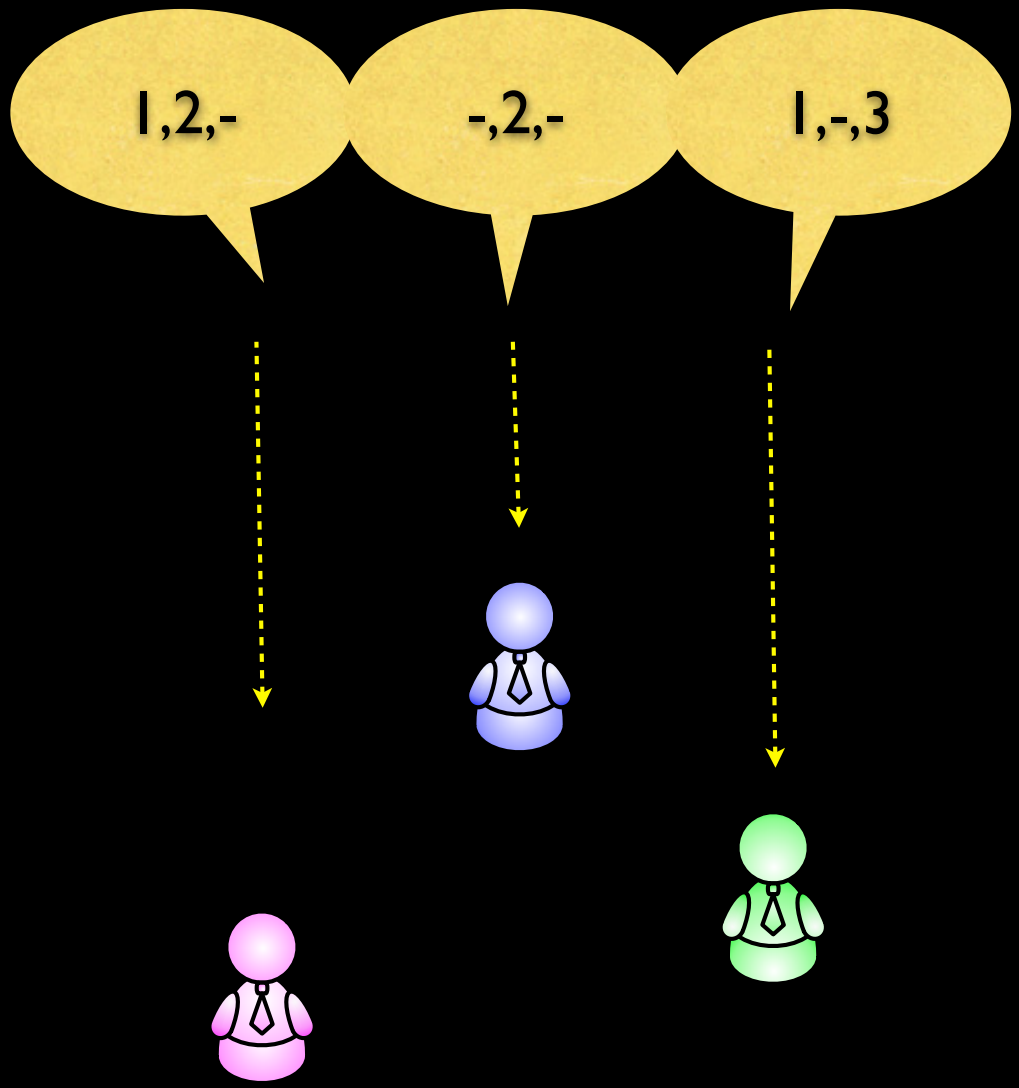
snapshots

- Each process obtains a set of (ids,values)
- the sets can be ordered by containment

ok views

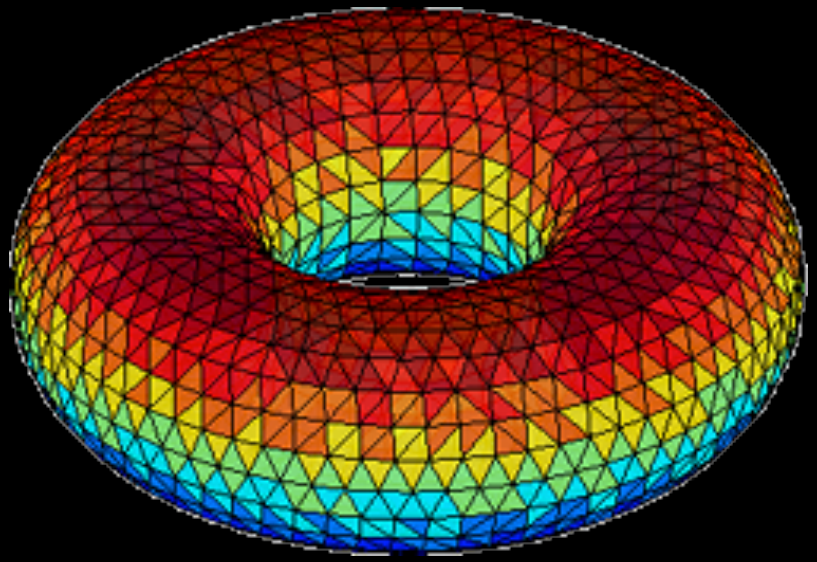


NOT ok views



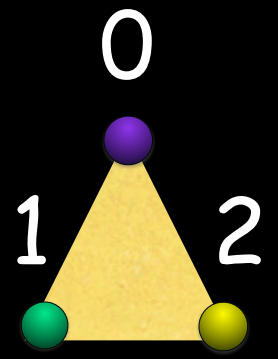
Representation using simplicial complexes

Collection of simplexes closed under containment



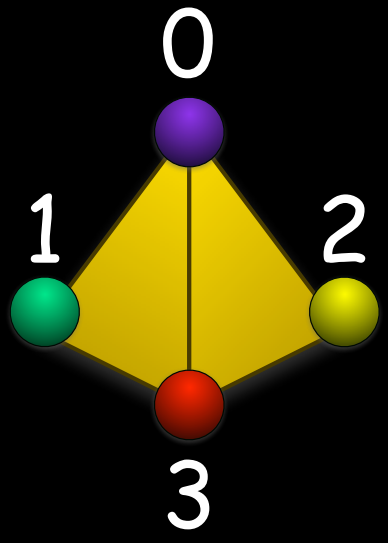
2-dim simplex: a global state for 3 processes

- three local states in some execution
- 2-dimensional simplex
- e.g. inputs 0,1,2



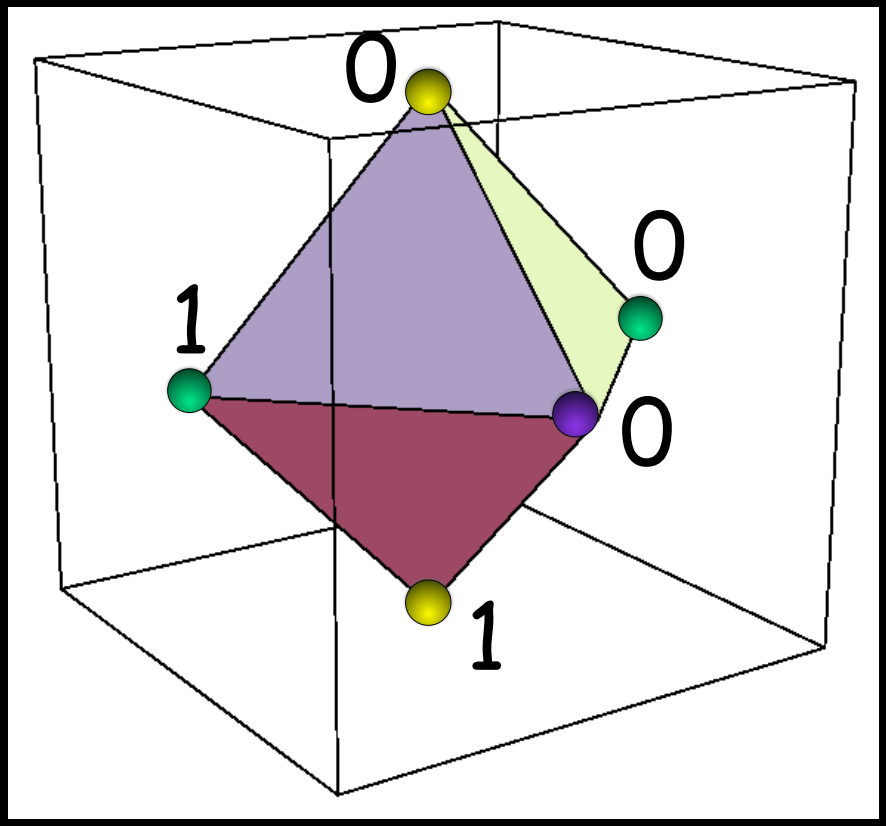
3-dim simplex

- 4 local states in some execution
- 3-dim simplex
- e.g. inputs 0,1,2,3

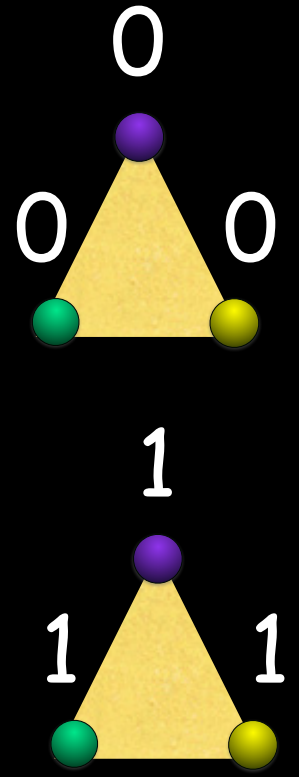


consensus task

3 processes



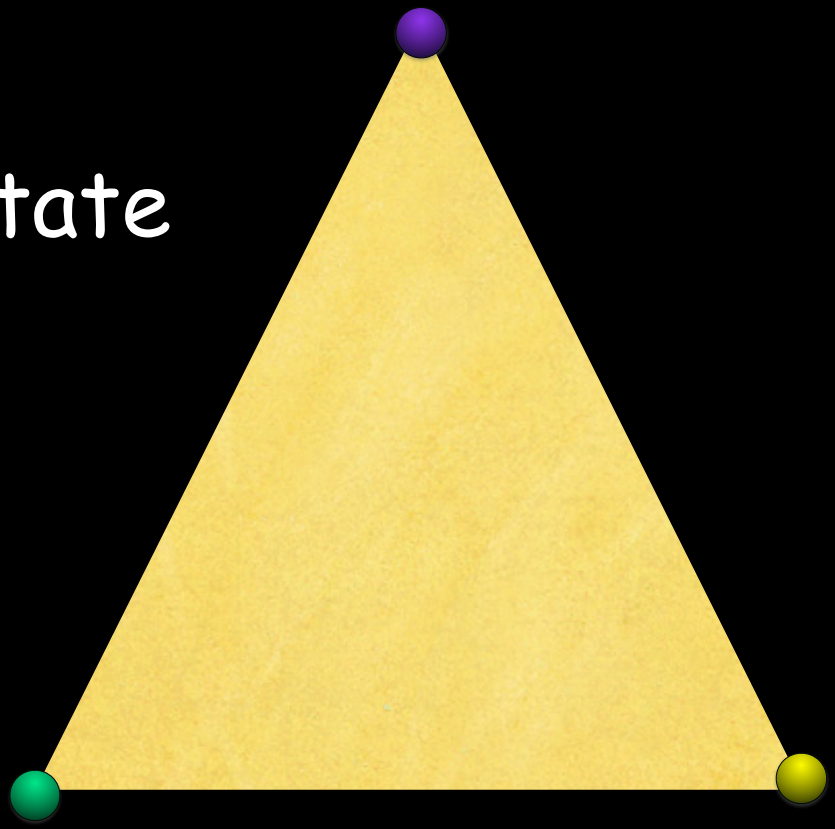
Input Complex



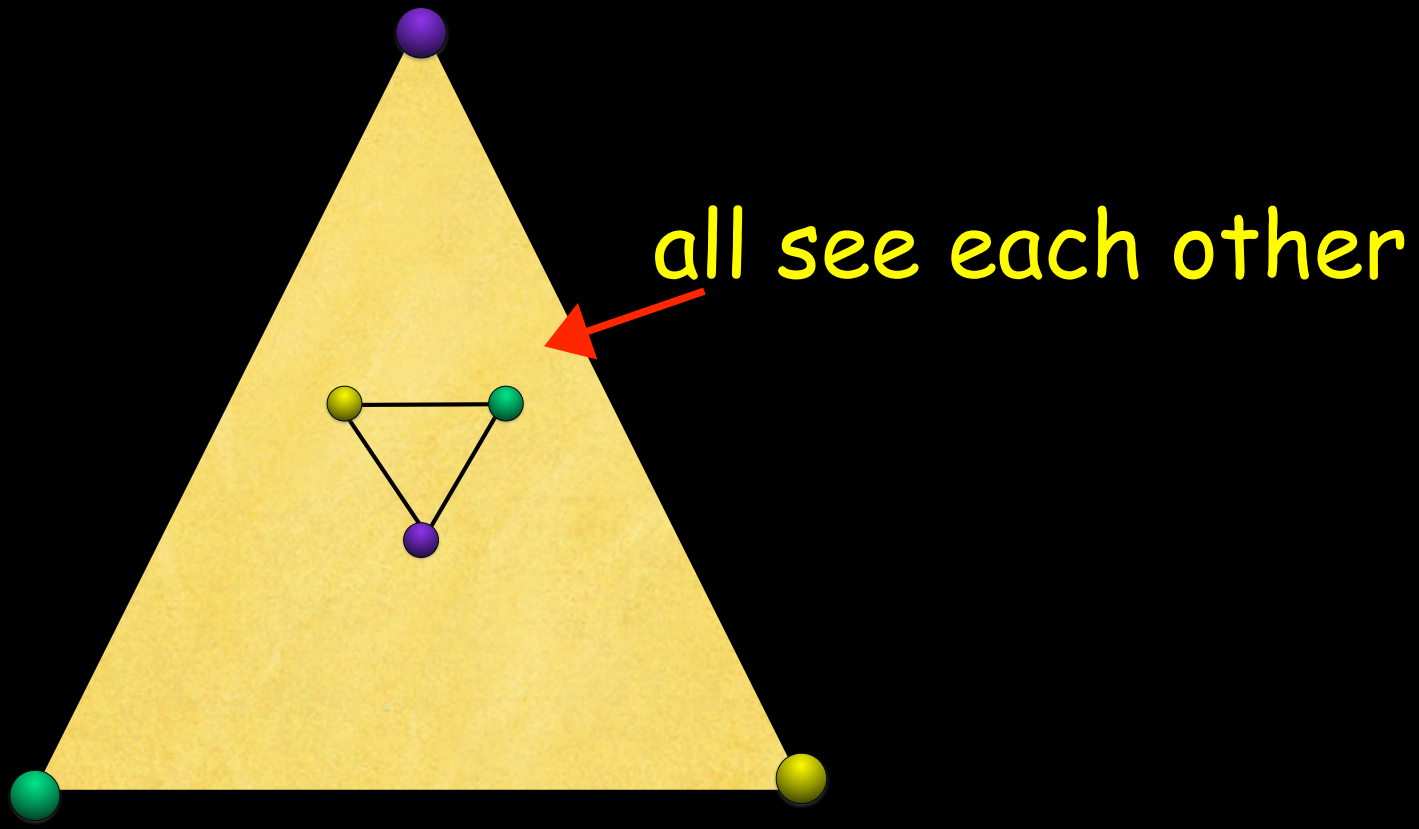
Output Complex

Complex of possible views after one round

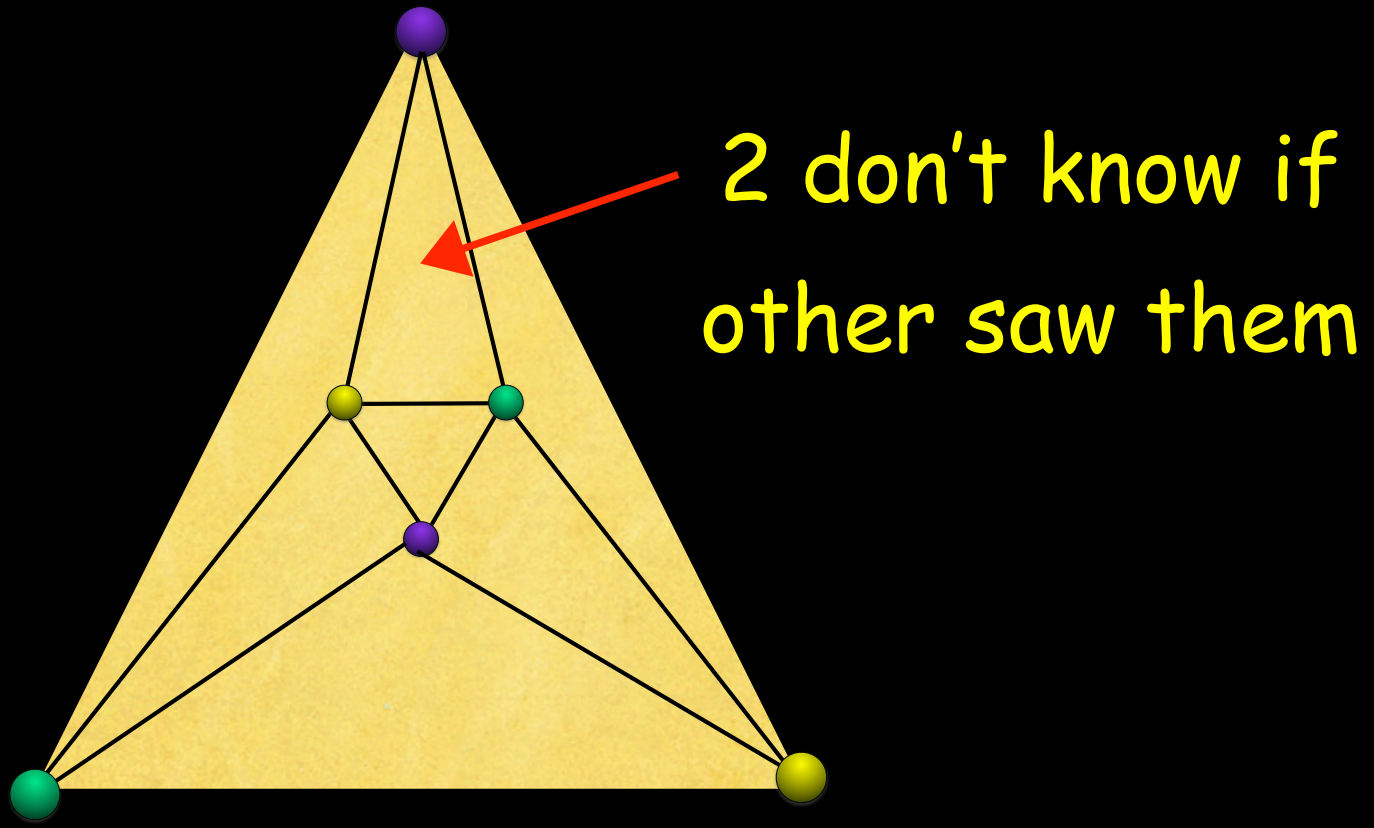
One initial state



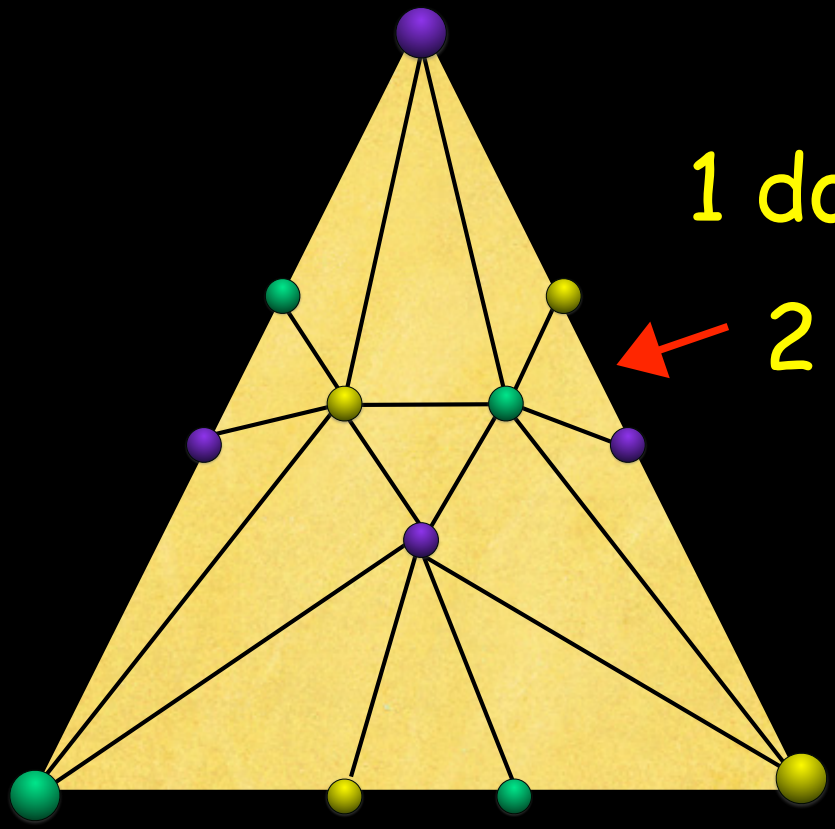
Complex of possible views after one round



Complex of possible views after one round



Complex of possible views after one round



1 doesn't know if

2 other saw it

Wait-free theorem for *n* processes

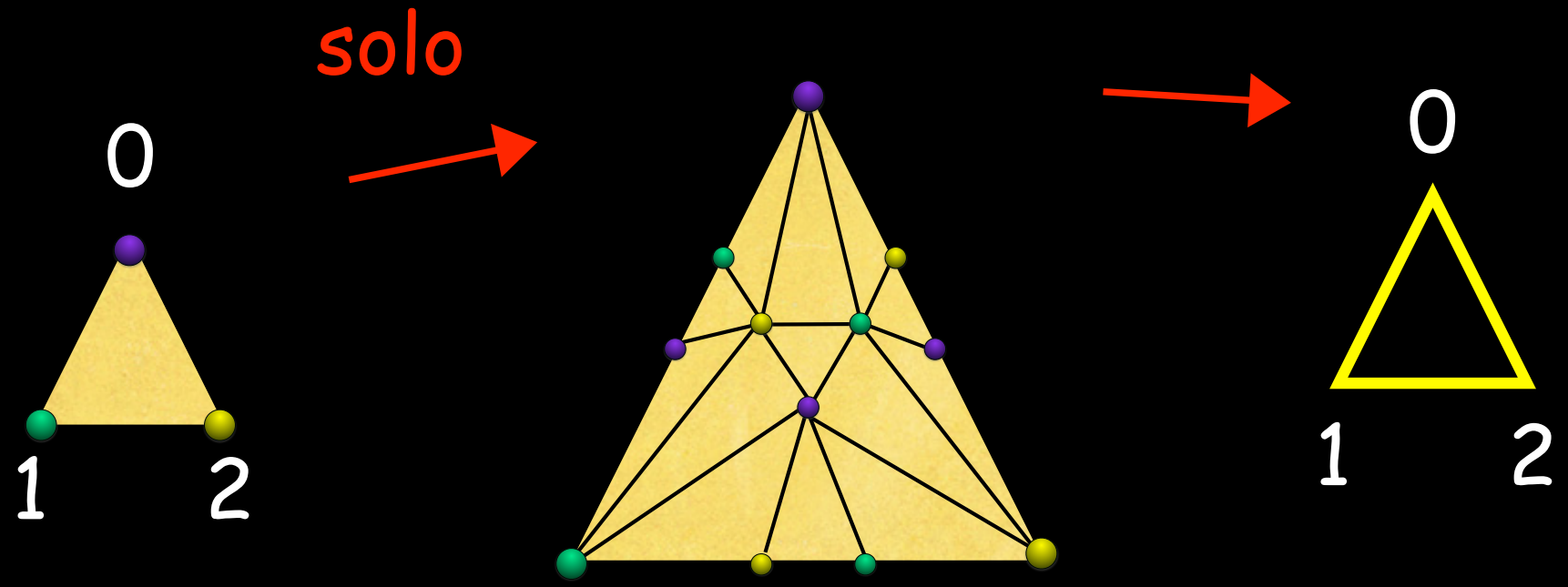
For any "universal" model, the protocol complex after k rounds can be assumed to be

- a subdivision of the input complex

implications in terms of

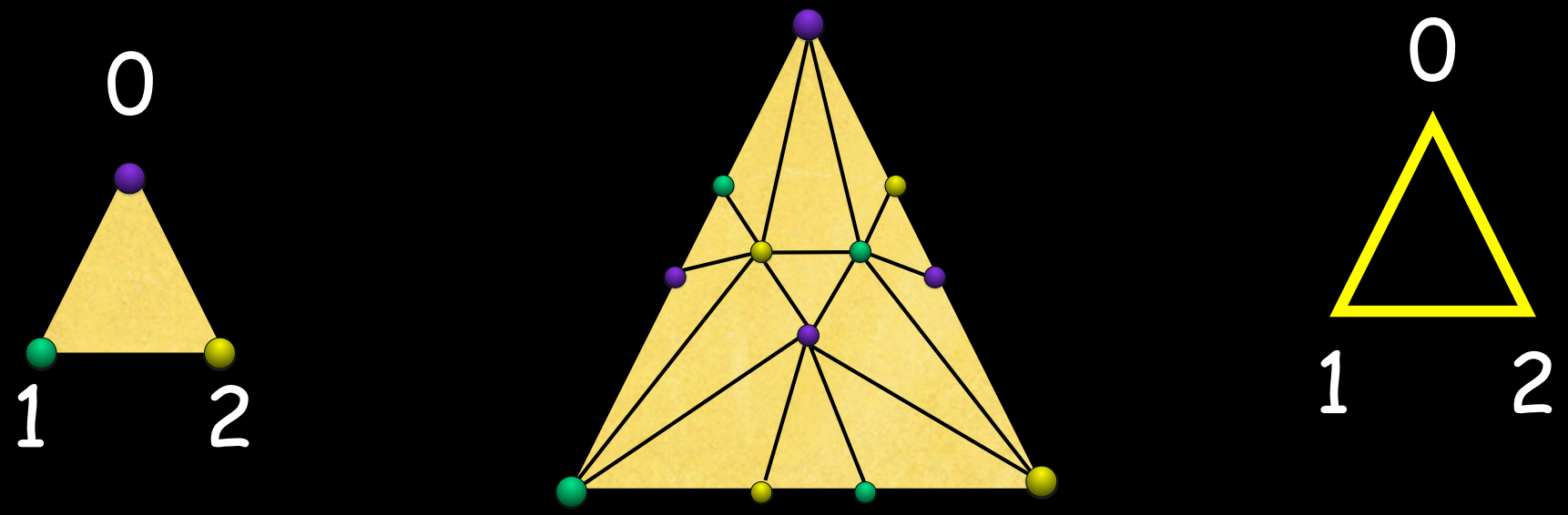
- task solvability
- complexity
- computability

3 cannot agree on 2 values



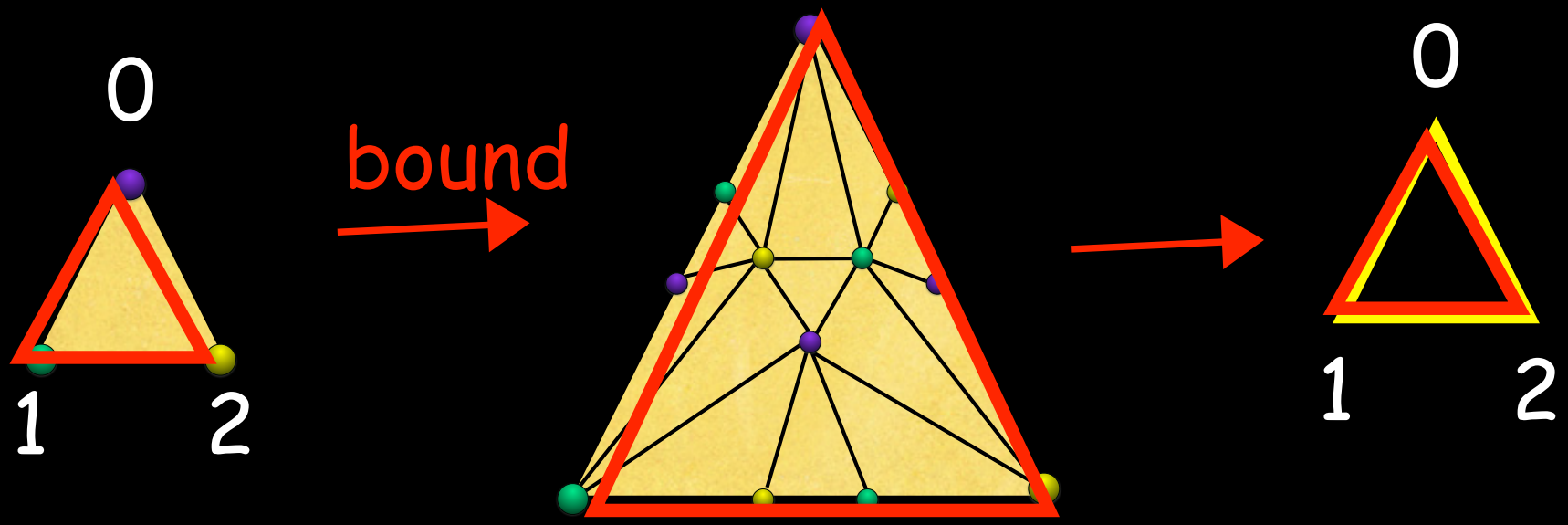
input complex after k round output complex

3 cannot agree on 2 values



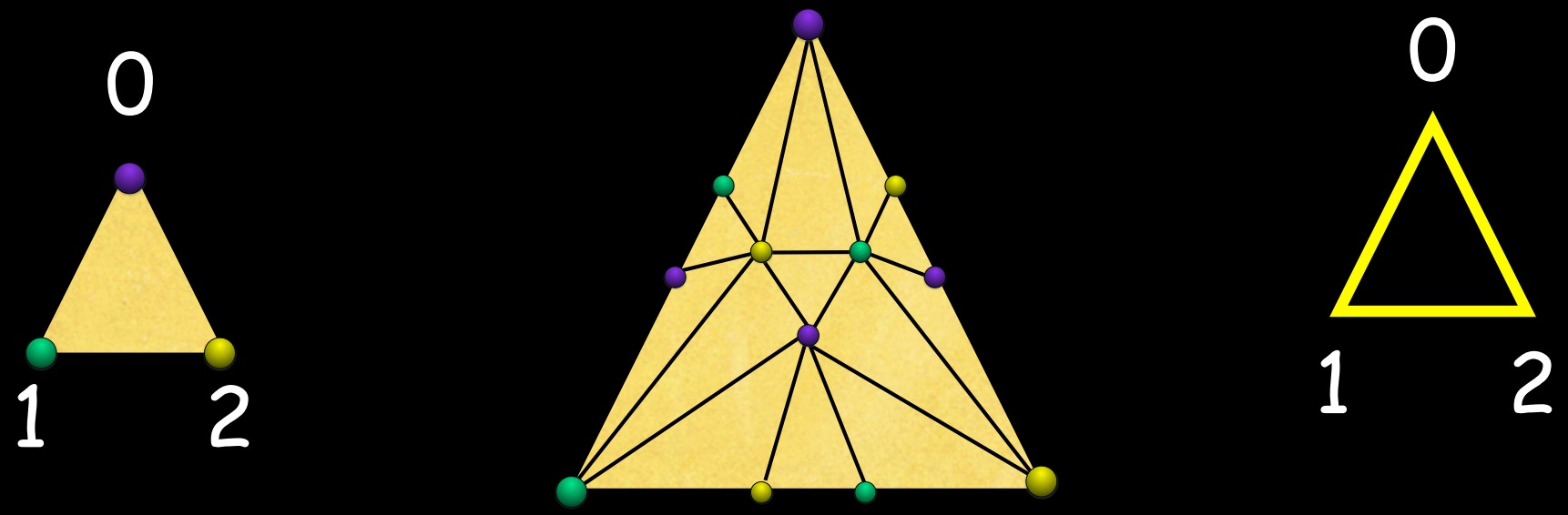
input complex after k round output complex

3 cannot agree on 2 values



input complex after k round output complex

3 cannot agree on 2 values



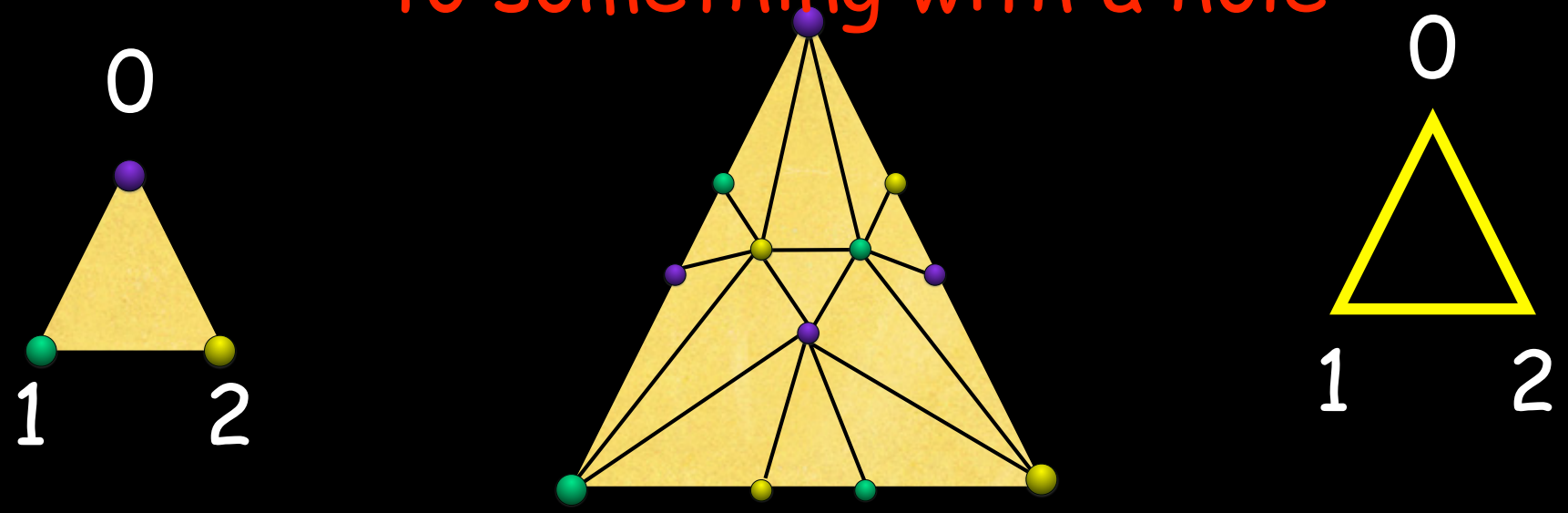
input complex after k round output complex

3 cannot agree on 2 values

Cannot map

something without a hole

to something with a hole



input complex after k round output complex

Conclusions

- Dist Computability is not about TM, but about topology

It is a matter of
perspectives,
of course

But perspectives can be
complicated, they
can evolve and they can
depend on the environment

Dual of Kripke models

Explored with Eric Goubault, Jeremy Ledent and Hans van Ditmarsch

In a series of papers since GANDALF 18

Perspectives Evolve with communication

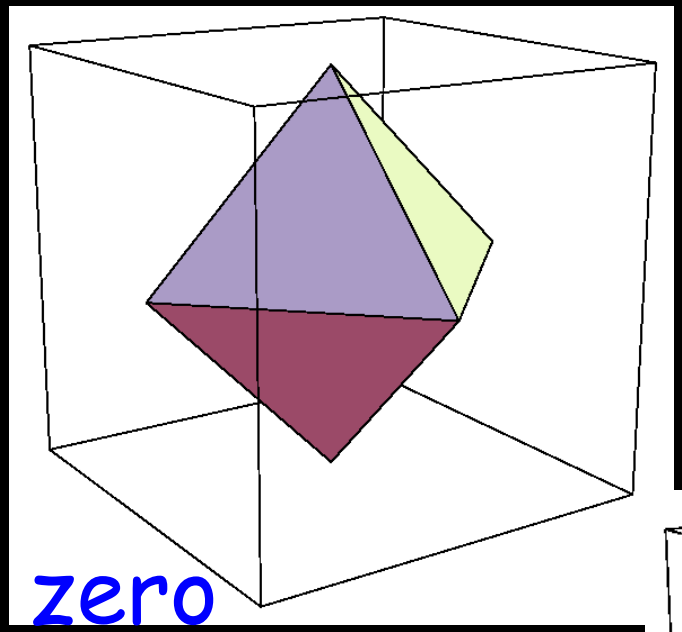
Werner,

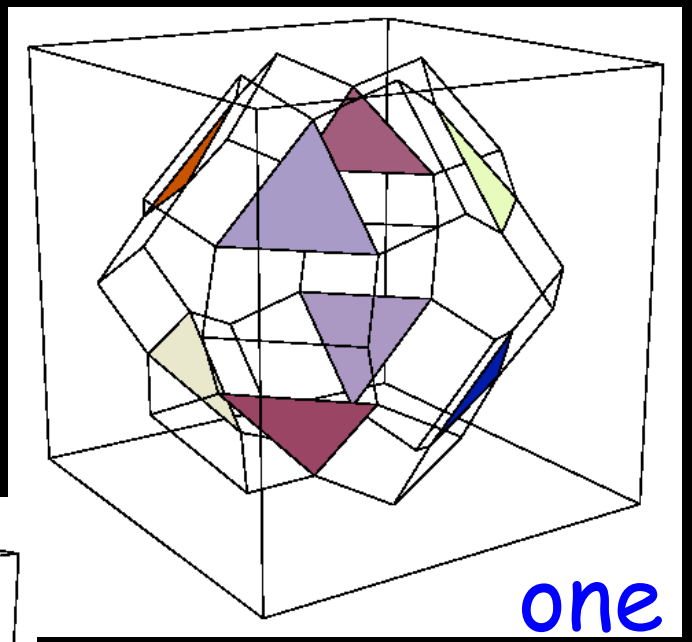
The Talmud Discussion

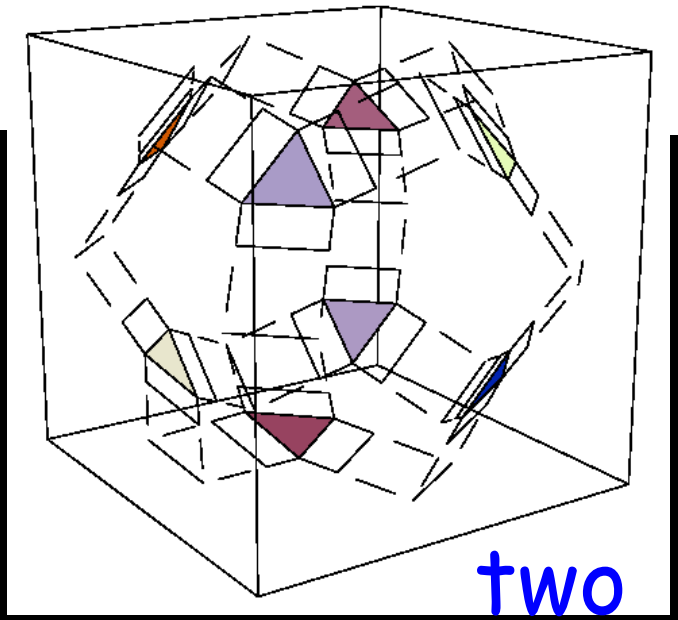
Perspectives evolve differently in different situations

Rashomon, Kurosawa 1950

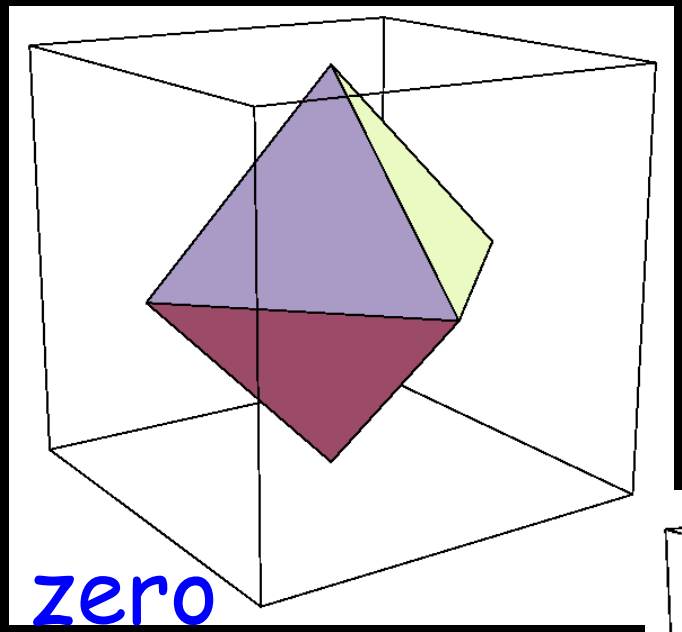
Synchronous message-passing, processes may crash

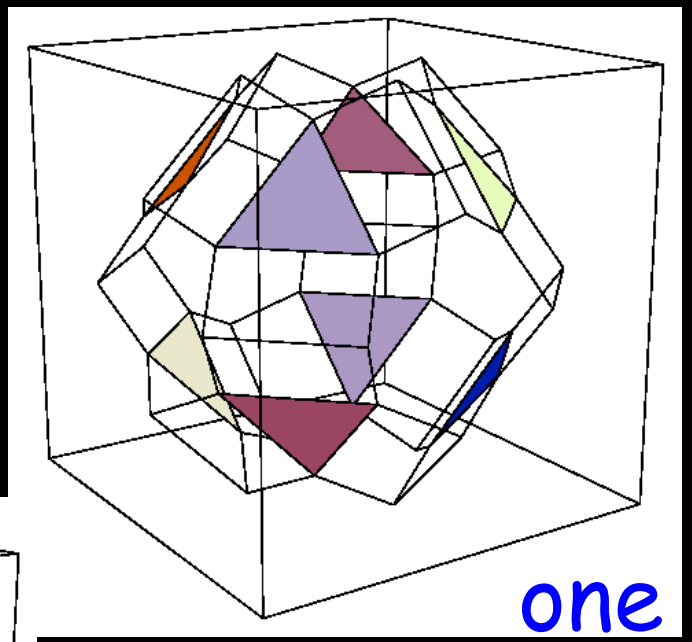


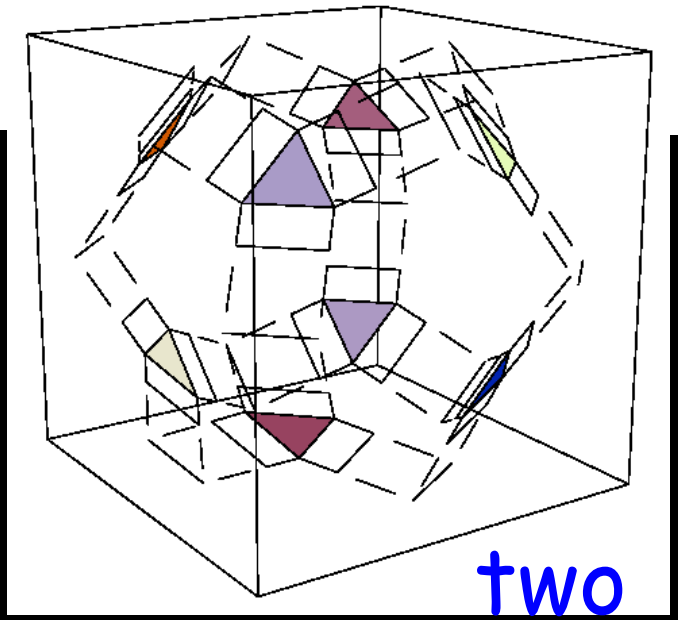




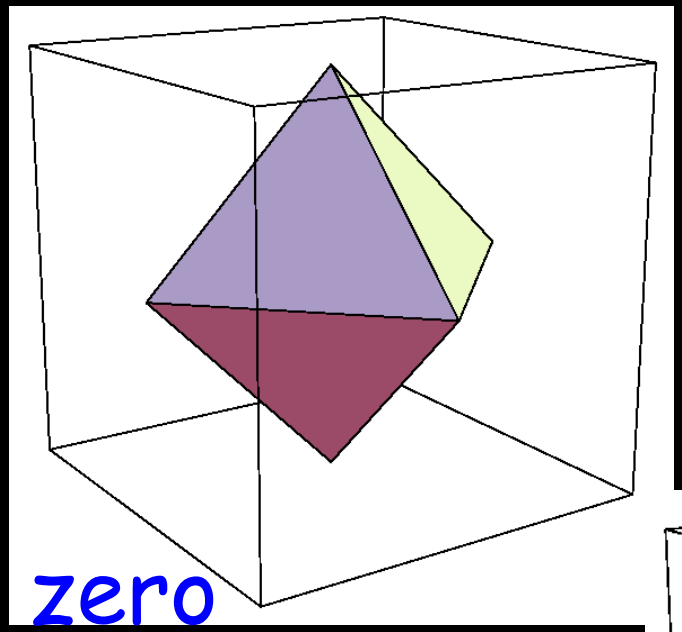
Synchronous message-passing, processes may crash

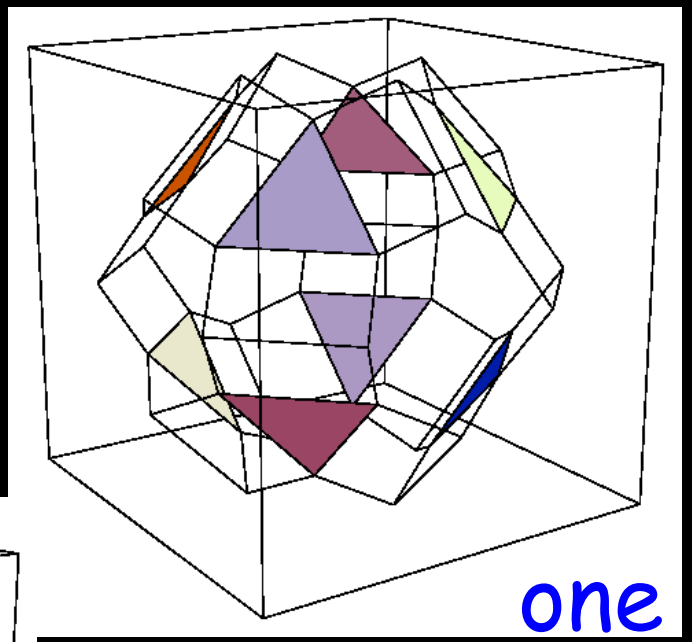


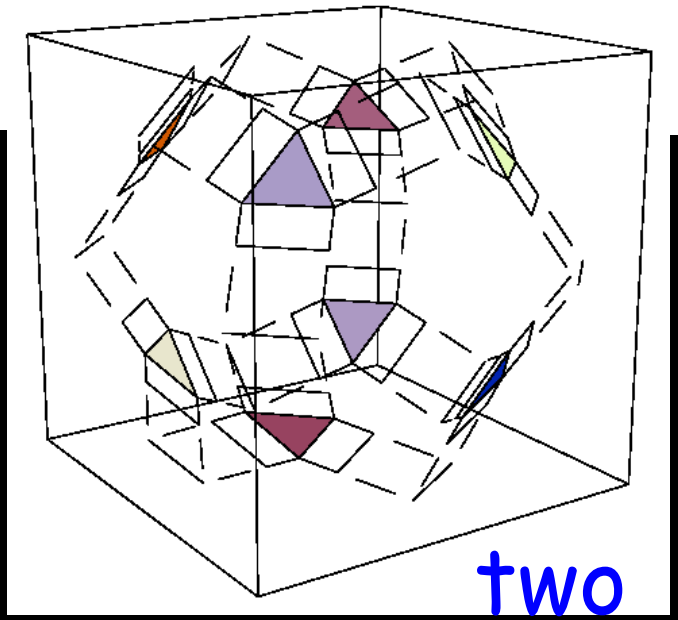




Synchronous message-passing, processes may crash

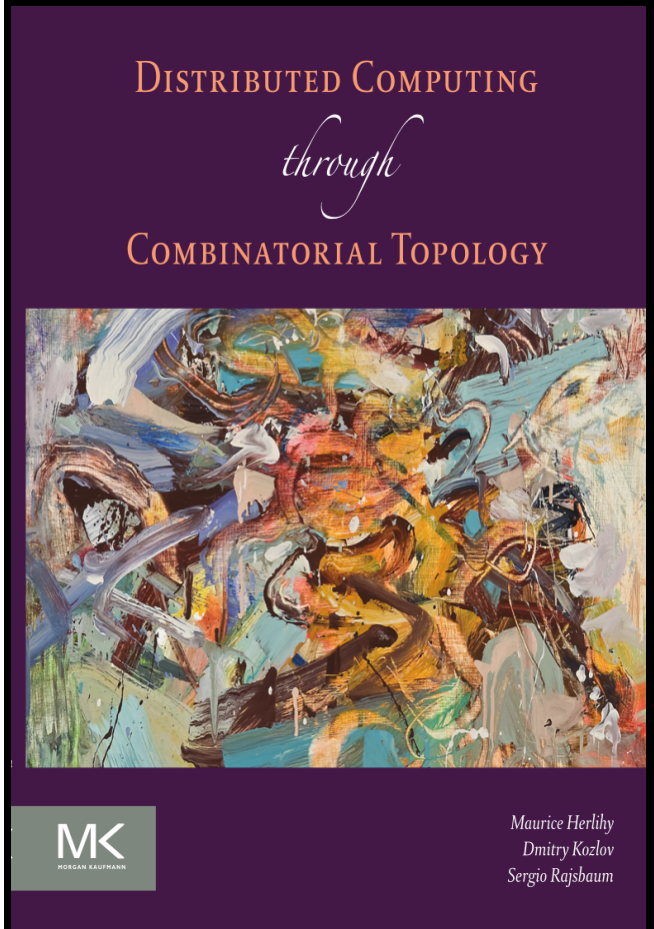






As perspectives

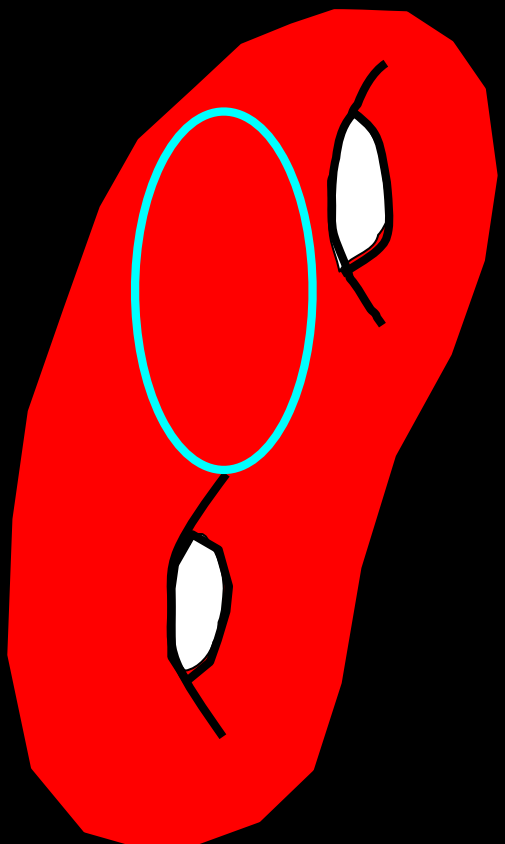
evolve they preserve topological properties !!!



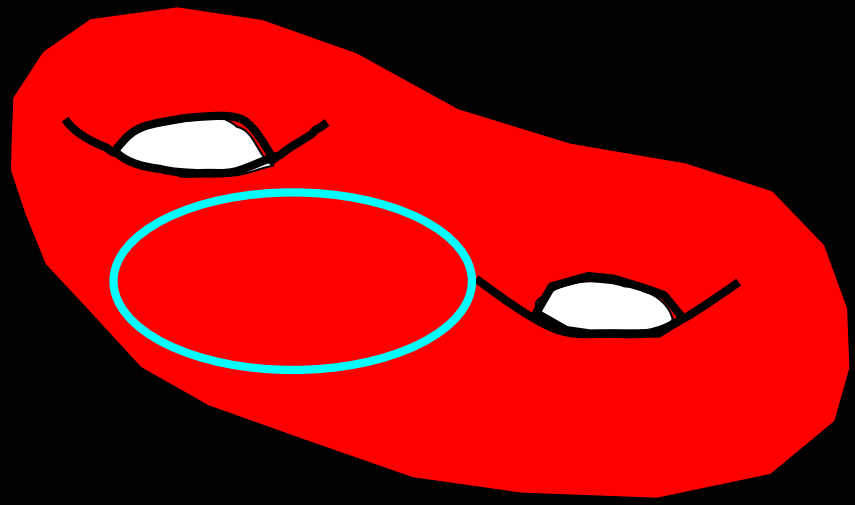
Computability is determined by how well the topology is preserved

And complexity by how fast the refinement can be made

A consequence

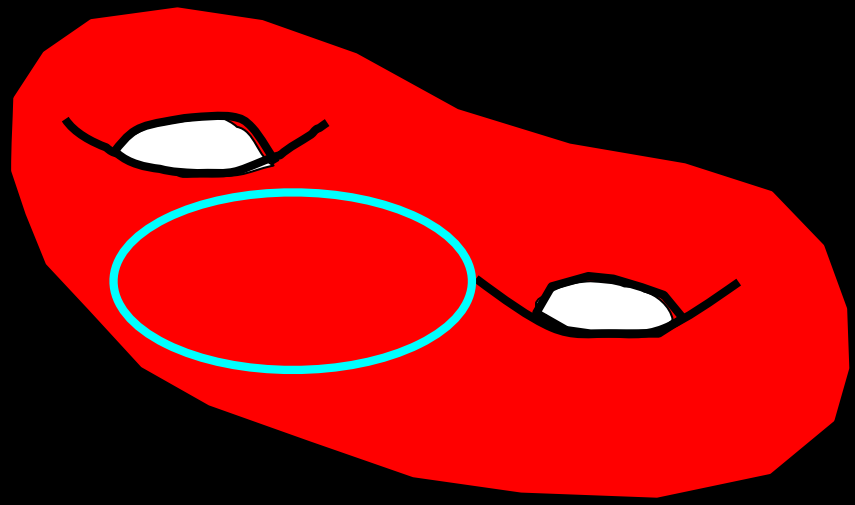


For a distributed computing model, is there an algorithm solving a given problem?



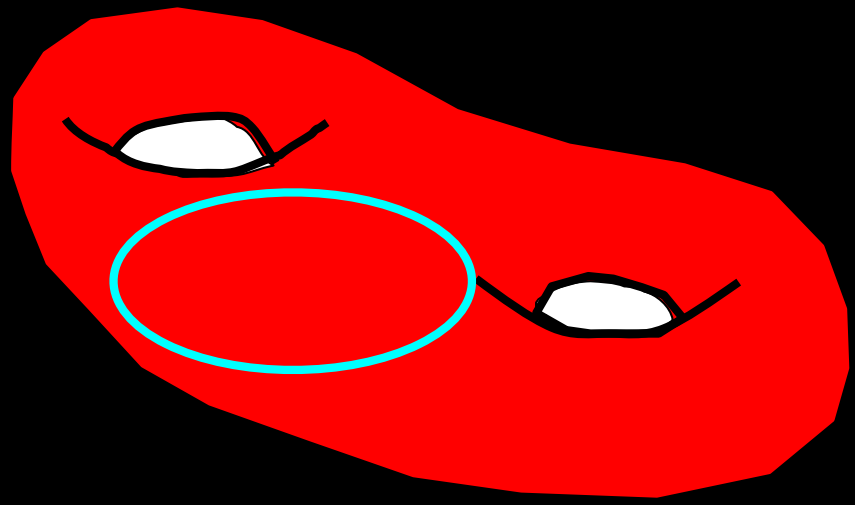
For a distributed computing model, is there an algorithm solving a given problem?

- Not in most models



For a distributed computing model, is there an algorithm solving a given problem?

- Not in most models

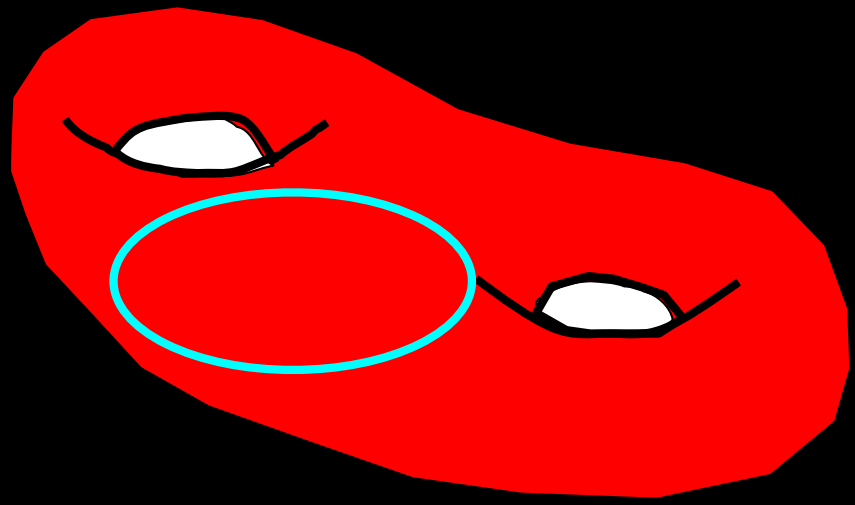


By reduction to a classic topology problem:

Can a given loop be contracted in a complex?

For a distributed computing model, is there an algorithm solving a given problem?

- Not in most models

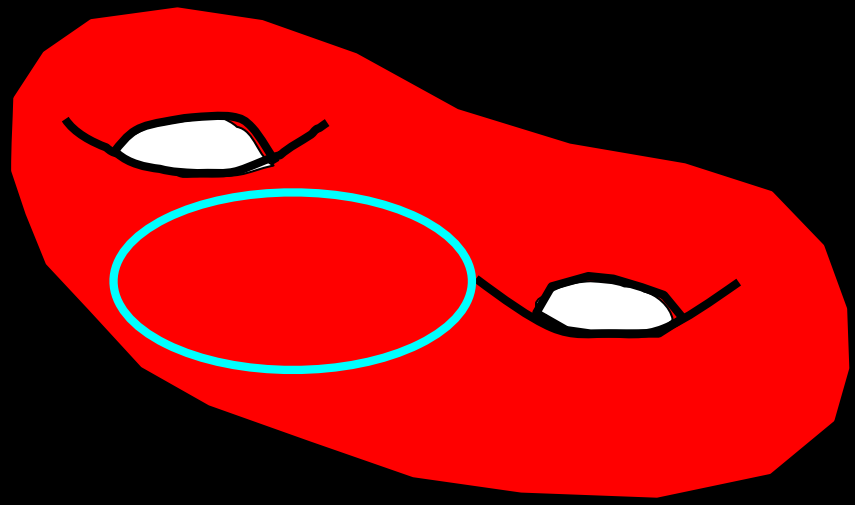


By reduction to a classic topology problem:

Can a given loop be contracted in a complex?

For a distributed computing model, is there an algorithm solving a given problem?

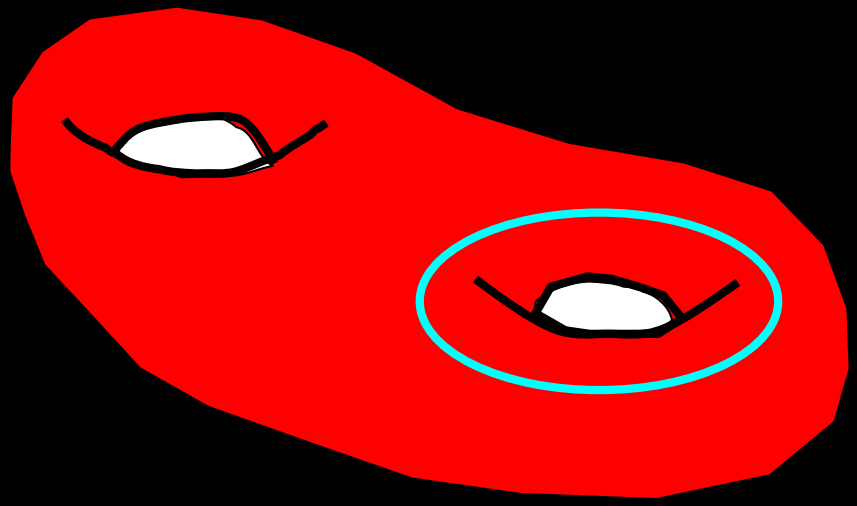
- Not in most models



By reduction to a classic topology problem:

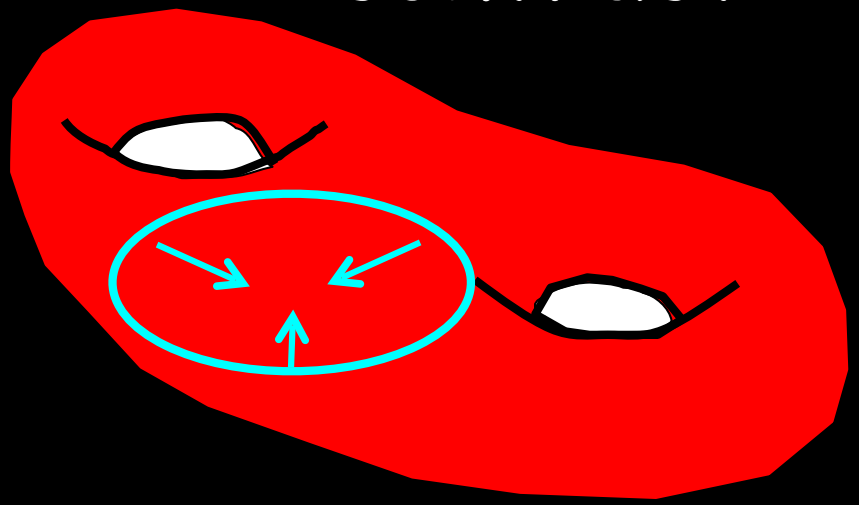
Can a given loop be contracted in a complex?

Contractibility is undecidable



not contractible

contractible



END

Thank you