Topological Structure of Asynchronous Computing II

Maruth Goyal

UT Austin

Spring 2021

Table of Contents

Combining Topology and Computation

Theorem (Asynchronous Computability Theorem)

A decision task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ *has a wait-free protocol using read-write memory if and only if there exists a chromatic subdivision* σ *of* $\mathcal{I}$ *and a color-preserving simplicial map*

$$\mu : \sigma(\mathcal{I}) \to \mathcal{O}$$

such that for each simplex S *in* $\sigma(\mathcal{I})$, $\mu(S) \in \Delta(\mathrm{carrier}(S, \mathcal{I}))$.

- We have all these topological constructions, but how do we embed our decision task to work with these constructions?

Combining Topology and Computation

Theorem (Asynchronous Computability Theorem)

A decision task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ *has a wait-free protocol using read-write memory if and only if there exists a chromatic subdivision* σ *of* $\mathcal{I}$ *and a color-preserving simplicial map*

$$\mu : \sigma(\mathcal{I}) \to \mathcal{O}$$

such that for each simplex S *in* $\sigma(\mathcal{I})$, $\mu(S) \in \Delta(\mathrm{carrier}(S, \mathcal{I}))$.

- We have all these topological constructions, but how do we embed our decision task to work with these constructions?
- Need to (1) Represent the Input/Output sets $\mathcal{I}$ and $\mathcal{O}$ using complexes, and (2) lift Δ to a topological specification.

Combining Topology and Computation

- We first need to introduce a generalization of the geometric simplices from earlier in order to embed our tasks.
- However, conveniently there is provably a correspondence between this generalization and a geometric representation.

Combining Topology and Computation

- We first need to introduce a generalization of the geometric simplices from earlier in order to embed our tasks.
- However, conveniently there is provably a correspondence between this generalization and a geometric representation.

Definition

An *abstract simplex* is simply a non-empty set.

Definition

An *abstract complex* $\mathfrak{K}$ is a collection of abstract simplices closed under containment. i.e., if $S \in \mathfrak{K}$ then so is any face of S.

Combining Topology and Computation

Definition

Let $\vec{I} \in I$ be an input vector. The *input simplex* corresponding to $\vec{I}$, denoted $\mathfrak{I}(I)$, is the abstract colored simplex whose vertices $\langle P_i, v_i \rangle$ correspond to the participating entries in $\vec{I}$, for which $\vec{I}[i] = v_i \neq \perp$. Output simplices defined similarly.

Combining Topology and Computation

Definition

Let $\vec{I} \in I$ be an input vector. The *input simplex* corresponding to $\vec{I}$, denoted $\mathfrak{I}(I)$, is the abstract colored simplex whose vertices $\langle P_i, v_i \rangle$ correspond to the participating entries in $\vec{I}$, for which $\vec{I}[i] = v_i \neq \perp$. Output simplices defined similarly.

Definition

The *input complex* corresponding to I, dneoted by $\mathcal{I}$ is the collection of input simplices $\mathfrak{I}(I)$ corresponding to the input vectors of I. Output complex $\mathfrak{O}$ defined similarly.

Combining Topology and Computation

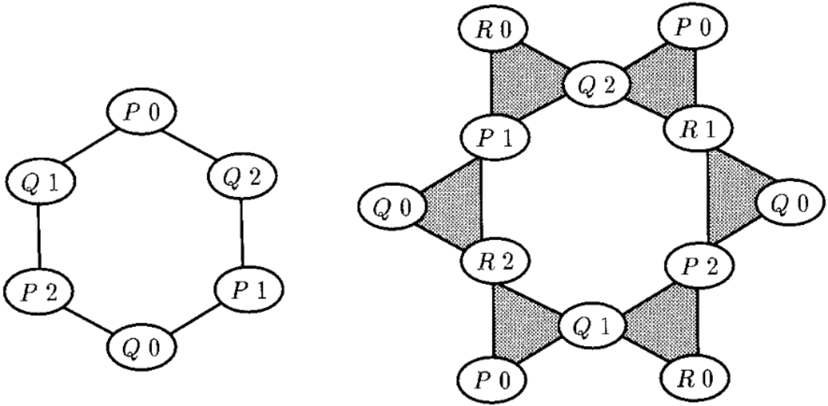

FIG. 8. Some output complexes for the renaming task.

Combining Topology and Computation

Definition

The *topological task specification* corresponding to the task specification Δ, denoted $\Delta \subseteq \mathcal{I} \times \mathcal{O}$, is defined to contain all pairs $(\mathfrak{T}(\vec{I}), \mathfrak{T}(\vec{O}))$ where $(\vec{I}, \vec{O})$ is in the task specification Δ.

Putting it all Together I

Theorem (Asynchronous Computability Theorem)

A decision task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ *has a wait-free protocol using read-write memory if and only if there exists a chromatic subdivision* σ *of* $\mathcal{I}$ *and a color-preserving simplicial map*

$$\mu : \sigma(\mathcal{I}) \to \mathcal{O}$$

such that for each simplex S *in* $\sigma(\mathcal{I})$, $\mu(S) \in \Delta(\mathrm{carrier}(S, \mathcal{I}))$.

My intuition:

- Last condition $\mu(S) \in \Delta(\mathrm{carrier}(S, \mathcal{I}))$ enforces that μ is mapping to valid output simplexes (i.e., protocol actually solves the task)
- The coloring enforces some notion of "independence" among tasks as desired in a wait-free protocol

Putting it all Together II

- The subdivision allows considering more fine-grained / intermediate states of processors, and the color-preserving map says that these states can be mapped to a valid output state which still preserves that independence.

Combining Topology and Computation

- To make the notion of subdivisions more concrete, it is helpful to introduce the notion of a "protocol complex", $\mathcal{P}$.

Combining Topology and Computation

- To make the notion of subdivisions more concrete, it is helpful to introduce the notion of a "protocol complex", $\mathcal{P}$.
- In particular, Consider the input complex $\mathcal{I}$, but now with additional vertices corresponding to the local states of different processors after execution of the protocol, before it finally outputs something.

Combining Topology and Computation

- To make the notion of subdivisions more concrete, it is helpful to introduce the notion of a "protocol complex", $\mathcal{P}$.
- In particular, Consider the input complex $\mathcal{I}$, but now with additional vertices corresponding to the local states of different processors after execution of the protocol, before it finally outputs something.
- This subdivision is called the "protocol complex" $\mathcal{P}$.

Combining Topology and Computation

- To make the notion of subdivisions more concrete, it is helpful to introduce the notion of a "protocol complex", $\mathcal{P}$.
- In particular, Consider the input complex $\mathcal{I}$, but now with additional vertices corresponding to the local states of different processors after execution of the protocol, before it finally outputs something.
- This subdivision is called the "protocol complex" $\mathcal{P}$.
- There is then a "decision map" $\delta : \mathcal{P} \to \mathcal{O}$ which maps vertices of this protocol complex to vertices in the output complex.

Combining Topology and Computation

- To make the notion of subdivisions more concrete, it is helpful to introduce the notion of a "protocol complex", $\mathcal{P}$.
- In particular, Consider the input complex $\mathcal{I}$, but now with additional vertices corresponding to the local states of different processors after execution of the protocol, before it finally outputs something.
- This subdivision is called the "protocol complex" $\mathcal{P}$.
- There is then a "decision map" $\delta : \mathcal{P} \to \mathcal{O}$ which maps vertices of this protocol complex to vertices in the output complex.
- Intuitively, a process uses its local state to decide upon its output.

Combining Topology and Computation

- To make the notion of subdivisions more concrete, it is helpful to introduce the notion of a "protocol complex", $\mathcal{P}$.
- In particular, Consider the input complex $\mathcal{I}$, but now with additional vertices corresponding to the local states of different processors after execution of the protocol, before it finally outputs something.
- This subdivision is called the "protocol complex" $\mathcal{P}$.
- There is then a "decision map" $\delta : \mathcal{P} \to \mathcal{O}$ which maps vertices of this protocol complex to vertices in the output complex.
- Intuitively, a process uses its local state to decide upon its output.
- Moreover, based on how we defined the decision task Δ and output complex $\mathcal{O}$, it is also necessary that the simplex corresponding to the processor states in $\mathcal{P}$ be mapped to an appropriate simplex in $\mathcal{O}$, thus δ is simplicial.

Combining Topology and Computation

- Consider protocol where P and Q write their private values p, q resp. ,and then read the entire array.
- The initial state of the array is $(\perp, \perp)$

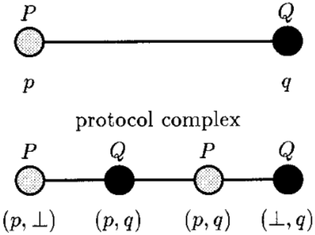

FIG. 14. Simplicial representation of a one-round execution.

Table of Contents

Applications of Asynchronous Computability Theorem

- We will look at 2 examples of applying the main theorem, specifically to
    1. Binary consensus
    2. k-set agreement (generalized consensus)

Binary Consensus Setup

- Suppose we have $n+1$ processors, each assigned a binary value.

Binary Consensus Setup

- Suppose we have $n+1$ processors, each assigned a binary value.
- The task is for all processors to decide on and output the same binary value, such that this value is necessarily the input of at least one processor.

$\vec{I}$	$\Delta(\vec{I})$
$(0, \perp)$	$(0, \perp)$
$(1, \perp)$	$(1, \perp)$
$(0, 0)$	$(0, 0)$
$(0, 1)$	$(0, 0)$, $(1, 1)$

Binary Consensus Setup

- Suppose we have $n+1$ processors, each assigned a binary value.
- The task is for all processors to decide on and output the same binary value, such that this value is necessarily the input of at least one processor.
- This results in an input complex where every vertex has 2^n neighbors, corresponding to different binary assignments. Moreover, this complex is connected. This is called the Binary n-sphere, $\mathscr{B}^n$.

$\vec{I}$	$\Delta(\vec{I})$
$(0, \perp)$	$(0, \perp)$
$(1, \perp)$	$(1, \perp)$
$(0, 0)$	$(0, 0)$
$(0, 1)$	$(0, 0)$, $(1, 1)$

Binary Consensus Setup

- Suppose we have $n+1$ processors, each assigned a binary value.
- The task is for all processors to decide on and output the same binary value, such that this value is necessarily the input of at least one processor.
- This results in an input complex where every vertex has 2^n neighbors, corresponding to different binary assignments. Moreover, this complex is connected. This is called the Binary n-sphere, $\mathscr{B}^n$.
- The output complex is disconnected, containing just 2 lines corresponding to the choice made by all processors.

$\vec{I}$	$\Delta(\vec{I})$
$(0, \perp)$	$(0, \perp)$
$(1, \perp)$	$(1, \perp)$
$(0, 0)$	$(0, 0)$
$(0, 1)$	$(0, 0)$, $(1, 1)$

Binary Consensus Setup

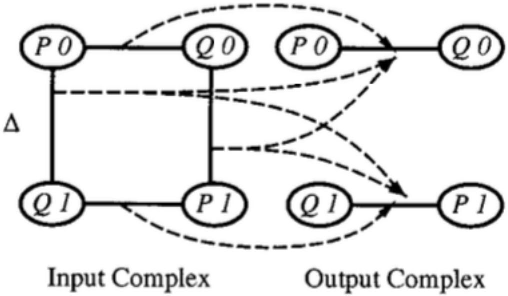

FIG. 17. Simplicial complexes for 2-process consensus.

Binary Consensus Proof

Theorem (Asynchronous Computability Theorem)

A decision task $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ has a wait-free protocol using read-write memory if and only if there exists a chromatic subdivision σ of $\mathcal{I}$ and a color-preserving simplicial map

$$\mu : \sigma(\mathcal{I}) \to \mathcal{O}$$

such that for each simplex S in $\sigma(\mathcal{I})$, $\mu(S) \in \Delta(\text{carrier}(S, \mathcal{I}))$.

- It is sufficient to show that there can be no simplicial map from the input complex to the output complex for the task.
- We will exploit the property that input complex is connected

Binary Consensus Proof

Claim

There is no wait-free protocol for binary consensus.

Proof.

1. We know $\mathcal{I}$ is connected, and hence so is any subdivision $\sigma(\mathcal{I})$.

Binary Consensus Proof

Claim

There is no wait-free protocol for binary consensus.

Proof.

1. We know $\mathcal{I}$ is connected, and hence so is any subdivision $\sigma(\mathcal{I})$.
2. Now, consider the decision map $\mu : \sigma(\mathcal{I}) \to \mathcal{O}$. Let I_1 denote the simplex where all processors receive 1.

Binary Consensus Proof

Claim

There is no wait-free protocol for binary consensus.

Proof.

1. We know $\mathcal{I}$ is connected, and hence so is any subdivision $\sigma(\mathcal{I})$.
2. Now, consider the decision map $\mu : \sigma(\mathcal{I}) \to \mathcal{O}$. Let I_1 denote the simplex where all processors receive 1.
3. Then, $\Delta(I_1) = O_1$. Thus, if P_1 denotes the vertex in $\sigma(\mathcal{I})$ corresponding to processor P receiving 1, it must be the case that $\mu(P_1) = P_1$.

Binary Consensus Proof

Claim

There is no wait-free protocol for binary consensus.

Proof.

1. We know $\mathcal{I}$ is connected, and hence so is any subdivision $\sigma(\mathcal{I})$.
2. Now, consider the decision map $\mu : \sigma(\mathcal{I}) \to \mathcal{O}$. Let I_1 denote the simplex where all processors receive 1.
3. Then, $\Delta(I_1) = O_1$. Thus, if P_1 denotes the vertex in $\sigma(\mathcal{I})$ corresponding to processor P receiving 1, it must be the case that $\mu(P_1) = P_1$.
4. By identical reasoning, for another processor Q, it must be the case that $\mu(Q_0) = Q_0$.

Binary Consensus Proof

Claim

There is no wait-free protocol for binary consensus.

Proof.

1. We know $\mathcal{I}$ is connected, and hence so is any subdivision $\sigma(\mathcal{I})$.
2. Now, consider the decision map $\mu : \sigma(\mathcal{I}) \to \mathcal{O}$. Let I_1 denote the simplex where all processors receive 1.
3. Then, $\Delta(I_1) = O_1$. Thus, if P_1 denotes the vertex in $\sigma(\mathcal{I})$ corresponding to processor P receiving 1, it must be the case that $\mu(P_1) = P_1$.
4. By identical reasoning, for another processor Q, it must be the case that $\mu(Q_0) = Q_0$.
5. However, Q_0 and P_1 are connected in $\mathcal{I}$, but not in $\mathcal{O}$.

Binary Consensus Proof

Claim

There is no wait-free protocol for binary consensus.

Proof.

1. We know $\mathcal{I}$ is connected, and hence so is any subdivision $\sigma(\mathcal{I})$.
2. Now, consider the decision map $\mu : \sigma(\mathcal{I}) \to \mathcal{O}$. Let I_1 denote the simplex where all processors receive 1.
3. Then, $\Delta(I_1) = O_1$. Thus, if P_1 denotes the vertex in $\sigma(\mathcal{I})$ corresponding to processor P receiving 1, it must be the case that $\mu(P_1) = P_1$.
4. By identical reasoning, for another processor Q, it must be the case that $\mu(Q_0) = Q_0$.
5. However, Q_0 and P_1 are connected in $\mathcal{I}$, but not in $\mathcal{O}$.
6. Since simplicial maps must preserve connectivity, this means μ cannot be simplicial! Thus by the theorem, no wait-free protocol!

k-set Agreement Setup

- This problem is a generalization of the consensus problem.
- Every processor has an assigned input value. At the end the processors must output a value, such that the following is satisfied:
  1. Every processor's output is the input of some processor
  2. There are at most k distinct outputs among all the processors.

k-set Agreement Proof

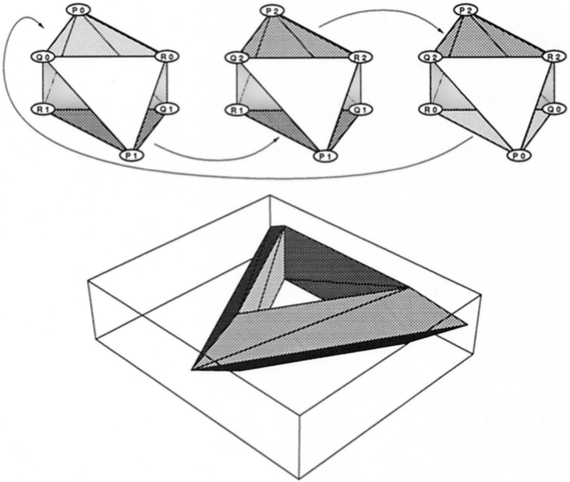

FIG. 22. Output complex for (3,2)-set agreement.

k-set Agreement Proof

Lemma (Sperner's Lemma)

Let $\sigma(T)$ be a subdivision of an n-simplex T. If $F : \sigma(T) \to T$ is a map sending each vertex of $\sigma(T)$ to a vertex in its carrier, then there is at least one n-simplex $S = (\vec{s_0}, \ldots, \vec{s_n})$ in $\sigma(T)$ such that $F(\vec{s_i})$ are distinct.

- Proof Idea: If the subdivision induced by every protocol solving k-set agreement fits this lemma, then some n-simplex S in $\sigma(T)$ is mapped to an output simplex with n distinct outputs.

k-set Agreement Proof

Lemma (Sperner's Lemma)

Let $\sigma(T)$ be a subdivision of an n-simplex T. If $F : \sigma(T) \to T$ is a map sending each vertex of $\sigma(T)$ to a vertex in its carrier, then there is at least one n-simplex $S = (\vec{s_0}, \ldots, \vec{s_n})$ in $\sigma(T)$ such that $F(\vec{s_i})$ are distinct.

- Proof Idea: If the subdivision induced by every protocol solving k-set agreement fits this lemma, then some n-simplex S in $\sigma(T)$ is mapped to an output simplex with n distinct outputs.
- **But**, by definition, the output complex $\mathcal{O}$ contains no such simplex! Thus, no such protocol can exist.
    - This desired simplex corresponds to the "hole" in the output complex.

k-set Agreement Proof

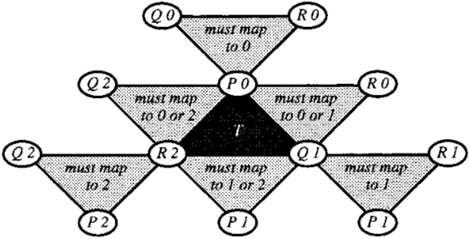

FIG. 23. Part of the set agreement input complex.

- Author's claim: If you observe simplex T, then any vertex in $\sigma(P0, R2)$ must be mapped to a value in $\{0, 2\}$, and similarly $\sigma(P0, Q1)$ in $\{0, 1\}$, and $\sigma(R2, Q1)$ in $\{1, 2\}$.
- Since the **values** being mapped to are a subset of the carrier's (edge's) set of values, the map satisfies the pre-conditions of Sperner's lemma.

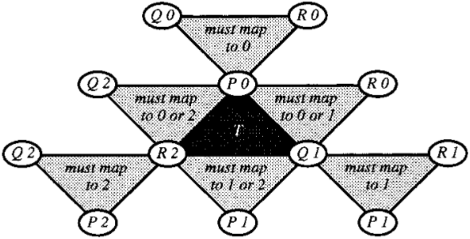

FIG. 23. Part of the set agreement input complex.

- Author's claim: If you observe simplex T, then any vertex in $\sigma(P0, R2)$ must be mapped to a value in $\{0, 2\}$, and similarly $\sigma(P0, Q1)$ in $\{0, 1\}$, and $\sigma(R2, Q1)$ in $\{1, 2\}$.
- Since the **values** being mapped to are a subset of the carrier's (edge's) set of values, the map satisfies the pre-conditions of Sperner's lemma. ???

k-set Agreement Proof

Claim

k-set Agreement has no wait-free read/write protocol for $k \leq n$ where there are $(n+1)$ processors.

Proof.

- It suffices to show there's no protocol for $k = n$. Assume for contradiction there is one, call it $\mathcal{P}$.

$\square$

k-set Agreement Proof

Claim

k-set Agreement has no wait-free read/write protocol for $k \leq n$ where there are $(n+1)$ processors.

Proof.

- It suffices to show there's no protocol for $k = n$. Assume for contradiction there is one, call it $\mathcal{P}$.
- Pick the n-simplex $T^n \subset \mathcal{I}^n$ such that all $n+1$ processors have distinct inputs.

□

k-set Agreement Proof

Claim

k-set Agreement has no wait-free read/write protocol for $k \leq n$ where there are $(n+1)$ processors.

Proof.

- It suffices to show there's no protocol for $k = n$. Assume for contradiction there is one, call it $\mathcal{P}$.
- Pick the n-simplex $T^n \subset \mathcal{I}^n$ such that all $n+1$ processors have distinct inputs.
- Consider any proper face $T^m \subset T^n$.

□

k-set Agreement Proof

Claim

k-set Agreement has no wait-free read/write protocol for $k \leq n$ where there are $(n+1)$ processors.

Proof.

- It suffices to show there's no protocol for $k = n$. Assume for contradiction there is one, call it $\mathcal{P}$.
- Pick the n-simplex $T^n \subset \mathcal{I}^n$ such that all $n+1$ processors have distinct inputs.
- Consider any proper face $T^m \subset T^n$.
- There must be an n-simplex $S^n \subset \mathcal{I}^n$ such that $T^m \subset S^n, \mathrm{vals}(T^m) = \mathrm{vals}(S^n)$.

□

k-set Agreement Proof

Proof.

- By the Asynchronous Computability Theorem, there exists a color-preserving simplicial map $\mu : \sigma(\mathcal{I}^n) \to \mathcal{O}$. By definition $\mu(\sigma(T^m))$ must be consistent with $\mu(S^n)$.

$\square$

- I think more precision is needed to make the highlighted statement
- In particular, perhaps the proof works if you identify all vertices in the input and output complexes with identical value etc.

k-set Agreement Proof

Proof.

- By the Asynchronous Computability Theorem, there exists a color-preserving simplicial map $\mu : \sigma(\mathcal{I}^n) \to \mathcal{O}$. By definition $\mu(\sigma(T^m))$ must be consistent with $\mu(S^n)$.
- By task specification $\mathrm{vals}(\Delta(S^n)) \subset \mathrm{vals}(S^n)$, and thus μ carries every vertex of $\sigma(T^m)$ to one in its carrier.

□

- I think more precision is needed to make the highlighted statement
- In particular, perhaps the proof works if you identify all vertices in the input and output complexes with identical value etc.

k-set Agreement Proof

Proof.

- By the Asynchronous Computability Theorem, there exists a color-preserving simplicial map $\mu : \sigma(\mathcal{I}^n) \to \mathcal{O}$. By definition $\mu(\sigma(T^m))$ must be consistent with $\mu(S^n)$.
- By task specification $\mathrm{vals}(\Delta(S^n)) \subset \mathrm{vals}(S^n)$, and thus μ carries every vertex of $\sigma(T^m)$ to one in its carrier.
- Conclude by applying Sperner's lemma to T^n.

$\square$

- I think more precision is needed to make the highlighted statement
- In particular, perhaps the proof works if you identify all vertices in the input and output complexes with identical value etc.

Table of Contents

Definition

A *span* for a protocol complex $\mathcal{P}(\mathcal{I})$ is a subdivision $\sigma(\mathcal{I})$ and a color-preserving simplicial map $\phi : \sigma(\mathcal{I}) \to \mathcal{P}(\mathcal{I})$ such that for every simplex $S \in \sigma(\mathcal{I})$

$$\phi(S) \in \mathcal{P}(\text{carrier}(S, \sigma(\mathcal{I})))$$

Lemma

Every protocol complex has a span.

- The required subdivision in the original theorem σ is the chromatic subdivision of $\mathcal{I}$ induced by the span
- The color-preserving simplicial map μ is $\delta \circ \phi$ where δ is the decision map acting on $\mathcal{P}(\mathcal{I})$.
- Use topological property of connectivity to inductively construct span
  - Construct for k-skeleton (i.e., all simplexes of dimension at most k), and then use connectivity to show that ϕ can be lifted to $k+1$-skeleton without collapsing dimension (color-preservation).

References I

Questions?

Thank You!