EARLY-STOPPING $k$-SET AGREEMENT IN SYNCHRONOUS SYSTEMS PRONE TO ANY NUMBER OF PROCESS CRASHES

PH. RAIPIN PARVEDY    M. RAYNAL    C. TRAVERS
Early-Stopping $k$-set Agreement in Synchronous Systems Prone to any Number of Process Crashes

Ph. Raipin Parvedy* M. Raynal** C. Travers***

Systèmes communicants

Publication interne n° 1677 — Juin 2004 — 10 pages

Abstract: The $k$-set agreement problem is a generalization of the consensus problem: each process proposes a value, and each non-faulty process has to decide a value such that a decided value is a proposed value, and no more than $k$ different values are decided.

This paper presents a surprisingly simple $k$-set agreement protocol for synchronous systems where up to $t < n$ processes can crash (where $n$ is the total number of processes). The proposed protocol is the first early stopping $k$-set agreement protocol that does not impose a constraint on $t$. It allows the processes to decide and stop by $\min([f/k] + 2, [t/k] + 1)$ rounds where $f$ is the number of actual crashes ($0 \leq f \leq t$). In addition to its conceptual simplicity, the protocol has an additional noteworthy feature, namely, it is particularly efficient in common case scenarios. This comes from the fact that it is based on a mechanism that allows the processes to take into account the actual pattern of failures and not only their number, thereby allowing the processes to decide in much less than $[f/k] + 2$ rounds in a lot of cases.

Key-words: Crash failure, Efficiency, $k$-set agreement, Message passing system, Round-based computation, Synchronous system, Uniform consensus.
Accord ensembliste synchrone avec arrêt au plus tôt

Résumé : Ce rapport présente un protocole d'accord ensembliste sur \( k \) valeurs dans un système synchrone sujet au crash de processus.

Mots clés : Systèmes répartis asynchrones, Tolérance aux fautes, Crash de processus, Accord ensembliste.
1 Introduction

Context of the paper The k-set agreement problem generalizes the uniform consensus problem (that corresponds to the case $k = 1$). It has been introduced by S. Chaudhuri to investigate how the number of choices ($k$) allowed to the processes is related to the maximum number ($t$) of processes that can crash [5]. The problem can be defined as follows. Each of the $n$ processes (processors) defining the system starts with a value (called a “proposed” value). Each process that does not crash has to decide a value (termination), in such a way that a decided value is a proposed value (validity) and no more than $k$ different values are decided (agreement).

When we consider asynchronous systems, the problem can trivially be solved when $k > t$. Differently, it has been shown that there is no solution in these systems as soon as $k \leq t$ [3, 13, 19]. (The asynchronous consensus impossibility, case $k = 1$, was demonstrated before, using different techniques [9]). Several approaches have been proposed to circumvent the impossibility to solve the k-set agreement problem in asynchronous systems (e.g., probabilistic protocols [17], or unreliable failure detectors with limited scope accuracy [12, 16]).

The situation is different in synchronous systems where the k-set agreement problem can always be solved, whatever the value of $t$ (and $k$). It has also been shown that, in the worst case, the lower bound on the number of rounds (time complexity measured in communication steps) is $[t/k] + 1$ [6]. (This bound generalizes the $t + 1$ lower bound associated with the consensus problem [1, 2, 8, 15].)

Although failures do occur, they are rare in practice. For the uniform consensus problem ($k = 1$), this observation has motivated the design of early deciding synchronous protocols [4, 7, 14, 18], i.e., protocols that can cope with up to $t$ process crashes, but decide in less than $t + 1$ rounds in favorable circumstances (when there are few failures). More precisely, these protocols allow the processes to decide in $\min(f + 2, t + 1)$ rounds, where $f$ is the number of processes that crash during a run, $0 \leq f \leq t$, which has been shown to be optimal (the worst scenario being when there is exactly one crash per round).

In a very interesting way, it has very recently been shown that the early deciding lower bound for the k-set agreement problem is $\min([f/k] + 2, [t/k] + 1)$ [10]. This lower bound, not only generalizes the corresponding uniform consensus lower bound, but also shows an “inescapable tradeoff” among the number $t$ of faults tolerated, the number $f$ of actual faults, the degree $k$ of coordination we want to achieve, and the best running time achievable. It is also important to notice that, when compared to consensus, k-set agreement divides the running time by $k$ (e.g., allowing two values to be decided halves the running time).

Content of the paper While there exist several not-early deciding k-set agreement protocols [2, 6, 15] (i.e., protocols that always terminate in $[t/k] + 1$ rounds), to our knowledge only one early deciding k-set agreement protocol has been proposed [11]. This protocol assumes $t < n - k$, which means that, contrarily to what we could “normally” hope, the maximum number $t$ of processes that can crash decreases when the coordination degree $k$ increases.

We propose here a protocol that does not impose a constraint on $t$ (it assumes only $t < n$, i.e., at least one process has to be correct for the problem to be meaningful). Moreover, differently from uniform consensus protocols where a correct process that decides in a round is required to halt only in the next round, the proposed protocol allows a process to decide and halt in the very same round. This means that, instead of the “early deciding” property, the protocol provides the stronger “early stopping” property in $\min([f/k] + 2, [t/k] + 1)$.

The proposed protocol enjoys two noteworthy features. The first lies in its design simplicity (that is a first class property). Interestingly, when we take $k = 1$, we obtain a uniform consensus protocol simpler than some already proposed uniform consensus protocols (with a more general proof, based on a totally different approach). The second feature lies in its two dimensional efficiency. The first dimension concerns the size of the messages: they contain only one bit plus one proposed value. The second dimension is a “very quick decision” property, namely, except in extreme cases where the crashes are evenly distributed in the rounds, the processes decide and stop in much less than $[f/k] + 2$ rounds. The achievement of this first class efficiency property is provided by the introduction of a very simple mechanism that allows a process
to count the number of processes that, from its point of view, have crashed during the last round, whatever the number of previous crashes. This differential approach allows a process to take into account the failure pattern and not only the number of failures that occur. If, during the very first rounds, there are either few crashes or a lot of crashes, the protocol terminates very quickly. As an example, the protocol stops after only three rounds when \(xk\) (\(\forall x > 1\)) processes have crashed before the protocol starts, and less than \(k\) processes crash thereafter. The \([f/k] + 2\) lower bound is attained only in the worst scenarios where there are \(k\) crashes per round. The proposed protocol is the first \(k\)-set agreement protocol enjoying this “very quick decision” property.

Roadmap The paper consists of four parts. Section 2 presents the computation model and gives a definition of the \(k\)-set agreement problem. Section 3 presents the protocol and proves it is correct. Finally Section 4 discusses the local predicate used by the processes to early decide and stop.

2 Computation Model and \(k\)-Set Agreement

2.1 Round-Based Synchronous System

The system model consists of a finite set of processes, namely, \(\Pi = \{p_1, \ldots, p_n\}\), that communicate and synchronize by sending and receiving messages through channels. Every pair of processes \(p_i\) and \(p_j\) is connected by a channel denoted \((p_i, p_j)\).

The system is synchronous. This means that each of its executions consists of a sequence of rounds. These are identified by the successive integers 1, 2, etc. For the processes, the current round number appears as a global variable \(r\) that they can read, and whose progress is managed by the underlying system. A round is made up of three consecutive phases:

- A send phase in which each process sends messages.
- A receive phase in which each process receives messages.
  The fundamental property of the synchronous model lies in the fact that a message sent by a process \(p_i\) to a process \(p_j\) at round \(r\), is received by \(p_j\) at the same round \(r\).
- A computation phase during which each process processes the messages it received during that round and executes local computation.

The underlying communication system is assumed to be failure-free: there is no creation, alteration, loss or duplication of message.

2.2 Process Failure Model

A process is faulty during an execution if its behavior deviates from that prescribed by its protocol, otherwise it is correct. As already indicated, \(t\) is an upper bound on the number of faulty processes. A failure model defines how a faulty process can deviate from its protocol.

We consider here the crash failure model. A faulty process stops its execution prematurely. After it has crashed, a process does nothing. Let us observe that if a process crashes in the middle of a sending phase, only a subset of the messages it was supposed to send might actually be sent. As already indicated, \(t\) denotes the upper bound on the number of processes that can crash, while \(f\) denotes the number of actual crashes during a particular run. We have \(0 \leq f \leq t < n\).

2.3 The \(k\)-Set Agreement Problem

The problem has been informally stated in the Introduction: every process \(p_i\) proposes a value \(v_i\) and each correct process has to decide on a value in relation to the set of proposed values. More precisely, the \(k\)-set agreement problem is defined by the following three properties:

- Termination: Every correct process eventually decides.
3 A k-Set Agreement Protocol

This section describes a k-set agreement protocol that allows the correct processes to decide by round \( \min([f/k] + 2, [t/k] + 1) \), for \( t < n \). The protocol relies on a simple mechanism that allows a process to learn that it knows one of \( k \) smallest values currently present in the system.

3.1 Protocol Description

A process \( p_i \) invokes the k-set protocol by calling the function \( k\text{-set\_agreement}(v_i) \) where \( v_i \) is the value it proposes (Figure 1). If it does not crash, \( p_i \) terminates when it executes return \( (est_i) \) at line 4 (early stopping) or line 11, where \( est_i \) is the value it decides.

The value decided by a process \( p_i \) is the smallest proposed value it has ever seen. That value is kept in the local variable \( est_i \). The well-known "flooding strategy" (a basic technique encountered in nearly all agreement protocols [2, 15, 18]) is used to allow the processes to improve their knowledge on the smallest proposed values, namely, during each round, every active process sends the current value of \( est_i \) to all the processes.

The achievement of the early stopping property is based on the following idea. Let \( UP[r - 1] \) be the set of processes not crashed by the end of round \( r - 1 \), and \( R_i[r] \) be the set of processes from which \( p_i \) has received messages during round \( r \). Although \( p_i \) has no means to known the exact value of \( UP[r - 1] \) in the general case, as process crashes are stable we always have \( R_i[r] \subseteq UP[r - 1] \subseteq R_i[r - 1] \). More, in the particular case where \( R_i[r - 1] = R_i[r] \), \( p_i \) has received a message from each process \( p_j \in UP[r - 1] \), i.e., from all the processes that were active at the beginning of \( r \). It can then correctly conclude that it knows the smallest value among the values still present in the system at the beginning of \( r \).

Let us observe that, as the failure model is the crash model and a process \( p_i \) sends at most one message per round to each other process, we can use, instead of \( R_i[r] \), a local variable \( nb_i[r] \) counting the number of processes from which \( p_i \) has received a message during \( r \). The predicate \( R_i[r - 1] = R_i[r] \) then becomes \( nb_i[r - 1] - nb_i[r] = 0 \).

As we are interested in solving k-set agreement, it is not necessary for \( p_i \) to know the smallest value present in the system, it is sufficient for it to known one amongst the \( k \) smallest values present in the system. This knowledge can be obtained by weakening the locally evaluatable predicate \( nb_i[r - 1] - nb_i[r] = 0 \) into \( nb_i[r - 1] - nb_i[r] < k \). This weakening is due to the following observation. When \( nb_i[r - 1] - nb_i[r] < k \), \( p_i \) knows that it misses values from at most \( k - 1 \) processes in the system. In the worst case these \( k - 1 \) missing values are smaller than the value of \( est_i \) at the end of \( r \), from which we conclude that, at the end of \( r \), the value of its current estimate \( est_i \) is one of the \( k \) smallest values present in the system.

Unfortunately, the local predicate \( nb_i[r - 1] - nb_i[r] < k \) is not powerful enough to allow \( p_i \) to conclude that the other processes know it has one of the \( k \) smallest values. Consequently, \( p_i \) cannot decide and stop immediately. To be more explicit, let us consider the case where \( p_i \) has (not any of the \( k \) smallest values but) the smallest value \( v \) in the system, is the only process that knows \( v \), decides it at the end of \( r \) and then crashes by the end of \( r \). The other processes can then decide \( k \) other values as \( v \) is no longer is the system from round \( r + 1 \). An easy way to fix this problem consists in requiring \( p_i \) to proceed to \( r + 1 \) before deciding (this is similar to the way used to guarantee uniform agreement in consensus protocols). When \( nb_i[r - 1] - nb_i[r] < k \) becomes true, \( p_i \) sets a boolean \( can\_decide_i \) to true and proceeds to the next round \( r + 1 \). As, before deciding at line 4 of \( r + 1 \), \( p_i \) has first sent the pair \( (est_i, can\_decide_i) \) to all processes, any
process $p_j$ active during $r + 1$ not only knows $v$ but, as $can\_decide_i$ is true, knows also that $v$ is one of $k$ smallest values present in the system during $r + 1$.

<table>
<thead>
<tr>
<th>Function $k$-set-agreement ($\nu$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1) $est_i \leftarrow \nu$; $nb_i[0] \leftarrow \nu$; $can_decide_i \leftarrow false$;</td>
</tr>
<tr>
<td>(2) when $r = 1, 2, \ldots, \lceil t/k \rceil + 1$ do $r %$ round number $%$</td>
</tr>
<tr>
<td>(3) begin-round</td>
</tr>
<tr>
<td>(4) send $(est_i, can_decide_i)$ to all $%$ including $p_i$ itself $%$</td>
</tr>
<tr>
<td>(5) if $can_decide_i$ then return $(est_i)$ end-if</td>
</tr>
<tr>
<td>(6) let $nb_i[r]$ = number of messages received by $p_i$ during $r$;</td>
</tr>
<tr>
<td>(7) let $decide_i = \lor$ on the set of $can_decide_j$ boolean values received during $r$;</td>
</tr>
<tr>
<td>(8) $est_i \leftarrow \min{est_j$ values received during the current round $r}$;</td>
</tr>
<tr>
<td>(9) if $((nb_i[r - 1] - nb_i[r] &lt; k) \lor decide_i)$ then $can_decide_i \leftarrow true$ end-if</td>
</tr>
<tr>
<td>(10) end-round;</td>
</tr>
<tr>
<td>(11) return $(est_i)$</td>
</tr>
</tbody>
</table>

Figure 1: Early stopping synchronous $k$-set agreement: code for $p_i$ ($t < n$)

### 3.2 Proof of the Protocol

**Lemma 1** [Validity] A decided value is a proposed value.

**Proof** The proof of the validity consists in showing that an $est_i$ local variable always contains a proposed variable. This is initially true (round $r = 0$). Then, a simple induction reasoning proves the property: assuming the property is true at a round $r \geq 1$, it follows from the protocol code (lines 4 and 8), and the fact that a process receives at least the value it has sent, that the property remains true at round $r + 1$. \hspace{1cm} $\square$ **Lemma 1**

**Lemma 2** [Termination] Every correct process decides.

**Proof** The proof is an immediate consequence of the fact that a process executes at most $\lceil t / k \rceil + 1$ rounds and the computation model is the synchronous round-based computation model. \hspace{1cm} $\square$ **Lemma 2**

**Lemma 3** [Agreement] No more than $k$ different values are decided.

**Proof** Let $EST[0]$ be the set of proposed values, and $EST[r]$ the set of $est_i$ values of the processes that decide during $r$ or proceed to $r + 1$ ($r \geq 1$). We first state and prove three claims.

**Claim C1.** $\forall r \geq 0: EST[r + 1] \subseteq EST[r]$.

**Proof of the claim.** The claim follows directly from the fact that, during a round, the new value of an $est_i$ variable computed by a process is the smallest of the $est_j$ values it has received. So values can only disappear, due to the minimum function used at line 8 or to process crashes. **End of the proof of the claim C1.**

**Claim C2.** Let $p_i$ be a process such that $can\_decide_i$ is set to true at the end of $r$. Then $est_i$ is one of the $k$ smallest values in $EST[r]$.

**Proof of the claim.** Let $v$ be the value of $est_i$ at the end of $r$ ($v \in EST[r]$). If $can\_decide_i$ is set to true at the end of $r$, $nb_i[r - 1] - nb_i[r] < k$ is satisfied or $p_i$ has received a message carrying a pair $(v_1, true)$, and $v_1$ has been taken into account when computing the new value of $est_i$ at line 8 during round $r$, i.e., $v \leq v_1$. So, there is a chain of processes $j = j_0, j_{a-1}, \ldots, j_0 = i$ that has carried the boolean value $true$ to $p_i$. This chain is such that $a \geq 0$, $nb_j[r - a - 1] - nb_j[r - a] < k$ is satisfied, and any value $v'$ sent by a process participating in this chain is such that $v \leq v'$ (as each process in the chain computes the minimum of the values it has received). In particular, we have $v \leq v''$ where $v''$ is the value sent by the first process in the chain. (The case $a = 0$ corresponds to the “one-process” chain case where the local predicate is satisfied at $p_i$.) Due to
claim C1, $EST[r] \subseteq EST[r-a]$. Consequently, if $v''$ is one of the $k$ smallest values of $EST[r-a]$, $v \leq v''$ implies $v$ is one of the $k$ smallest values of $EST[r]$.

So, taking $r-a=r'$, we have to show that $n_{b_j}[r'[1]-n_{b_j}[r'] < k$ implies that the value $v'$ of $est_j$ at the end of $r'$, is one of the $k$ smallest values of $EST[r']$. As the crashes are stable, $n_{b_j}[r'[1]-n_{b_j}[r'] < k$, allows concluding that $p_j$ has received a message from all but at most $k-1$ processes that where not crashed at the beginning of $r'$. As $p_j$ computes the minimum of all the values it has received, and misses at most $k-1$ values of $EST[r']$, this means that the value $v''$ computed by $p_j$ at the end of $r'$ is one of the $k$ smallest values present in $EST[r']$. End of the proof of the claim C2.

Claim C3. Let $p_i$ be process that decides (at line 5 or 11) during the round $r$. Its boolean flag $can\_decide_i$ is then equal to true.

Proof of the claim. The claim is trivially true if $p_i$ decides at line 5. If $p_i$ decides at line 11, it decides during the last round, namely $r = \lfloor t/k \rfloor + 1$. Let us consider two cases.

- At round $r$, $p_i$ receives from a process $p_j$ a message such as $can\_decide_j = true$. In that case, $p_i$ sets $can\_decide_i$ to true at line 9, and the claim follows.

- In the other case, no process $p_j$ has decided at a round $r' < r$ (otherwise, $p_i$ would have received from $p_j$ a message such that $can\_decide_j = true$). Let $t = k x+y$ with $y < k$ (hence, $x = \lfloor t/k \rfloor = r-1$). As $n_{b_j}[r'[1]-n_{b_j}[r'] < k$ was not satisfied at each round $r'$ such that $1 \leq r' \leq x = r-1$, we have $n_{b_j}[x] \leq n-ky$. Moreover, as $p_i$ has not received from any $p_j$ a message such that $can\_decide_j$ is equal to true, if, during $r$, $p_i$ does not receive a message from $p_j$ it is because $p_j$ has crashed. So, as at most $t$ processes crash, we have $n_{b_j}[x+1] \geq n-t = n-(k x+y)$. It follows that $n_{b_j}[x]-n_{b_j}[x+1] \leq y < k$.

The claim follows.

End of the proof of the claim C3.

To prove the lemma, we now consider two cases according to the line during which a process decides.

- No process decides at line 5. This means that a process $p_i$ that decides, decides at line 11 during the last round. Due to the claim C3, such a $p_i$ has then its flag $can\_decide_i$ equal to true. Due to the claim C2, it decides one of the $k$ smallest values in $EST[\lfloor t/k \rfloor + 1]$.

- A process decides at line 5. Let $r$ be the first round during which a process $p_i$ decides at that line and $v$ be the value it decides. Since $p_i$ decides at $r$:
  - $p_i$ has set its boolean flag $can\_decide_i$ to true at the end of $r-1$, and its estimate $est_i = v$ is consequently one of the $k$ smallest values in $EST[r-1]$ (Claim C2). It follows that two processes that decide during $r$ decide values that are among the $k$ smallest values in $EST[r-1]$.
  - $p_i$ has to all the processes (line 4) the pair $(v, true)$ before deciding at line 5 during $r$. This implies that a (non-crashed) process $p_j$ that does not decide at $r$ receives $v$ at $r$ and uses it to compute its new value of $est_j$. Due to the minimum function used at line 8, it follows that, from now on, we will always have $est_j \leq v$.

Let us assume that $p_j$ does not crash. If it decides, it decides at $r' > r$, and then it necessarily decides a value $v' \leq v$. As $EST[r'] \subseteq EST[r-1]$ (claim C1), we have $v' \in EST[r-1]$. Combining $v' \leq v$, $v' \in EST[r-1]$, and the fact that $v$ is one of the $k$ smallest values in $EST[r-1]$, it follows that the value $v'$ decided by $p_j$ is one of the $k$ smallest values in $EST[r-1]$. □

**Theorem 1 [k-Set Agreement]** The protocol solves the k-set agreement problem.

**Proof** The proof follows from the Lemmas 1, 2, and 3. □

**Theorem 2 [Early Stopping]** No process halts after the round $\min(\lfloor t/k \rfloor + 2, \lfloor t/k \rfloor + 1)$.
Proof Let us first observe that a process decides and halts at the same round; this occurs when it executes return \( (est_i) \) at line 4 or 11. As observed in Lemma 2, the fact that no process decides after \( \lceil t/k \rceil + 1 \) rounds is an immediate consequence of the code of the protocol and the round-based synchronous model. So, considering that \( 0 \leq f \leq t \) processes crash, we show that no process decides after the round \( \lceil f/k \rceil + 2 \). Let \( f = xk + y \) (with \( y < k \)). This means that \( x = \lceil f/k \rceil \).

The worst case scenario is when, for any process \( p_i \) that evaluates the local decision predicate \( nb_i[r-1] - nb_i[r] < k \), this predicate is false as many times as possible. Due to the pigeonhole principle, this occurs when exactly \( k \) processes crash during each round. This means that we have \( nb_i[1] = n - k, \ldots, nb_i[x] = n - kx \) and \( nb_i[x+1] = n - f = n - (kx + y) \), from which we conclude that \( r = x + 1 \) is the first round such that \( nb_i[r-1] - nb_i[r] = y < k \). It follows that the processes \( p_i \) that execute the round \( x + 1 \) set their can.decide boolean to true. Consequently, the processes that proceed to \( x + 2 \) decide at line 5 during that round. As \( x = \lceil f/k \rceil \), they decide at round \( \lceil f/k \rceil + 2 \).

\[ \square \text{Theorem 2} \]

4 Discussion

Instead of using the local predicate \( nb_i[r-1] - nb_i[r] < k \), an early stopping protocol could be based on the local predicate \( fault_i[r] < k \) where \( fault_i[r] = n - nb_i[r] \) (the number of processes perceived as faulty by \( p_i \)). While both predicates can be used to ensure early stopping, we show here that \( nb_i[r-1] - nb_i[r] < k \) is a more efficient predicate than \( fault_i[r] < k \) (more efficient in the sense that it can allow for earlier termination). To prove it, we show the following:

- \((i)\) Let \( r \) be the first round during which the local predicate \( fault_i[r] < k \) is satisfied. The predicate \( nb_i[r-1] - nb_i[r] < k \) is then also satisfied.
- \((ii)\) Let \( r \) be the first round during which the local predicate \( nb_i[r-1] - nb_i[r] < k \) is satisfied. It is possible that \( fault_i[r] < k \) be not satisfied.

We first prove (i). As \( r \) is the first round during which \( fault_i[r] < k \) is satisfied, we have \( fault_i[r-1] \geq k \ (r - 1) \). So, we have \( fault_i[r] - fault_i[r-1] < k \) \( r - k \ (r - 1) \). Replacing the sets \( fault_i[r] \) and \( fault_i[r-1] \) by their definitions we obtain \( n - nb_i[r] - (n - nb_i[r-1]) \leq k \), i.e., \( (nb_i[r-1] - nb_i[r]) \leq k \).

A simple counter-example is sufficient to prove (ii). Let us consider a run where \( f1 > ak \) \( (a \geq 2) \) processes crash initially \( (i.e., \) before the protocol starts), and \( f2 < k \) processes crash thereafter. We have \( n - f1 \geq nb_i[1] \geq nb_i[2] \geq n - (f1 + f2) \), which implies that \( nb_i[r-1] - nb_i[r] < k \) is satisfied at round \( r = 2 \). On the other side, \( fault_i[2] \geq f1 = ak > 2k \), from which we conclude that \( fault_i[r] < k \) is not satisfied at \( r = 2 \).

This discussion shows that, while the early decision lower bound can be obtained with any of these predicates, the predicate \( nb_i[r-1] - nb_i[r] < k \) is more efficient in the sense it takes into consideration the actual failure pattern (a process counts the number of failures it perceives during a round, and not only from the beginning of the run). Differently, the predicate \( fault_i[r] < k \) considers only the actual number of failures and not their pattern (it basically always considers the worst case where there are \( k \) crashes per round, whatever their actual occurrence pattern).

References


\[ \text{This predicate is implicitly used in the proof of the (not-early deciding) k-set agreement protocol described in [15].} \]


