A Classification of Degenerate Loop Agreement

Xingwu Liu[1], Juhua Pu[2], and Jianzhong Pan[3]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China,
xingwuliu@gmail.com
[2] School of Computer Science and Engineering, BeiHang University, Beijing, China,
pujh@buaa.edu.cn
[3] School of Mathematics and System Sciences, Chinese Academy of Sciences, Beijing, China,
pjz@amss.ac.cn

Abstract. Loop agreement is a type of distributed decision tasks including many well-known tasks such as set agreement, simplex agreement, and approximation agreement. Because of its elegant combinatorial structure and its important role in the decidability problem of distributed decision tasks, loop agreement has been thoroughly investigated. A classification of loop agreement tasks has been proposed, based on their relative computational power: tasks are in the same class if and only if they can implement each other. However, the classification does not cover such important tasks as consensus, because any loop agreement task allows up to three distinct output values in an execution. So, this paper considers classifying a variation of loop agreement, called degenerate loop agreement, which includes consensus. A degenerate loop agreement task is defined in terms of its decision space and two distinguished vertices in the space. It is shown that there are exactly two equivalence classes of degenerate loop agreement tasks: one represented by the trivial task, and the other by consensus. The classification is totally determined by connectivity of the decision space of a task; if the distinguished points are connected in the space, the task is equivalent to the trivial task, otherwise to consensus.

Key words: distributed computing, loop agreement, computability, classification

1 Introduction

A distributed computing system consists of finitely many sequential processes communicating via accessing shared read/write registers and other mechanisms [10]. The mechanisms include communication channels, synchronizing primitives, and general services [1, 6]. The processes are asynchronous and may fail by stopping, so it is indistinguishable whether an irresponsive process has failed or is only running slowly. A protocol is a distributed program in such a system. A task is a distributed coordination problem where each process starts with a private input value and decides an output value such that the decisions of all processes meet some specification [7]. Well-known examples of tasks include consensus[5], set consensus[4], and renaming [2]. A protocol is said to solve a task if starting with any legal input assignment, the outputs produced in any execution of the protocol meet the task specification.

Loop agreement [8] is an interesting type of tasks in the theory of distributed computing. A loop agreement task is defined in terms of an edge loop in a 2-complex, with

Please use the following format when citing this chapter:

Liu, X., Pu, J. and Pan, J., 2008, in IFIP International Federation for Information Processing, Volume 273; Fifth IFIP International Conference on Theoretical Computer Science; Giorgio Ausiello, Juhani Karhumäki, Giancarlo Mauri, Luke Ong; (Boston: Springer), pp. 203–213.

three distinguished points on the loop. It stands for a task with the distinguished points as input values and the vertices of the 2-complex as output values. In an execution, if the inputs are the same, the outputs all coincide with the input; if the inputs have two distinct values, the outputs span a simplex along the segment of the loop connecting the two points; otherwise, the outputs span an arbitrary simplex in the complex. Loop agreement is attractive for the following reasons. 1. It has elegant combinatorial structure. 2. It plays a critical role in proving the undecidability of a variety of distributed tasks [7]. 3. It is so general as to include many well-known tasks such as set agreement and approximation agreement.

There are two very influential pieces of work on the computability issue of loop agreement [7, 8]. Ref. [7] showed that a loop agreement task is solvable in certain models if and only if the loop is contractible in the 2-complex, so the solvability of loop agreement tasks in these models is undecidable.

In [8], a classification of loop agreement tasks was presented based on their relative computational power. It considered whether a task T_1 can implement T_2, i.e. T_2 can be solved by calling an instance of a solution to T_1, followed by a protocol using shared read/write registers. Loop agreement tasks can be classified according to the equivalence relation induced by implementation. [8] assigned an algebraic signature to each loop agreement task, which is a pair of the fundamental group of the 2-complex and the path class represented by the loop. It was shown that T_1 can implement T_2 if and only if there is a homomorphism from the signature of T_1 to that of T_2. As a result, the signature completely characterizes the computability of a loop agreement task.

The above work is so elegant. However, its significance is a little weakened in that loop agreement does not include consensus. Consensus is a task whose set of input values is $\{0,1\}$, and in any execution, all the processes agree on the input to some process. Consensus is among the most important tasks in distributed computing, due to its universality [6]. As a result, this paper choose to study an variation of loop agreement, called degenerate loop agreement, which includes consensus. The aim is to adapt the classification of loop agreement tasks in [8] to degenerate loop agreement tasks.

The main contribution of this paper is a complete classification of degenerate loop agreement tasks. Based on the equivalence relation induced by mutual implementation, degenerate loop agreement tasks are divided into two classes: one represented by consensus, the other by the trivial task. The classification is topologically determined; any disconnected task is equivalent to consensus, while connected ones are equivalent to the trivial task.

The rest of this paper is organized as follows. In Section 2, preliminaries on complexes and distributed tasks are presented. In Section 3, degenerate loop agreement tasks are defined. Section 4 proves that there are exactly two classes of degenerate loop agreement tasks, up to the equivalence induced by implementation. Section 5 concludes this paper.

2 Preliminaries

This section will introduce our distributed computing model and formalize the notion of a task. Necessary material from combinatorial topology is also presented, since degenerate loop agreement will be specified using simplicial complexes. Simplicial complexes and their topological properties have long been utilized in distributed computability theory [3, 9, 12]. This paper will exploit connectivity of 1-complexes.

2.1 System model and task formalization

The computing model and task formalization coincide with those in [8], so we will present very briefly. Interested readers please refer to Subsection 3.1 of [8].

We adopt the shared-memory model [10] for distributed computing, where a system consists of a finite set of asynchronous sequential processes, which communicate through accessing shared memory. The shared memory includes read/write registers and possibly more powerful objects and services. A process may delay indefinitely, or fail by stopping.

A task is a distributed coordination problem in which each process starts with a private input value, communicates with others via shared memory, produces an output value, and halts.

Formally, an n-process task T is specified by a triple $(\mathscr{I}, \mathscr{O}, \Delta)$, where $\mathscr{I} \subseteq (D_I \bigcup \{\perp\})^n \setminus \{(\perp, \cdots, \perp)\}$ is the set of input vectors, $\mathscr{O} \subseteq (D_O \bigcup \{\perp\})^n \setminus \{(\perp, \cdots \perp)\}$ is the set of output vectors, and $\Delta \subseteq \mathscr{I} \times \mathscr{O}$ is the task specification. D_I and D_O are respectively the input and output data types. $\mathscr{I}$ and $\mathscr{O}$ are both prefix-closed [8]. An element $I \in \mathscr{I}$ represents an assignment of input values in an execution: if $I_i \neq \perp$, the i^{th} process starts with input I_i, otherwise it does not participate in that execution. The meaning of output vectors can be likewise understood. Δ carries an input vector to a set of matching output vectors, specifying the *legal* outputs for that input assignment. Here, vectors $I \in \mathscr{I}$ and $O \in \mathscr{O}$ are said to match, when for any i, $I_i = \perp$ if and only if $O_i = \perp$.

An n-process protocol is said to t-resiliently solve a task $(\mathscr{I}, \mathscr{O}, \Delta)$, if for every execution where the input vector is I and at least $n-t$ processes decide, the decision vector is a prefix of some output vector in $\Delta(I)$. When $t = n-1$, the protocol is said to be wait-free.

We also borrow the notion of implementation from [8]. A task T_1 is said to be implementable from task T_2, if T_1 can be solved by calling an instance of a protocol that solves T_2, possibly followed by access to shared read/write registers. Implementation naturally induces an equivalence relation where two tasks are equivalent if and only if they are mutually implementable. This relation partitions tasks into equivalence classes, which is the very idea of the classification in this paper.

2.2 Simplicial Complexes

We recall the notion of simplicial complexes and simplicial maps. Readers can also refer to the standard textbook [13, 11] for more information.

Arbitrarily choose a finite set of points $\{v_0, v_1, \cdots, v_m\}$ in the $n-$dimensional Euclidean space R^n. If they are affinely independent, the convex closure $s = \left\{ \sum_{i=0}^{m} \lambda_i v_i \in R^n | \sum_{i=0}^{m} \lambda_i = 1 \text{ and } \lambda_i \geq 0 \text{ for } 0 \leq i \leq m \right\}$ is called the simplex spanned by $\{v_0, v_1, \cdots, v_m\}$, denoted by $\overline{\{v_0, v_1, \cdots, v_m\}}$, and m is called the dimension of s. The simplex spanned by any subset of $\{v_0, v_1, \cdots, v_m\}$ is called a face of s. Each v_i is called a vertex of s.

Two simplices are said to well-positioned, if the intersection of them is either empty or a face of each of them. A finite set of pairwise well-positioned simplices, together with all their faces, is called a (simplicial) complex. A complex is said to be an $n-$complex, if all the simplexes are of dimension no more than n. A complexes C' is said to be a subcomplex of C, if $C' \subseteq C$. Vertices A, B in a complex C are said to connected, if there is a sequence of vertices $v_0 = A, v_1, \cdots, v_n = B$, such that for each $0 \leq i \leq n-1$, $\{v_i, v_{i+1}\}$ spans a simplex in C; such a sequence of vertices is called a path connecting A and B.

A map f from complex C to C' is simplicial, if for each vertex v of C, $f(v)$ is also a vertex of C', and for each simplex $s = \overline{\{v_0, v_1, \cdots, v_m\}} \in C$, $f(s)$ is spanned by the set $\{f(v_i) | 0 \leq i \leq m\}$. Obviously, to define a simplicial map, one only has to define its behavior on vertices.

3 Degenerate Loop Agreement

Definition 1. A 1-complex K, together with two distinct vertices $A, B \in K$, determines a task $(\mathscr{I}, \mathscr{O}, \Delta)$ where $\mathscr{I} = (D_I \bigcup \{\perp\})^n \setminus \{(\perp, \cdots, \perp)\}$, $\mathscr{O} = (D_O \bigcup \{\perp\})^n \setminus \{(\perp, \cdots, \perp)\}$, $D_I = \{0, 1\}$, D_O is the set of vertices of K, and

$$\Delta(I) = \begin{cases} \{O | O \text{ matches } I, \text{ and } val(O) = \{A\}\} & if\ val(I) = \{0\} \\ \{O | O \text{ matches } I, \text{ and } \overline{val(O)} = \{B\}\} & if\ val(I) = \{1\} \\ \{O | O\ matches\ I, \ and\ \overline{val(O)} \in K\} & otherwise \end{cases} \quad (1)$$

The task is called a degenerate loop agreement task and denoted by $T = (K, A, B)$. K is called the decision space of T.

Intuitively, the input values of $T = (K, A, B)$ are 0 and 1, and the output ones are the vertices of K. When all the inputs are 0 (or 1, respectively), all processes decide A (or B, respectively); otherwise, the decided values spans a simplex in K.

Hereunder, a degenerate loop agreement task $T = (K, A, B)$ will be illustrated by the complex K marked with A and B. See Figure 1 as an example.

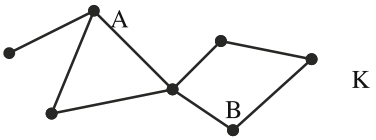

Fig. 1 The illustration of a task $T = (K, A, B)$

Example 1. A famous example of degenerate loop agreement task is consensus, which intuitively means that all processes must agree on a value from their inputs. Formally, $consensus = (\{A, B\}, A, B)$, as illustrated in Figure 2.

Fig. 2 consensus

There is a canonical fact on consensus.

Lemma 1. (Theorem 12.6, [10]) *Consensus can't be solved using read/write registers.*

Example 2. Another example of degenerate loop agreement task is $T = (K, A, B)$, where K consists of the simplex $\overline{\{A, B\}}$ and its faces. See Figure 3 as an illustration of T. Since T can be solved by the protocol where each process trivially outputs its input, it is called the trivial task in this paper.

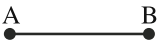

Fig. 3 The trivial task

There is an obvious fact on the trivial task. The proof is omitted here.

Lemma 2. *A degenerate loop agreement task can be solved using read/write registers if and only if it can be implemented by the trivial task.*

4 A Classification of Degenerate Loop Agreement

The main result is that degenerate loop agreement is divided into two classes, as stated in Theorem 1 at the end of this section. To prove this theorem, this section is organized as follows. First, Corollary 1 *normalizes* degenerate loop agreement tasks by removing *redundant* components from their decision spaces. Second, Lemma 6 shows that all disconnected degenerate loop agreement tasks are equivalent. Third, Lemma 8 shows

that all connected degenerate loop agreement tasks are equivalent. Some other technical lemmas are also included.

First of all, we identify a condition which allows one degenerate loop agreement task to implement another. It can be an corollary of Lemma 6.2 in [8], but we provide a much simpler proof.

Lemma 3. *Given two degenerate loop agreement tasks T=(K,A,B) and* $T' = (K', A', B')$*, if there is a simplicial map* $f : K \to K'$ *such that* $f(A) = A'$ *and* $f(B) = B'$*, then T implements* T'*.*

Proof: Choose an arbitrary protocol P for T, and construct a protocol P_f as follows. Each process of P_f first runs protocol P, resulting in a temporary decision value v. Then it outputs $f(v)$ as its final decision. We show that P_f solves T'.

Consider an arbitrary execution of P_f, with S_I/S_O as its set of input/output values, respectively. Assume S'_O to be the set of output values of P in this execution. The following is a case analysis.

Case 1: $S_I = \{0\}$. Then $S'_O = \{A\}$ since P solves T. Because $f(A) = A'$, we have $S_O = \{A'\}$. Likewise, if $S_I = \{1\}$, then $S_O = \{B'\}$.

Case 2: $S_I = \{0,1\}$. Then S'_O spans a simplex in K. Because $f : K \to K'$ is a simplicial map, $S_O = \{f(v) | v \in S'_O\}$ spans a simplex in K'.

As a result, P_f solves T', and hence T implements T'. □

Then we show that a task gets stronger if some part of its decision space is removed, as shown in the following lemma.

Lemma 4. *Given two 1-complexes* K *and* K'*, if* K *is a subcomplex of* K' *and* A, B *are vertices of K, then the degenerate loop agreement task* $T = (K, A, B)$ *implements* $T' = (K', A, B)$*.*

Proof: The inclusion $i : K \to K'$, $v \mapsto v$ is a simplicial map. By Lemma 3, $T = (K, A, B)$ implements $T' = (K', A, B)$. □

Definition 2. Given a degenerate loop agreement task $T = (K, A, B)$, a connected component C of K is called an idle component of T, if C contains neither A nor B.

Lemma 5. *Let C be an idle component of a degenerate loop agreement task T=(K,A,B). Then T is equivalent to* $T' = (K \setminus C, A, B)$*.*

Proof: On the one hand, T' implement T, by Lemma 4.

On the other hand, define a simplicial map $f : K \to K'$,

$$f(v) = \begin{cases} A \ if \ v \ is \ a \ vertex \ in \ C \\ v \ \ otherwise \end{cases} \tag{2}$$

See Figure 4 for an illustration of f. By Lemma 3, T' implement T.

To sum up, T is equivalent to T'. □

According to Lemma 5, a task can be equivalently transformed by eliminating all its idle components, so we immediately have the following corollary.

Corollary 1. *Any degenerate loop agreement task is equivalent to one without idle components.*

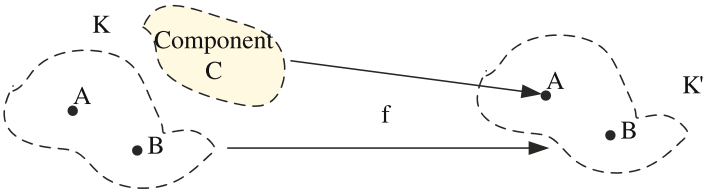

Fig. 4 The map f in Lemma 5

As a result, all the tasks hereunder are assumed to have no idle components, without loss of generality.

Definition 3. A degenerate loop agreement task $T = (K, A, B)$ is said to be connected if A and B are connected in K. Otherwise it is said to be disconnected.

Connectivity is a topological property. The following lemmas show that it plays a critical role in classifying degenerate loop agreement tasks.

Lemma 6. *Any two disconnected degenerate loop agreement tasks are equivalent.*

proof: The basic idea is to show that any disconnected degenerate loop agreement task $T = (K, A, B)$ is equivalent to consensus. Without loss of generality, assume the decision space of consensus is $K' = \{A, B\}$.

First, K' is a subcomplex of K. By Lemma 4, consensus implements T.

Second, define a simplicial map $f : K \to K'$,

$$f(v) = \begin{cases} A \text{ if } v \text{ is in the component containing } A \\ B \text{ if } v \text{ is in the component containing } B \end{cases} \tag{3}$$

See Figure 5 for an illustration of f. By Lemma 3, T implements consensus.

Altogether, T is equivalent to consensus, and the lemma holds. □

To show that that all connected degenerate loop agreement tasks are also equiva-

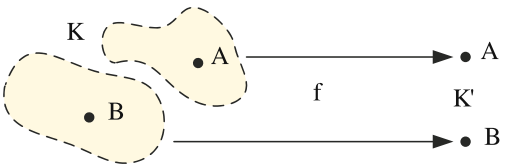

Fig. 5 The map f in Lemma 6

lent, we have to construct a protocol π_m for a special task $\tau_m = (\kappa_m, 0, 1)$, where m is a positive integer. The decision space κ_m of τ_m is a 1-complex in R^1, consisting of the simplices $\overline{\{\frac{i}{2^m}, \frac{i+1}{2^m}\}}$, $0 \le i \le 2^m - 1$, as well as their faces. κ_m is illustrated in Figure 6. The $n-$process protocol π_m is illustrated in Figure 7. It is actually the 1-dimensional

version of the barycentric agreement protocol in [8]. For the completeness of presentation, the correctness of π_m is proved here, in a way that is a little different from that in [8].

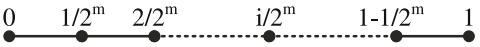

Fig. 6 Decision space κ_m of the task τ_m

```
shared array view[m-1][n];
protocol (input l)
   for r=0..m-1 do
        view[r][P]:=l;
        l:=average of the values in scan(view[r]);
        end for
   decide l;
end protocol
```

Fig. 7 The protocol π_m (for process P)

Lemma 7. *The protocol π_m solves τ_m.*

Proof: First, π_m is wait-free, since each process does not wait for others to progress and it only executes a bounded number of steps before terminating.

Second, We claim that for each r, the values in $view[r]$ always span a simplex in κ_r. The proof is by induction.

Step 1. When $r = 0$, $view[r]$ contains either 0, 1, or 0 and 1, so it spans a simplex in κ_0. The claim holds in this case.

Step 2. Hypothesize that the claim holds for $r_0 < m - 1$.

Step 3. It is obvious that the set of values scanned by one process when $r = r_0$ is either a subset or a superset of that scanned by another process when $r = r_0$. Hence when $r = r_0$, some (possibly zero) processes decide a value in $view[r_0]$, and the others decides the average of the values in $view[r_0]$. As a result, the values in $view[r_0 + 1]$ spans a simplex in κ_{r_0+1}.

To sum up, the values in $view[m-1]$ always span a simplex in κ_{m-1}. Following the argument in step 3, we have that the final values decided by the protocol π_m spans a simplex in κ_m. Furthermore, it is clear that when all the inputs are A, the processes only decides A, likewise for the case of B. So, π_m solves τ_m. □

Now we are ready to adapt Lemma 6 to the case of connected tasks.

Lemma 8. *Any two connected degenerate loop agreement tasks are equivalent.*

Proof: Our idea is to show that any connected degenerate loop agreement task $T = (K, A, B)$ is equivalent to the trivial task. The proof proceeds in two steps. Without loss

of generality, assume that the trivial task is $T' = (K', 0, 1)$, where $K' = \{0, 1, \overline{\{0,1\}}\}$.

Step 1: to prove that T implement the trivial task. Define a simplicial map $f : K \to K'$,

$$f(v) = \begin{cases} 0 \ if \ v = A \\ 1 \ otherwise \end{cases} \tag{4}$$

See Figure 8 for an illustration of f. By Lemma 3, T implements the trivial task.

Step 2: to prove that the trivial task implements T. Because K is connected, there

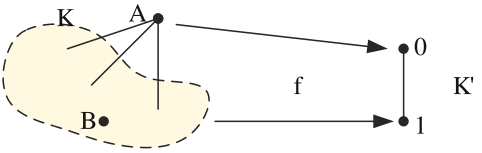

Fig. 8 The map f in Lemma 8

is a path in K connecting A and B. Fix one such path $u_0, u_1, u_2, \cdots, u_n$, where $u_0 = A$ and $u_n = B$. Let $m = \lceil log_2 n \rceil$. By Lemmas 7 and 2, the trivial task implements τ_m.

We now have to show that τ_m implements T. Define a simplicial map $g : \kappa_m \to K$,

$$g(\frac{i}{2^m}) = \begin{cases} u_i \ 0 \le i \le n \\ B \ \ n \le i \le 2^m \end{cases} \tag{5}$$

See Figure 9 for an illustration of f. By Lemma 3, τ_m implements T, so the trivial task implements T.

To sum up, every connected degenerate loop agreement task is equivalent to the trivial task, and the lemma holds. □

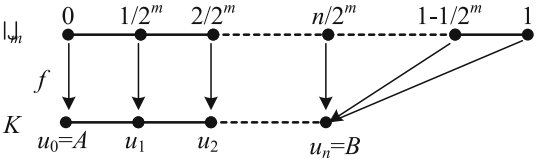

Fig. 9 The map g in Lemma 8

Theorem 1. *There are two equivalence classes of degenerate loop agreement tasks.*

Proof: By Lemma 6 and Lemma 8, degenerate loop agreement tasks can be divided into at most two equivalence classes: one represented by consensus, and the other by the trivial task. By Lemma 1, consensus can not be implemented from the trivial task. As a result, there are exactly two equivalence classes of degenerate loop agreement tasks. □

5 Conclusion

Loop agreement is an interesting type of distributed decision tasks and has been thoroughly studied. However, it does not include the important task of consensus, so this paper considers one of its variation, called degenerate loop agreement, which includes consensus. Classifying degenerate loop agreement tasks is explored to characterize their computational power: two tasks are in the same class if and only if they can implement each other. It turn out that there are exactly two classes: one represented by consensus, including all disconnected tasks, and the other by the trivial task, including all connected tasks. Hence this classification is totally determined by topology of the decision spaces. Compared with the classification of loop agreement where 1-dimensional holes are decisive, our work involves mainly 0-dimension holes, i.e. connectivity. We hope that this provides a further step towards bridging the gap between topology and computer science.

Acknowledgements The work is supported by China's Natural Science Foundation (60603004).

References

1. Attie, P., Guerraoui, R., Kouznetsov, P., Lynch, N., Rajsbaum, S.: The impossibility of boosting distributed service resilience. In: Proceedings of the 25th IEEE International Conference on Distributed Computing Systems, pp. 39–48. IEEE Computer Society, Washington DC, USA (2005)
2. Attiya, H., Bar-Noy, A., Dolev, D., Koller, D., Peleg, D., Reischuk, R.: Achievable cases in an asynchronous environment. In: Proceedings of the 28th Annual Symposium on Foundations of Computer Science, pp. 337–346. IEEE Computer Society (1987)
3. Borowsky, E., Gafni, E.: Generalized flp impossibility result for t-resilient asynchronous computations. In: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing STOC '93, pp. 91–100. ACM Press, New York, NY, USA (1993)
4. Chaudhuri, S.: Agreement is harder than consensus: set consensus problems in totally asynchronous systems. In: Proceedings of the ninth annual ACM symposium on Principles of distributed computing PODC '90, pp. 311–324. ACM Press, New York, NY, USA (1990)
5. Fischer, M., Lynch, N., Paterson, M.: Impossibility of distributed consensus with one faulty process. Journal of the ACM (JACM) **32**(2), 374–382 (1985)
6. Herlihy, M.: Wait-free synchronization. ACM Transactions on Programming Languages and Systems **13**(1), 124–149 (1991)
7. Herlihy, M., Rajsbaum, S.: The decidability of distributed decision tasks (extended abstract). In: Proceedings of the twenty-ninth annual ACM symposium on theory of computing, pp. 589–98. ACM Press, New York, NY, USA (1997)
8. Herlihy, M., Rajsbaum, S.: A classification of wait-free loop agreement tasks. Theoretical Computer Science **291**(1), 55–77 (2003)
9. Hoest, G., Shavit, N.: Towards a topological characterization of asynchronous complexity. In: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing PODC '97, pp. 199–208. ACM Press, New York, NY, USA (1997)
10. Lynch, N.: Distributed Algorithms. Morgan Kaufmann Publishers (1996)
11. Munkres, J.: Elements of Algebraic Topology. Perseus Press (1993)

12. Saks, M., Zaharoglou, F.: Wait-free k-set agreement is impossible: The topology of public knowledge. In: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing STOC '93, pp. 91–100. ACM Press, New York, NY, USA (1993)
13. Spanier, E.: Algebraic Topology. Springer (1994)