Available online at www.sciencedirect.com

Information Processing Letters 96 (2005) 167–174

Information Processing Letters

www.elsevier.com/locate/ipl

A bivalency proof of the lower bound for uniform consensus

Xianbing Wang [a,*], Yong Meng Teo [a,b], Jiannong Cao [c]

[a] *Singapore-Massachusetts Institute of Technology Alliance, 117576 Singapore*
[b] *Department of Computer Science, National University of Singapore, 117543 Singapore*
[c] *Department of Computing, Hong Kong Polytechnic University, Hong Kong*

Received 22 June 2004; received in revised form 26 July 2005

Available online 30 August 2005

Communicated by M. Yamashita

Abstract

Bivalency argument is a widely-used technique that employs forward induction to show impossibility results and lower bounds related to consensus. However, for a synchronous distributed system of n processes with up to t potential and f actual crash failures, applying bivalency argument to prove the lower bound for reaching uniform consensus is still an open problem. In this paper, we address this problem by presenting a bivalency proof that the lower bound for reaching uniform consensus is $(f+2)$-rounds where $0 \leqslant f \leqslant t-2$.
© 2005 Elsevier B.V. All rights reserved.

Keywords: Uniform consensus; Bivalency argument; Synchronous distributed systems; Distributed computing

1. Introduction

Consensus is one of the fundamental problems in distributed computing theory and practice. Assuming that a distributed system consists of a set of n processes, $\{p_1, p_2, \ldots, p_n\}$, each process p_i initially proposes a value v_i, and all non-faulty processes need to decide on one common value v in the set of proposed values $V = \{v_i \mid i = 1, \ldots, n\}$. Without losing generality, in this paper, we consider the consensus problem with $V = \{0, 1\}$ when proving the lower bound for uniform consensus. We assume the crash failure model that is considered to prove lower bounds for both consensus and uniform consensus [1–10]. When a process crashes, it permanently stops its activity [8]. A process behaves correctly until it crashes. A process that does not crash is called a *correct* process. Otherwise, it is a *faulty* process. More precisely, the consensus problem is defined by the following three properties:

(1) *Termination*: Every correct process eventually decides on a value.
(2) *Validity*: If a process decides on v, then v has been proposed by some processes.

* Corresponding author.
E-mail address: wangxb@comp.nus.edu.sg (X. Wang).

0020-0190/$ – see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.ipl.2005.08.002

(3) *Agreement*: No two correct processes decide differently.

The agreement property applies only to correct processes. Thus, it is possible that a process decides on a distinct value just before crashing. *Uniform consensus* prevents such a possibility. It replaces the agreement property with the following:

(3′) *Uniform Agreement*: No two processes (correct or not) decide differently.

Synchronous consensus protocols are based on the notion of *round*. In a synchronous distributed system, every execution of the consensus protocol consists of a sequence of *rounds*. While in round r, each process executes sequentially the following steps:

(1) send r-round messages to the other processes,
(2) wait for r-round messages from the other processes, and
(3) execute local computations.

All processes start and finish the same *round* synchronously. Both message delays and relative process speeds are bounded and these bounds are known. When a process crashes in a round, it sends a subset of the messages that it intends to send in that round, and does not execute any subsequent round [8]. The underlying communication system is assumed to be failure-free: there is no creation, no alteration, no loss nor duplication of messages. If a protocol allows processes to reach *consensus* where at most t $(t < n - 1)$ processes can crash, the protocol is said to tolerate t faults, and is called a t-resilient consensus protocol. It has been proven in [1,2,4,8,9] that the lower bound on the number of rounds is $t + 1$ for any synchronous consensus protocol tolerating up to t crash failures. If a protocol can achieve consensus and *stops* before round $t + 1$ when there are actually f $(f \leqslant t)$ faults, we call it an *early stopping* protocol. The well-known lower bound, $\min(t + 1, f + 2)$ rounds, for early stopping consensus protocols in synchronous distributed systems has been proven [4]. If we consider only the time at which processes decide, protocols in which all processes decide before round $t + 1$ with actual f faults are called *early-deciding protocols*. The lower bound, $(f + 1)$-rounds, for early deciding consensus protocols in synchronous distributed system has also been proven [3,8,9].

Recently, *forward induction* proofs for consensus impossibilities and lower bounds have received attention. For a number of years, only backward induction proofs were known for the synchronous model [4]. Forward induction proofs are simpler and unify various models (synchronous, asynchronous, partially synchronous, etc.) [6]. *Bivalency argument* is a technique that uses forward induction to show impossibility results and lower bounds that are related to consensus. The underlying principle is that there exists a state, called the *bivalent state*, from which two different executions lead to different decisions. The bivalency technique was first introduced in [5] and used in [2] to show the lower bound for achieving consensus in synchronous distributed systems with crash failures. The proof is simpler and more intuitive than traditional ones. However, as mentioned in [10], it is not immediately clear how to extend the proof in [2] to the case of uniform consensus. Keidar and Rajsbaum showed that bivalency argument, as used extensively in the literature, cannot show the $(f + 2)$-rounds lower bound for uniform consensus because, in essence, it relies on a validity property which is too weak [7].

In this paper, we apply the bivalency argument to show that for every $0 \leqslant f \leqslant t - 2$, there exists an execution of any uniform consensus algorithm with actual f failures that takes at least $f + 2$ rounds for all the correct processes to decide on a common value. For $f = t - 1$, the lower bound of $(f + 1)$-rounds for reaching uniform consensus has been proven [3] and is not considered in this paper. Our contributions include a *new* proof for the early deciding uniform consensus lower bound, and a new approach of using the FLP bivalency argument [5].

2. Related works

Charron-Bost and Schiper [3] proved the complete lower bound, in the case where f is less than $t - 1$, that early-deciding uniform consensus protocols require at least $f + 2$ rounds whereas early-deciding consensus protocols require only $f + 1$ rounds. For $f = t - 1$ or $f = t$, they showed that both consensus and uniform consensus require $f + 1$ rounds. In particular, they presented a protocol which can achieve uniform consen-

```
Function Counter_Example($v_i$)
Round 1: Send a message to all the processes (including $p_i$).
        let $S_1$ be the set of processes from which Round 1 messages have been received.
        if $|S_1| = n$ then return (1) fi
Round 2: Send a message to all the processes.
        let $S_2$ be the set of processes from which Round 2 messages have been received.
        if $|S_2| < |S_1|$ then init ← 1 else init ← 0 fi
        return Uniform_Consensus(init).
```

Fig. 1. The counter-example algorithm for bivalent validity in [7].

sus by the end of round $f+1$ when $f=t-1$. Keidar and Rajsbaum [7] took a different approach to prove the same lower bound for synchronous early-deciding uniform consensus protocols, and showed that for $1<t<n$, every t-resilient uniform consensus protocol must perform two rounds in failure-free execution before all processes decide. In [10], a novel oracle argument is introduced to prove both lower bounds for synchronous consensus and uniform consensus. The underlying idea is as follows: given a consensus algorithm A that can tolerate f faults and only executes f rounds of message exchange, another algorithm A' can be constructed to tolerate $f-1$ faults within only $f-1$ rounds. A' does so by making "oracle calls" to A. Repeating this process, an algorithm that only needs 0 rounds for 0 faults can be constructed, which is easily proven impossible.

None of the proofs mentioned above consider the bivalency argument. The proof in [3] proceeds by *backward induction* and is therefore difficult to follow. The oracle-based proof is fundamentally different from bivalency-based proofs [10]. Keidar and Rajsbaum presented a proof in [7] using a different *forward induction* technique called layering [9], which introduce *potence connectivity* to denote connectivity among states in executions of a consensus protocol. They pointed out that "Since all bivalency proofs we know of use the validity property only in order to show that an initial bivalent state exists, such proofs also hold for the *SBV* version of their problems" [7]. Here, *SBV* refers to *bivalent validity with respect to system S* where at most one process fails in each round. It is a weaker validity property stating that there is an initial state which is bivalent with respect to S. Then *SBV* consensus is defined as the problem that satisfies agreement, termination and *SBV*, but not the validity property. They presented a counter example, illustrated in Fig. 1, to solve *SBV* uniform consensus. In Fig. 1, the Uniform_Consensus() algorithm solves the uniform consensus problem, and the Counter_Example() algorithm solves the *SBV* uniform consensus problem. In particular, all processes decide on 1 by the end of one round in all failure-free executions. Because for $f=0$, the counter-example achieves *SBV* uniform consensus with only one round, they concluded that the bivalency argument as used extensively in the literature cannot show the $(f+2)$-rounds lower bound proven in their paper. Their reason is that, in essence, the bivalent argument relies on a validity property that is too weak.

However, when all processes propose 0, they decide on 1 in the failure-free execution. This is accepted by the *SBV* bivalent validity but violates the validity property. In this paper, we present a bivalency proof for the lower bound of synchronous uniform consensus. For the case of $0<f\leqslant t-2$, the validity property of consensus is only used to show initial bivalent configuration. But for $f=0$, we use the validity property in the bivalency proof of this paper.

3. The bivalency argument proof

Bivalency argument proofs are based on the observation that a state in which some processes have decided cannot be bivalent. These proofs proceed by contradiction. For any synchronous consensus algorithm with n processes to tolerate up to t crash failures, contradiction can be reached in a synchronous round-based system S with at most one process crashing in each round [2,9]. Since S is just a subset of executions of a consensus algorithm for synchronous distributed systems and represents the worst case scenario, a lower bound for achieving consensus or uniform consensus in S also holds in synchronous distributed systems.

The following notations are introduced for the system S and used in the bivalency argument proofs.

- *configuration*, a configuration of the system S is a collection of states, one for each process, at the end of each round;
- 0-*valent*, a configuration C is 0-*valent* if starting from C the only possible decision value that correct processes can make is 0;
- 1-*valent*, a configuration C is 1-*valent* if starting from C the only possible decision value that correct processes can make is 1;
- *univalent*, a configuration C is univalent if it is either 0-valent or 1-valent;
- *bivalent*, a configuration C is bivalent if it is not univalent;
- k-*round partial run*, r_k, denotes an execution of a consensus algorithm up to the end of round k.

Considering the configuration C_k at the end of round k of partial run r_k, we say that r_k is 0-valent, 1-valent, univalent, or bivalent if C_k is 0-valent, 1-valent, univalent, or bivalent, respectively.

- *homovalent*, two partial runs are *homovalent* if both are either 1-valent or 0-valent. This means both partial runs are univalent and have the same valence;
- *indistinguishable*, two partial runs *are indistinguishable* if the configurations at the end of the last rounds of both partial runs are the same; they have the same non-crashed processes and each non-crashed process maintains the same information in both partial runs. Thus, the two partial runs cannot be distinguished.

We say that *a partial run r_k decides on v* if all correct processes decide on v by the end of round k of r_k.

3.1. Bivalency argument of Aguilera and Toueg [2]

Our bivalency proof extends the proof of Aguilera and Toueg [2]. The following theorem and lemmas have been reproduced from [2] and will be used in our proof later.

Theorem 1. *Consider a synchronous round-based system S with n processes and at most t crash failures where at most one process crashes in each round. If $n > t + 1$ then there is no algorithm that solves consensus in t rounds in S.*

The proof proceeds by contradiction. Suppose there is an algorithm A that solves consensus in t rounds in S. Without loss of generality, every process is supposed to send a message to every other process at each round. The following three lemmas have been proven in [2].

Lemma 1. *Any $(t-1)$-round partial run r_{t-1} is univalent.*

Lemma 2. *There is a bivalent initial configuration.*

Lemma 3. *There is a bivalent $(t-1)$-round partial run r_{t-1}.*

Lemma 3 contradicts Lemma 1 and thus completes the proof of the theorem. The last two lemmas also appear in other papers, e.g., Lemma 2 is proven by the same method in both [5,9].

3.2. Bivalency proof of uniform consensus lower bound

The main result of this section is Theorem 2, which proves the uniform consensus lower bound that for every $0 \leqslant f \leqslant t-2$, there exists an execution of any uniform consensus algorithm that takes at least $f+2$ rounds for all the correct processes to decide. Proof of Theorem 2 is by contradiction, assuming that there is a protocol A that solves uniform consensus in $f+1$ rounds in S. It considers two cases according to the value of f: (1) $0 < f \leqslant t-2$; and (2) $f = 0$. In the first case, we introduce another synchronous distributed system S', which consists of the same n processes as in system S, but only with up to f potential crash failures where at most one process crashes in each round. We will prove that all executions extended from a $(f-1)$-round bivalent partial run r_{f-1} in system S' decide on the same value. This means r_{f-1} is univalent in system S' which contradicts the fact that r_{f-1} is bivalent. In case (2), we can get two initial configurations of A in system S, C' and C'', that differ by the initial value of only one process p, such that the 1-round partial runs extended from C' and C'' without failure, $r_1'^*$ and $r_1''^*$, decide differently. Then, a contradiction can be reached from this point.

We first introduce several lemmas. Lemmas 4 and 5 are about the properties of the system S. Lemmas 6 and 7 are based on the assumption made in Theorem 2.

Lemma 4 is proven as the induction step of Lemma 3.

Lemma 4. *In the system S, every bivalent k-round partial run* $(0 \leqslant k \leqslant t-2)$, *$r_k$, can be extended by one round into a bivalent $(k+1)$-round partial run.*

Proof. The proof is by contradiction. Assume that every one-round extension of r_k is univalent.

Let r_{k+1}^* be a one-round extension of partial run r_k such that no crash occurs in round $k+1$. r_{k+1}^* is univalent by assumption. Without loss of generality, assume it is 1-valent. Since r_k is bivalent, and every one-round extension of r_k is univalent, there is at least one single-round extension r_{k+1}^0 of r_k that is 0-valent.

Note that r_{k+1}^* and r_{k+1}^0 must differ in round $k+1$. Since round $k+1$ of r_{k+1}^* is failure-free, there must be exactly one process p that crashes in round $k+1$ of r_{k+1}^0 because in system S, at most one process crashes per round, and $k \leqslant t-2$. Since p crashes in round $k+1$ of r_{k+1}^0, it may fail to send a message to some processes, say $\{q_1, q_2, \ldots, q_m\}$, where $0 \leqslant m \leqslant n$.

Starting from r_{k+1}^0, we now define $(k+1)$-round partial runs $r_{k+1}^1, \ldots, r_{k+1}^m$ as follows. For every j, $1 \leqslant j \leqslant m$, r_{k+1}^j is identical to r_{k+1}^{j-1} except that p sends a message to q_j before it crashes in round $k+1$. Note that for every j, $0 \leqslant j \leqslant m$, r_{k+1}^j is univalent. There are two possible cases:

1. For all j, $0 \leqslant j \leqslant m$, r_{k+1}^j is 0-valent. So r_{k+1}^m and r_{k+1}^* are 0-valent and 1-valent respectively. The only difference between r_{k+1}^m and r_{k+1}^* is that p crashes at the end of round $k+1$ in r_{k+1}^m while p is correct up to and including round $k+1$ in r_{k+1}^*. Consider the following runs: r extending r_{k+1}^* by crashing process p at the beginning of round $k+2$, and r' extending r_{k+1}^m without failure. Then r and r' are indistinguishable. However, r is 1-valent and r' is 0-valent because r_{k+1}^* is 1-valent and r_{k+1}^m is 0-valent—contradiction.
2. There is one j, $1 \leqslant j \leqslant m$, such that r_{k+1}^{j-1} is 0-valent while r_{k+1}^j is 1-valent. Extend r_{k+1}^{j-1} and r_{k+1}^j into partial runs, r and r', respectively, by crashing process q_j at the beginning of round $k+2$. Because in system S, at most one process crashes per round, and $k \leqslant t-2$, there is one process which may crash in round $k+2$. Then r and r' are indistinguishable. However, r is 0-valent and r' is 1-valent because r_{k+1}^{j-1} is 0-valent and r_{k+1}^j is 1-valent—contradiction. □

Lemma 5. *By the uniform agreement property, no process (correct or not) can decide in any bivalent partial run in S.*

Proof. It is obvious that the lemma should be true. Otherwise, assume process p decides in a bivalent partial run r in S. Without loss of generality, assume p decides on 1. According to the definition of the bivalent partial run, there is a 0-valent partial run extended from partial run r. However, p has decided on 1, which violates the uniform agreement property. □

Lemmas 6 and 7 are based on the assumption that there is a protocol A that solves uniform consensus in $f+1$ rounds in S. That is, in any execution of A with f $(0 \leqslant f \leqslant t-2)$ actual crash failures, all the correct processes must decide by the end of round $f+1$.

Lemma 6. *Any partial run r_k $(0 < k \leqslant f+1)$ of A without failure during round k in S is a univalent partial run.*

Proof. This lemma is similar to Lemma 6.6 in [9], which consider fast t-resilient consensus protocols. A protocol is called *fast* if consensus is always reached in at most $t+1$ rounds [9].

Assume the contrary, there exists a bivalent partial run r_k $(0 < k \leqslant f+1)$ of A without failure during round k in S. Then we can prove that A cannot solve uniform consensus with f actual failures by the end of round $f+1$.

Case 1: Consider $k = f+1$. Because r_k is bivalent, by Lemma 5, no process can decide by the end of round $f+1$—contradiction.

Case 2: Consider $k < f+1$. According to the definition of S, there is at most one process crashed in each round from round 1 to round $f+1$ except round k. Because r_k is bivalent and $f \leqslant t-2$, by Lemma 4, we can construct new bivalent partial runs as extensions from r_k by crashing one process in each round from round $k+1$ to round $f+1$. Finally, we get a bivalent $(f+1)$-round partial run. In this case, by Lemma 5, no process can decide by the end of round $f+1$—contradiction. □

Lemma 7. *When extending from a bivalent f-round partial run, r_f, all $(f+1)$-round partial runs of A in S, in which at least one process receives all prescribed messages in round $f+1$, are homovalent.*

Proof. First, consider the $(f+1)$-round partial run extended from r_f without failures, r^*_{f+1}. By Lemma 6, r^*_{f+1} is univalent. Because A solves uniform consensus in $f+1$ rounds in S, all processes will decide in round $f+1$. Now consider another $(f+1)$-round partial run extended from r_f with one crash, r_{f+1}, in which process p_i receives all prescribed messages in round $f+1$. p_i cannot distinguish whether it is in r_{f+1} or r^*_{f+1}. It also decides in r_{f+1}. If r_{f+1} is bivalent, by Lemma 5, p_i cannot decide in round $f+1$—contradiction.

Thus, r_{f+1} is univalent. Without losing generality, assume r^*_{f+1} is 1-valent and all processes in r^*_{f+1} decide on 1. Then p_i also decides on 1 in round $f+1$ of r_{f+1}. Thus, r_{f+1} must be 1-valent, otherwise it violates the uniform agreement property. □

Theorem 2. *Consider a synchronous round-based system S of n processes with up to t potential and f actual crash failures, where at most one process crashes in each round. If $t < n$ and $0 \leqslant f \leqslant t-2$, then there is no early deciding protocol that solves uniform consensus in $f+1$ rounds in S.*

Proof. Proof of Theorem 2 is by contradiction, assuming that there is a protocol A that solves uniform consensus in $f+1$ rounds in S. That is, in any execution of A with f $(0 \leqslant f \leqslant t-2)$ actual crash failures, all the correct processes must decide by the end of round $f+1$. The proof considers two cases according to the value of f: (1) $0 < f \leqslant t-2$; and (2) $f = 0$.

Case 1: $\mathbf{0 < f \leqslant t-2}$

We introduce another synchronous distributed system S', which consists of the same n processes as in the system S, but only with up to f potential crash failures that at most one process crashes in each round. Now we consider A for both systems. By Lemmas 2 and 4, there is an $(f-1)$-round bivalent partial run r_{f-1} of A in system S'. Clearly, r_{f-1} is also an $(f-1)$-round bivalent partial run in system S. Because in both systems at most one process can crash in a round, then by Lemma 6, $f-1$ processes crash before round f. We will prove that all executions extended from r_{f-1} in system S' decide on the same value.

In the following, consider r_{f-1} in system S and extend it to round f. Consider r^*_f extended from r_{f-1} without failure occurring in round f, by Lemma 6, it is univalent. Without losing generality, assume r^*_f is 1-valent. Let r^k_fs be those one-round extensions of partial run r_{f-1} in which k processes do not receive the message from the crashed process in round f where $0 \leqslant k \leqslant n-f$ (because there are $f-1$ processes crashed before round f), and r^{k*}_{f+1} denote one-round extension of partial run r^k_f without failures in round $f+1$. By Lemma 6, all those r^{k*}_{f+1}s are univalent.

Basis: Consider r^0_f. Those partial runs are indistinguishable from r^*_f except that one process, p, crashes at the end of round f in r^0_f. There are two cases:

(1) Some processes in both r^0_f and r^*_f decide in round f, by Lemma 7, r^0_f should be 1-valent.

(2) No such process exists. Then extend both r^0_f and r^*_f to round $f+1$, r^{0*}_{f+1} and r^*_{f+1}, simply by crashing p at the beginning of round $f+1$ in r^*_{f+1}. Then two extensions r^{0*}_{f+1} and r^*_{f+1} are indistinguishable, because r^*_f is 1-valent, r^*_{f+1} is 1-valent and decides on 1 eventually.

Thus, all r^{0*}_{f+1}s decide on 1.

Hypothesis: Suppose $0 \leqslant k < n-f$, all r^{k*}_{f+1}s decide on 1.

Induction step: Consider any r^{k+1}_f, where $0 \leqslant k < n-f$, there must exist a partial run, r^k_f, which differs from r^{k+1}_f by only one process, p_i. p_i receives the message sent by the crashed process in round f of r^k_f, but does not receive it in round f of r^{k+1}_f. By Lemma 6, r^{k*}_{f+1} and r^{k+1*}_{f+1} are univalent and r^{k*}_{f+1} is 1-valent by hypothesis.

Consider extending r_f^k and r_f^{k+1} to $r_{f+1}^{k\prime}$ and $r_{f+1}^{k+1\prime}$ respectively by crashing p_i such that only p_j receives the message sent from p_i in both partial runs. Thus, $r_{f+1}^{k+1\prime}$ is indistinguishable from $r_{f+1}^{k\prime}$ except p_j. By Lemma 7, $r_{f+1}^{k\prime}$ and $r_{f+1}^{k+1\prime}$ are univalent because p_j receives all messages in round $f+1$ and $r_{f+1}^{k\prime}$ is 1-valent because r_{f+1}^{k*} is 1-valent.

If some process other than p_j decides in the round $f+1$ of $r_{f+1}^{k\prime}$, and it will does so in $r_{f+1}^{k+1\prime}$, then $r_{f+1}^{k+1\prime}$ is 1-valent too by the uniform agreement property. Otherwise, only p_j decides in both partial runs, extending both $r_{f+1}^{k\prime}$ and $r_{f+1}^{k+1\prime}$ to round $f+2$ (because $f \leqslant t-2$, there exist extended executions in which crashes occur in round $f+1$ and round $f+2$), $r_{f+2}^{k\prime}$ and $r_{f+2}^{k+1\prime}$, by crashing p_j at the beginning of round $f+2$ (for $f=t-1$, we cannot do this because t processes have already crashed before round $f+2$, and no crash occurs in round $f+2$. This is the reason why our proof does not work for $f=t-1$). Then, $r_{f+2}^{k\prime}$ and $r_{f+2}^{k+1\prime}$ are indistinguishable and both are univalent. $r_{f+2}^{k\prime}$ is 1-valent because $r_{f+1}^{k\prime}$ is 1-valent. Thus, $r_{f+2}^{k+1\prime}$ is 1-valent as $r_{f+2}^{k\prime}$, meaning that $r_{f+1}^{k+1\prime}$ is also 1-valent. By Lemma 7, r_{f+1}^{k+1*} must be 1-valent and decide on 1 too.

By induction, all $(f+1)$-round partial runs, extended from r_{f-1}, without failure in round $f+1$ decide on 1. Because r_f^* is univalent, then all $(f+1)$-round partial runs extended from it are 1-valent.

Now consider system S'. Because S' only has up to f crash failures, and there are $f-1$ processes crashed before round f, then when extended from r_{f-1}, only one process can crash in round f or round $f+1$ in S'. Thus, all $(f+1)$-round partial run extensions from r_{f-1} in S' are the same as discussed before in system S. However, all those extensions are homovalent and eventually decide on the same value by the end of round $f+1$. Thus, the r_{f-1} in system S' must be univalent—contradiction.

Case 2: f = 0

Considering initial configurations C^0 and C^1 where all processes propose 0 in C^0 and all processes propose 1 in C^1. According to the validity property of consensus, C^1 is 1-valent and C^0 is 0-valent. Then all one-round partial runs extended from C^0 are 0-valent and all one-round partial runs extended from C^1 are 1-valent. Clearly, there are two initial configurations, C' and C'', that differ by the initial value of only one process p, such that the one-round partial runs extended from C' and C'' without failure, $r_1'^*$ and $r_1''^*$, decide differently. Otherwise, both C^0 and C^1 are the same v-valent, where v is 0 or 1, and this violates the validity property of consensus. Without losing generality, assume that $r_1'^*$ is 1-valent and $r_1''^*$ is 0-valent.

Now consider one-round partial runs, r_1' and r_1'', extended from C' and C'' respectively by crashing p, and only one process q receives the message sent from p in round 1. Then r_1' and r_1'' differ only by q. By Lemma 7, both r_1' and r_1'' are univalent, r_1' is 1-valent, and r_1'' is 0-valent. If some process other than q decides in round 1 of r_1', it decides in r_1'' too. It cannot distinguish r_1' and r_1'' but decides differently—contradiction. Otherwise, extending both r_1' and r_1'' to two-round partial runs, r_2' and r_2'', by crashing q at the beginning of round 2. Then r_2' and r_2'' are indistinguishable. However, r_2' is 1-valent and r_2'' is 0-valent because r_1' is 1-valent and r_1'' is 0-valent—contradiction.

Thus, Theorem 2 must be true. □

4. Conclusions

Bivalency argument is a technique that uses forward induction to show impossibility results and lower bounds that are related to consensus. The proofs are simpler and more intuitive than the traditional ones. In this paper, for a synchronous distributed system of n processes with up to t potential and f actual crash failures, we apply the bivalency argument to show that for every $0 \leqslant f \leqslant t-2$, there exists an execution of any uniform consensus algorithm that takes at least $f+2$ rounds for all the correct processes to decide. Previously, finding a bivalency proof for the lower bound of synchronous uniform consensus had been an open problem.

Acknowledgement

This work is partially supported by the Hong Kong UGC under the CERG grant B-Q567 (PolyU 5075/02E) and the National University of Singapore under Academic Research Fund R-252-000-180-112. We wish to thank anonymous referees for their helpful suggestions and comments.

References

[1] H. Attiya, J. Welch, Distributed Computing: Fundamentals, Simulations and Advanced Topics, McGraw-Hill, New York, 1998, 451 p.

[2] M.K. Aguilera, S. Toueg, A simple bivalency proof that t-resilient consensus requires $t + 1$ rounds, Inform. Process. Lett. 71 (3–4) (1999) 155–158.

[3] B. Charron-Bost, A. Schiper, Uniform consensus harder than consensus, J. Algorithms 51 (1) (2004) 15–37.

[4] D. Dolev, R. Reischuk, R. Strong, Early stopping in Byzantine agreement, J. ACM 37 (4) (1990) 720–741.

[5] M. Fischer, N. Lynch, M. Paterson, Impossibility of distributed consensus with one faulty process, J. ACM 32 (2) (1985) 374–382.

[6] M. Herlihy, S. Rajsbaum, M.R. Tuttle, Unifying synchronous and asynchronous message-passing models, In: Proc. 17th ACM Symp. on Principles of Distributed Computing (PODC), June 1998, pp. 133–142.

[7] I. Keidar, S. Rajsbaum, A simple proof of the uniform consensus synchronous lower bound, Inform. Process. Lett. 85 (1) (2003) 47–52.

[8] N. Lynch, Distributed Algorithms, Morgan Kaufmann, Los Altos, CA, 1996.

[9] Y. Moses, S. Rajsbaum, A layered analysis of consensus, SIAM J. Comput. 31 (4) (2002) 989–1021. Previous version in: PODC 1998.

[10] J. Xu, A unified proof of minimum time complexity for reaching consensus and uniform consensus—an Oracle-based approach, in: IEEE 21st Symp. on Reliable Distributed Systems (SRDS 2002), Osaka, Japan, October 2002.