

A Weakly-Adaptive Condition-Based Consensus Algorithm in Asynchronous Distributed Systems

Taisuke IZUMI and Toshimitsu MASUZAWA

Graduate School of Information Science and Technology, Osaka University

E-mail: {t-izumi, masuzawa}@ist.osaka-u.ac.jp

Keywords: distributed computing, fault tolerance, consensus problem, condition-based approach

1 Introduction

The *consensus* problem is one of the most fundamental and important problems for designing fault-tolerant distributed systems. In the consensus problem, each process proposes a value, and all non-faulty processes have to agree on a common value that is proposed by a process. Despite of many applications (e.g., atomic broadcast [2, 6], shared object [1, 7], weak atomic commitment [5]), it is known that the consensus problem has no deterministic solution in asynchronous systems subject to only a single crash fault [4]. Thus, to circumvent this impossibility, several approaches, such as *partial synchrony* [3] and *unreliable failure detectors* [2], have been proposed. As one of such approaches, the *condition-based approach* is recently introduced [9]. This approach introduces restriction (called *condition*) on inputs so that the generally-unsolvable problem can be solved for the restricted set of inputs. In the case of the consensus problem, a condition is defined as a subset of all possible *input vectors* whose entries correspond to the proposal of each process. The first result of the condition-based approach clarifies the condition for which the uniform consensus can be solved in asynchronous systems subject to crash faults [9]. This result proposed a class of conditions, called *t-legal conditions*, and proved that the *t*-legal conditions form the class of necessary and sufficient conditions to solve the consensus in asynchronous systems where at most *t* process can crash.

In general, for smaller number of *t*, the *t*-legal conditions become weaker restrictions and can contain more input vectors. An example of such *t*-legal conditions is the condition C_t^{\max} , proposed in [9]. The condition C_t^{\max} is the set of input vectors where the maximum value in each vector appears at *t* entries or more in the vector. From the definition, the condition C_t^{\max} is a subset of the condition C_{t-1}^{\max} . In this sense, the sizes of sufficient conditions are strongly related to the number of crash faults that the system suffers. That is, the consensus algorithm designed for tolerating a larger number of faults can solve the consensus problem only for a smaller set of input vectors. However, in the system with *t* crash faults at a maximum, it is not often that *t* processes actually crashes. In other words, the actual number of crash faults is relatively smaller than the maximum. This observation brings one question as follows: Can we construct the condition-based algorithm that solves the consensus problem for more relaxed conditions than *t*-legal one when the actual number of crash faults is smaller than *t*?

In this paper, we investigate this problem by applying the *adaptive condition-based approach*. In the adaptive condition-based approach, a restriction to input vectors is not represented by a single subset of inputs, but represented by a hierarchical sequence of conditions called a *condition sequence*. Adaptive condition-based algorithms are instantiated by a condition sequence, and guarantee some property according to the position of input vector in the hierarchy of the given condition sequence (i.e., the execution for the input vector with higher position achieves better property). The first result for the adaptive condition-based approach considers time complexity in the synchronous

consensus [8]. It considers a synchronous condition-based consensus algorithm that is instantiated by a hierarchical condition sequence such that k -th condition reduces the worst-case time complexity of the synchronous consensus by k rounds.

We use the condition sequence to design the asynchronous condition-based consensus. More precisely, this paper considers consensus algorithms as the following form: They are instantiated by a condition sequence, and guarantee to solve the consensus problem if the input vector belongs to the k -th condition in the given condition sequence and at most k processes crash. The contribution of this paper is to identify the class of condition sequences for which such adaptive condition-based consensus can be constructed. To identify it, we present a property of condition sequences called *legality*. We prove that the adaptive condition-based algorithm for a condition sequence S can be constructed if and only if S is legal. The notion of condition sequences and its legality is a generalization of standard conditions and d -legality proposed in [9] respectively. In this sense, our result can be regarded as a generalization of the original condition-based result [9]. In addition, we also present an instance of legal condition sequences. It is defined for a threshold value k . If the actual number of crash faults is smaller than k , the algorithm for this condition sequence solves the consensus problem for a larger set of input vectors than the non-adaptive algorithm. However, if more than k processes crash, this algorithm solves the consensus only for a smaller set of input vectors than the existing one. In this sense, this algorithm is *weakly-adaptive*. Practically, the actual number of crash faults is quite smaller than the maximum one in usual situations. In other words, it hardly occurs that the actual number of crash faults exceeds the threshold value k . This implies that the proposed solution is more useful than the existing one.

2 Preliminaries

2.1 Distributed Systems

We consider the standard asynchronous distributed systems as defined in [2]. A distributed system consists of n processes $\mathcal{P} = \{p_0, p_1, p_2, \dots, p_{n-1}\}$, in which any pair of processes can communicate with each other by exchanging messages. The system is asynchronous in the sense that there is no bound for communication delay and local processing time. Each process can crash. If a process crashes, it prematurely stops its execution and makes no operation subsequently. Notice that each process can crash at any time. A process that does not crash (even in the future) is called a *correct* process. We assume that there is some upper bound t on the number of processes that can crash in the whole system. We also assume $t < n/2$. Every process knows the value of t a priori. The actual number of crash faults is denoted by f . The value of f is unknown to each process. All communication channels are reliable; neither message creation, alteration, or loss occurs.

2.2 The Condition-Based Consensus

A (uniform) consensus algorithm provides each process p_i with two primitives, $propose_i(v)$ and $decide_i(v)$, as the interface to the upper application layer. In a consensus algorithm, each correct process p_i initially proposes a value v by $propose_i(v)$, and eventually chooses a decision value v' by $decide_i(v')$. Then, the decision value must be chosen from the values proposed by processes so that all processes decide the same value. The condition-based consensus is a weaker variant of the consensus. It introduces a subset of all possible inputs, called a *condition*, and correctly solves the consensus problem if the actual input belongs to the given condition or no process crashes. To define the condition-based consensus, we first give the formal definition of conditions. Let \mathcal{V} be the set of values that can be proposed. We assume that \mathcal{V} is a finite ordered set. The proposed values in an execution are represented as an *input vector*, where the i -th entry represents the value of

p_i 's proposal $v_i \in \mathcal{V}$. A *condition* is a subset of all possible input vectors \mathcal{V}^n . The condition-based consensus for a condition C is specified as follows:

Validity A decision value is one of proposals.

Uniform Agreement No two processes decide different values.

Guaranteed Termination If (1) the input vector I belongs to C and no more than f processes crash, or (2) all processes are correct, or (3) a process decides, then every correct process decides.

As we stated in the introduction, this paper investigates condition-based consensus algorithms whose condition does not depend on the maximum number of faults that can be tolerated, but depends on the actual number of faults. To handle this issue, we use the notion of the *adaptive* condition-based approach. In this approach, a hierarchical sequence of conditions is considered instead of a single (static) condition. It introduces a *condition sequence* $S = (C_1, C_2, \dots, C_t)$, where each C_k is a condition and $C_{k+1} \subseteq C_k$ holds for any $k(1 \leq k \leq t-1)$. Throughout this paper, the k -th condition C_k of a condition sequence represents the condition which is valid when the actual number of faults is k (i.e., letting \mathcal{A} be an adaptive condition-based consensus algorithm for S , C_k is the set of input vectors for which \mathcal{A} solves the consensus problem if at most k processes crash). Formally, the adaptive condition-based consensus for a condition sequence $S = (C_1, C_2, \dots, C_t)$ is specified by Validity, Uniform Agreement, and the following property:

Adaptive Guaranteed Termination If (1) the input vector I belongs to C_k and no more than k processes crash, or (2) all processes are correct, or (3) a process decides, then every correct process decides.

3 Solvability of Adaptive Condition-Based Consensus

3.1 Notations

For an input vector $I \in \mathcal{V}$, we define a *view* J of I to be a vector in $(\mathcal{V} \cup \{\perp\})^n$ obtained by replacing several entries in I by \perp (\perp is a default value such that $\perp \notin \mathcal{V}$). Let \perp^n be the view such that all entries are \perp . For views J_1 and J_2 , the containment relation $J_1 \leq J_2$ is defined as $\forall k(0 \leq k \leq n-1) : J_1[k] \neq \perp \Rightarrow J_1[k] = J_2[k]$. For a view $J \in (\mathcal{V} \cup \{\perp\})^n$ and a value $a \in \mathcal{V} \cup \{\perp\}$, $\#_a(J)$ denotes the number of entries of value a in the view J , that is $\#_a(J) = |\{k \in \{0, 1, \dots, n-1\} | J[k] = a\}|$. For a view J and a value a , we often describe $a \in J$ if there exists a value k such that $J[k] = a$. For an input vector $I \in \mathcal{V}^n$, let $[I]_k$ be the set of all views J such that J is obtained by replacing at most k entries of I by \perp . For a condition $C \subseteq \mathcal{V}^n$, let $[C]_k$ be the union of $[I]_k$ over all $I \in C$.

3.2 Characterization Theorem

This subsection presents the characterization theorem for the consensus solvability. The key idea of the characterization theorem derives from the notion of *d-legality* in [9]. We consider a graph representation of condition sequences, and the characterization is given as the property of such graphs. To provide the theorem, we first introduce the graph Gin that is defined for a condition sequence S . The basic notion of graph Gin is represented in [9]. In this paper, we introduce its adaptive version.

Definition 1 The graph $\text{Gin}(S)$ for a condition sequence $S = (C_1, C_2, \dots, C_t)$ is one such that

- the vertex set consists of all views in $\bigcup_{k=1}^t [C_k]_k$, and
- two views J_1 and J_2 are connected if $J_1 \leq J_2$ or $J_2 \leq J_1$ holds.

The legality of a condition sequence is defined as follows:

Definition 2 (Legality) A condition sequence S is legal if, for each connected component Com of $\text{Gin}(S)$, at least one common value is included in all views in Com .

Using the above definitions, we state the characterization theorem.

Theorem 1 There exists an adaptive condition-based consensus algorithm for a condition sequence S if and only if S is legal.

3.3 An Example of Sufficient Condition Sequences

Before the proof of the characterization theorem, we propose an example of legal condition sequences. We first introduce the condition that are basis of the example.

Definition 3 Let $\text{1st}(J)$ be the non- \perp value that appears most often in view J (if two or more values appear most often (if two or more values appear with same times in J , the maximum one is the value of $\text{1st}(J)$), the largest one is chosen), and \hat{J} be the vector obtained from J by replacing $\text{1st}(J)$ by \perp . The value of $\text{1st}(\hat{J})$ is denoted by $\text{2nd}(J)$. The frequency-based condition C_d^{freq} is defined as follows:

$$C_d^{\text{freq}} = \{I \in \mathcal{V}^n \mid \#_{\text{1st}(I)}(I) - \#_{\text{2nd}(I)}(I) > d\}$$

It is known that C_d^{freq} belongs to the d -legal conditions [9]. Thus, for the condition C_t^{freq} , we can construct a (non-adaptive) condition-based consensus algorithm tolerating t crash faults. Restating in the context of the adaptive condition-based approach, the condition sequence $S = \{C_t^{\text{freq}}, C_t^{\text{freq}}, \dots, C_t^{\text{freq}}\}$ is legal.

Using this condition, we can define a legal condition sequence.

Definition 4 (Frequency-Based Adaptive Condition Sequence) For any $k(1 \leq k \leq t)$, the condition sequence $Sa^{\text{freq}}(k)$ is defined as follows:

$$Sa^{\text{freq}}(k) = (\overbrace{C_k^{\text{freq}}, C_k^{\text{freq}}, \dots, C_k^{\text{freq}}}^k, \overbrace{C_{2t-k}^{\text{freq}}, \dots, C_{2t-k}^{\text{freq}}}^{t-k})$$

The above condition sequence is legal (the proof is given at Section 5). In the condition sequence $Sa^{\text{freq}}(k)$, a larger number of input vectors are guaranteed to terminate than the non-adaptive version of the condition-based consensus for C_t^{freq} if the actual number of faults is smaller than k . In contrast, if many faults (larger than k) occurs, a smaller number of input vectors are guaranteed to terminate. In this sense, there is the tradeoff between non-adaptive algorithms for C_t^{freq} and the adaptive algorithm for Sa^{freq} . However, in executions of real distributed systems, few processes crashes usually. In such typical situations, the adaptive algorithm is more useful.

4 Proof of Characterization Theorem

4.1 Proof of Sufficiency

This subsection presents the sufficiency proof of the theorem. It is shown by presenting a generic weakly-adaptive consensus algorithm for any legal condition sequence S .

We introduce a generic consensus algorithm that is instantiated by any legal condition sequence S . To develop the algorithm, we use the function h , that is the mapping from a view in the node set of $\text{Gin}(S)$ to a value in \mathcal{V} . The mapped value $h(J)$ for a view J is one that appears in common at all views in the connected component to which J belongs (such a value necessarily exists from the fact that S is legal). If two or more values appear in common, the largest one is chosen. We also use one communication abstraction called the *uniform reliable broadcast*. It provides each process p_i with two primitives $\text{UR_bcast}_i(m)$ and $\text{UR_deliver}_i(m)$, whose invocations respectively mean the broadcast of message m from p_i to all processes and the delivery of broadcast message m at p_i . Compared with the sending m to all processes by repeatedly invoking (point-to-point) Send primitives, the uniform reliable broadcast is different in the point of reliability. If a process crashes while broadcasting a message m to all processes by repeatedly invoking Send primitives, the message m might be received by only a subset of all processes. In contrast, using the uniform reliable broadcast to send a message m , all correct processes necessarily deliver m if a process delivers m . Formally, the uniform reliable broadcast is specified by the following properties:

Validity If a process p_i invokes $\text{UR_deliver}_i(m)$, then some process p_k invokes $\text{UR_bcast}_k(m)$.

Integrity A process p_i invokes $\text{UR_deliver}_i(m)$ at most once.

Termination If a correct process p_i invokes $\text{UR_bcast}_i(m)$ or a process p_k invokes $\text{UR_deliver}_k(m)$, any correct process p_j eventually invokes $\text{UR_deliver}_j(m)$.

The uniform reliable broadcast can be implemented in asynchronous systems [6]. Therefore, using this primitive does not implies that some additional assumption is introduced into the distributed system model.

Figure 1 presents the pseudo-code description of the generic weakly-adaptive condition-based consensus algorithm. The principle of this algorithm is almost the same as one in [9]. It consists of two phases. In the first phase, each process p_i exchanges its proposal with each other, and constructs the view J_i of input vector I . The view J_i is maintained incrementally. That is, it is updated on each reception of a message. When the view J_i is updated, process p_i tests (1) whether J_i appears in the vertex set of graph $\text{Gin}(S)$ or not, and (2) whether all initial \perp values disappear from J_i (that is, p_i gathers all proposals) or not. The usage of $\text{Gin}(S)$ induced by condition sequeunce S in the test (1) is the only point where our algorithm is different from the previous one in [9]. If either of two tests is passed, process p_i proceeds to the second phase. Notice that the role of the test (2) is to guarantee termination if no crash occurs. In the second phase, p_i broadcasts *echo* message by the uniform reliable broadcast. An echo message broadcast by p_i contains p_i 's decision estimation. If the test (1) is passed, the estimation is $h(J_i)$. Otherwise, the estimation is the maximum value in J_i . Each process in the second phase decides when either of two following conditions is satisfied. The first condition is that p_i receives more than $n/2$ echo messages carrying a same value v . In this case, p_i decides v . The second one is that p_i receives echo messages from all processes. In this case, p_i decides the maximum of all values carried by the received echo messages.

A generic algorithm for legal condition sequence $S = (C_0, C_1, \dots, C_t)$ (Code for p_i):

```

1: variable:
2:    $J_i, Y_i$  : init  $\perp^n$ 

3: Upon  $\text{Propose}_i(v)$  do:
4:    $\text{Send}_i(v, p_j)$  for every  $p_j \in \mathcal{P}$ 
5: Upon  $\text{Receive}_i(v)$  from  $p_j$  do:
6:    $J_i[j] \leftarrow v$ 
7:   if  $J_i$  is a vertex in  $\text{Gin}(S)$  then  $\text{UR\_bcast}_i(\text{echo}, h(J_i))$ 
8:   elseif  $\#_{\perp}(J_i) = 0$  then  $\text{UR\_bcast}_i(\text{echo}, \max(J_i))$  endif
9: Upon  $\text{UR\_deliver}(\text{echo}, v)$  from  $p_j$  do:
10:   $Y_i[j] \leftarrow v$ 
11:  if  $\#_v(Y_i) > n/2$  then  $\text{Decide}_i(v)$ 
12:  elseif  $\#_{\perp}(Y_i) = 0$  then  $\text{Decide}_i(\max(Y_i))$  endif /*  $\max(X)$  returns the maximum value in  $X$  */

```

Figure 1: An adaptive condition-based consensus algorithm for legal condition sequence S

4.1.1 Correctness

Lemma 1 If the input vector I belongs to C_k and at most k processes crash, (1) all the broadcast echo messages carry a same value, and (2) all correct processes broadcast the echo messages.

Proof First, we prove that each correct process p_i sends a echo message. Since at most k processes crash, each correct process p_i receives messages from at least $n - k$ processes in the first phase. This implies that $J_i \in [I]_k \subseteq [C_k]_k$ (= the vertex set of $\text{Gin}(S)$) eventually holds. Thus, each process p_i sends an echo message. Next, for two echo messages (echo, v_i) and (echo, v_j) broadcast by two processes p_i and p_j respectively, we show $v_i = v_j$. Let J_i and J_j be the views when p_i and p_j broadcast echo messages. Clearly, $J_i \leq I$ and $J_j \leq I$ holds. This implies that J_i and J_j belongs to a same connected component in $\text{Gin}(S)$. Thus, we obtain $v_i = h(J_i) = h(J_j) = v_j$. \square

Lemma 2 (Validity) If a process decides a value v , then, v is a value proposed by a process.

Proof Clearly, if a process p_i decides v , the decision value v appears in Y_i , and thus v is carried by some echo message. Let p_j is the broadcaster of the echo message. Whichever p_j broadcasts the echo message at line 9 or 11, v appears in J_j . Since each entry in J_j stores a proposal value, v is a value proposed by a process. \square

Lemma 3 (Adaptive Guaranteed Termination) If (1) the input vector I belongs to C_k and no more than k processes crash, (2) all processes are correct, or (3) a process decides, then every correct process decides.

Proof (1) If input vector I belongs to C_k and no more than k processes crash, each correct process sends the echo message carrying a same value v (Lemma 2). Then, since at least $n - t (> n/2)$ processes are correct, each process receives more than $n/2$ echo messages carrying the same value. This implies that each process decides by executing line 11. (2) If no processes crashes, J_i eventually becomes a view containing no \perp value for any p_i . This implies that each process p_i broadcasts an echo message by executing line 7 or 8. Since each process p_i is correct, this message is necessarily delivered by any process, and thus all initial \perp value disappear from Y_j for any p_j . This implies that each process p_j decides by executing line 11 or 12. (3) If a process p_k decides, either the conditions of the if statements at line 11 or 12 holds on p_k . Since echo messages are broadcast by

the uniform reliable broadcast primitive, each correct process p_i eventually receives the same set of echo messages as p_k . This implies that the conditions at line 11 or 12 also holds on p_i , that is, p_i also decides. \square

Lemma 4 (Uniform Agreement) No two processes decide differently.

Proof Let p_i and p_j be the processes that decide, and v_i and v_j be the decision values of p_i and p_j respectively. Then, we prove $v_i = v_j$. We consider the following three cases. **(Case1)** Both p_i and p_j decide at line 11: p_i delivers more than $n/2$ echo messages carrying v_i , and the same holds for p_j and v_j . Then, clearly, some process p_k sends an echo message (echo, v_i) and (echo, v_j) to p_i and p_j respectively. This implies $v_i = v_j$. **(Case2)** Both p_i and p_j decide at line 12: let Y'_i and Y'_j be the value of Y_i and Y_j when p_i and p_j decide respectively. Since $\#_{\perp}(Y'_i) = \#_{\perp}(Y'_j) = 0$ holds, both process p_i and p_j receive the same set of echo messages. This implies $Y'_i = Y'_j$. Thus, we obtain $v_i = \max(Y'_i) = \max(Y'_j) = v_j$. **(Case3)** Process p_i decides at line 11 and p_j decides at line 12: We prove that this case never occurs. Suppose for contradiction that the case occurs. Then, p_i delivers more than $n/2$ echo messages carrying the value v_i . Since p_j delivers echo messages from all processes, p_j also delivers more than $n/2$ echo messages carrying the value v_i . It follows that p_j decide at line 11 (notice that the decision at line 11 has higher priority than that at line 12). This is contradiction. \square

From Lemmas 2, 3, and 4, we can show the following lemma that implies the sufficiency of the characterization theorem.

Lemma 5 The adaptive condition-based consensus problem for any legal condition sequence S is solvable.

4.2 Proof of Necessity

This subsection presents the necessity proof of the characterization theorem. The key lemma to prove the theorem is same as one proposed in [9], and here, we simply quote it. For the proof, we first introduce several notations. Let \mathcal{A} be an adaptive condition-based consensus algorithm for condition sequence S . For view J , we define $E(\mathcal{A}, J)$ as the set of all possible executions of the algorithm \mathcal{A} where process p_i proposes $J[i]$ if $J[i] \neq \perp$, or initially crashes. Let $\text{val}(\mathcal{A}, J)$ be all possible decision values that appears in $E(\mathcal{A}, J)$.

The following lemma is the key of our necessity proof, and a paraphrase of Theorem 5.4 in [9].

Lemma 6 (Mostefaoui et.al. 2003) For any two views J_1 and J_2 that are connected in $\text{Gin}(S)$, $\text{val}(\mathcal{A}, J_1) = \text{val}(\mathcal{A}, J_2)$ holds.

Using this lemma, we can show the necessity of our characterization theorem.

Lemma 7 If some algorithm \mathcal{A} solves the adaptive condition-based consensus problem for condition sequence S , S is legal.

Proof Let Com be any connected component of $\text{Gin}(S)$, J_1 be some view in Com , and v be a value in $\text{val}(\mathcal{A}, J_1)$. From Lemma 6, $v \in \text{val}(\mathcal{A}, J_2)$ holds for any view J_2 in Com (that is, any view connecting to J_1). Since $v \in \text{val}(\mathcal{A}, J_1)$ holds, v is a decision value of the algorithm \mathcal{A} when the input vector is J_1 . Thus, v appears in J_1 because of the validity property of the consensus problem. By the same token, v also appears in J_2 . It follows that any view in Com have a common value v . The lemma is proved. \square

5 The Proof of Legality for Condition Sequence $Sa^{\text{freq}}(k)$

In this section, we prove the legality of the condition sequence $Sa^{\text{freq}}(k)$.

Theorem 2 For any $k(1 \leq k \leq t)$, the condition sequence

$$Sa^{\text{freq}}(k) = (\overbrace{C_k^{\text{freq}}, C_k^{\text{freq}}, \dots, C_k^{\text{freq}}}^k, \overbrace{C_{2t-k}^{\text{freq}}, \dots, C_{2t-k}^{\text{freq}}}^{t-k})$$

is legal.

Proof We prove that for any two views J_1 and J_2 in a connected component in $\text{Gin}(Sa^{\text{freq}}(k))$, $\text{1st}(J_1) = \text{1st}(J_2)$ holds. Suppose for contradiction that there exist two views J_1 and J_2 such that $\text{1st}(J_1) \neq \text{1st}(J_2)$ holds. We have a path P between J_1 and J_2 in $\text{Gin}(Sa^{\text{freq}})$. Then, there exist two views J'_1 and J'_2 on the path P satisfying $J'_1 \leq J'_2$ (that is, J'_1 is a neighbor of J'_2) and $\text{1st}(J'_1) \neq \text{1st}(J'_2)$. If both J'_1 and J'_2 belongs to $[C_{2t-k}^{\text{freq}}]_t$, there exists an input vector $I' \in C_{2t-k}^{\text{freq}}$ satisfying $J'_1 \leq J'_2 \leq I'$. For J'_1 and I' , $\#_{\text{1st}(I')}(I') - \#_{\text{1st}(J'_1)}(I') > 2t - k \geq t$ and $\#_{\perp}(J'_1) \leq t$ holds. This implies $\#_{\text{1st}(I')}(J'_1) - \#_{\text{1st}(J'_1)}(J'_1) > 0$, and thus we obtain $\text{1st}(I') = \text{1st}(J'_1)$. This argument also holds for J'_2 and I' . Thus, we obtain $\text{1st}(I') = \text{1st}(J'_2)$. Then, $\text{1st}(J'_1) = \text{1st}(J'_2) = \text{1st}(I')$ holds, however, it contradicts to $\text{1st}(J'_1) \neq \text{1st}(J'_2)$. Thus, both J'_1 and J'_2 does not belong to $[C_{2t-k}^{\text{freq}}]_t$. By the same argument, we can show that both J'_1 and J'_2 do not belong to $[C_k^{\text{freq}}]_k$. Therefore, we can conclude that one of J'_1 and J'_2 belongs to $[C_{2t-k}^{\text{freq}}]_t$ and the other belongs to $[C_k^{\text{freq}}]_k$ and not to $[C_{2t-k}^{\text{freq}}]_t$. From $J'_1 \leq J'_2$, it follows that $J'_1 \in [C_{2t-k}^{\text{freq}}]_t$ and $J'_2 \in [C_k^{\text{freq}}]_k$ holds. Then, w.l.o.g, we can assume $\#_{\perp}(J'_2) = k$ (because, if $\#_{\perp}(J'_2) \neq k$, there is a view J'_3 satisfying $J'_1 \leq J'_3 < J'_2$, $J'_3 \in [C_k^{\text{freq}}]_k$, $\#_{\perp}(J'_3) = k$ and $\text{1st}(J'_2) = \text{1st}(J'_3)$). Thus, we can replace J'_2 by J'_3 . Since $\#_{\text{1st}(J'_2)}(J'_2) - \#_{\text{1st}(J'_1)}(J'_2) > 0$ holds, we obtain $\#_{\perp}(J'_2) - \#_{\perp}(J'_1) \leq t - k$. This implies $\#_{\text{1st}(J'_1)}(J'_1) - \#_{\text{1st}(J'_2)}(J'_1) < t - k$, and contradicts to $J'_1 \in [C_{2t-k}^{\text{freq}}]_t$. \square

6 Discussion

The hierarchical structure of conditions is first considered in the paper [10]. In this paper, the authors proposed that the conditions for which there exists a (non-adaptive) condition-based consensus algorithm tolerating d crash faults form a hierarchy according to the value of d . They also remark the feasibility of the adaptive condition-based approach in the asynchronous consensus: The proposed algorithm solves the consensus problem for a certain condition C_d . In addition, it often achieves better performance if the input vector belongs to more restricted condition. However, the adaptiveness proposed in [10] is not only dependent on input vectors, but also on behavior of each process. That is, it only shows that the algorithm achieves better performance “in many executions” if the input vector belongs to more restricted condition than the given one, (hence the worst case performance is not improved). Thus, this result is not adaptive in our sense. As we mentioned in the introduction, the formal definition of the adaptive condition-based approach is first introduced in [8], where the synchronous consensus is considered. Therefore, this paper is the first one that applies the adaptive condition-based approach to the consensus problem in asynchronous systems.

As a future research issue, we remark the probabilistic aspects of the adaptive conditions, that is, considering the asynchronous consensus algorithms whose termination is “probabilistically” guaranteed according to the number of crash faults and input vectors.

Acknowledgment

This work is supported in part by a JSPS, Grant-in-Aid for Scientific Research ((B)(2)15300017), MIC, Strategic Information and Communication R&D Promotion Programme(SCOPE) and “The 21st Century Center of Excellence Program” of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

- [1] H. Attiya and J. L. Welch. Sequential consistency versus linearizability. *ACM Transactions on Computer Systems*, 12(2):91–122, 1994.
- [2] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, 1996.
- [3] D. Dolev, C. Dwork, and L. Stockmeyer. On the minimal synchronism needed for distributed consensus. *Journal of the ACM*, 34(1):77–97, 1987.
- [4] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.
- [5] R. Guerraoui. Revisiting the relationship between non-blocking atomic commitment and consensus. In *Proc. of 9th International Workshop on Distributed Algorithms(WDAG)*, volume 972 of *LNCS*, Sep 1995.
- [6] V. Hadzilacos and S. Toueg. Fault-tolerant broadcasts and related problems. In S. Mullender, editor, *Distributed Systems*, chapter 5, pages 97–145. Addison-Wesley, 1993.
- [7] M. Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13:124–149, 1991.
- [8] T. Izumi and T. Masuzawa. Synchronous condition-based consensus adapting to input-vector legality. In *Proc. of 18th International Conference on Distributed Computing(DISC)*, volume 3274 of *LNCS*, pages 16–29. Springer-Verlag, Oct 2004.
- [9] A. Mostéfaoui, S. Rajsbaum, and M. Raynal. Conditions on input vectors for consensus solvability in asynchronous distributed systems. *Journal of the ACM*, 50(6):922–954, 2003.
- [10] A. Mostéfaoui, S. Rajsbaum, M. Raynal, and M. Roy. Condition-based consensus solvability: a hierarchy of conditions and efficient protocols. *Distributed Computing*, 17(1):1–20, 2004.