Distrib Comput (1995) 8: 203–210

Closed schedulers: a novel technique for analyzing asynchronous protocols*

Ronit Lubitch, Shlomo Moran**

Department of Computer Science, Technion, Haifa 32000, Israel

Received: November 1992 / Accepted: May 1994

Ronit Lubitch received her B.Sc. degree in Mathematics and Computer Science from Tel Aviv University in 1989, and her Master degree in Computer Science from the Technion, in 1993. From 1992 she is working in Graffiti Software Industries, which expertise in the design and development of advanced photo realistic rendering, and animation software systems.

Shlomo Moran received his B.Sc. and Ph.D. degrees in Mathematics from the Technion in 1975 and 1979 resp. In 1979–1981 he was at the University of Minnesota as a visiting research specialist. In 1981 he joined the Computer Science Department at the Technion, where he is now a full professor. In 1985–1986 he visited at IBM T.J. Watson Research Center. In 1992–1993 he visited at AT&T Bell Laboratories and in Centrum voor Wiskunde en Informatica, Amsterdam. His research interests include distributed computing, Combinatorics and Graph Theory, and Complexity Theory.

Summary. Analyzing distributed protocols in various models often involves a careful analysis of the set of *admissible runs*, for which the protocols should behave correctly. In particular, the admissible runs assumed by a t-resilient protocol are runs which are fair for all but at most t processors. In this paper we define *closed* sets of runs, and suggest a technique to prove impossibility results for t-resilient protocols, by restricting the corresponding sets of admissible runs to smaller sets, which are closed, as follows: For each protocol PR and for each initial configuration c, the set of admissible runs of PR which start from c defines a tree in a natural way: the root of the tree is the empty run, and each vertex in it denotes a finite prefix of an admissible run; a vertex u in the tree has a son v iff v is also a prefix of an admissible run, which extends u by one atomic step.

The tree of admissible runs described above may contain infinite paths which are not admissible runs. A set of admissible runs is *closed* if for every possible initial configuration c, each path in the tree of admissible runs starting from c is also an admissible run. Closed sets of runs have the simple combinatorial structure of the set of paths of an infinite tree, which makes them easier to analyze. We introduce a unified method for constructing closed sets of admissible runs by using a model-independent construction of closed *schedulers*, and then mapping these schedulers to closed sets of runs. We use this construction to provide a unified proof of impossibility of consensus protocols.

Key words: Closed schedulers – Asynchronous protocols – Admissible runs

1 Introduction

A distributed decision task is a distributed task in which every processor eventually makes an irreversible decision step, such that the eventual decision values of the processors must satisfy the input/output relation that specifies the task [12, 2]. One of the more challenging problems in distributed computing is the characterization of the decision tasks that can be solved in a completely asynchronous environment, in the presence of crash (fail stop) failures, under which processors may stop participating in the protocol prematurely. A protocol that solves such a task in

* A preliminary extended version of this paper appeared in the Proceedings of 6-th International Workshop on Distributed Algorithms, Haifa, November 1992

** This work was supported in part by the Technion V.P.R. fund. Part of this research was conducted while this author was visiting at AT&T Bell Labs at Murray Hill and at CWI, Amsterdam

Correspondence to: S. Moran

the presence of at most t crash failures is called t-resilient. A somewhat simplified version of this question can be formulated as the following decision problem:

Input: A decision task T for n processes, specified by its (finite) input/output relation, and a number t, $1 \leqq t < n$.
Property: There is a t-resilient protocol for T.
The research on the above problem was initiated in the fundamental paper [7], which proved the non-existence of 1-resilient consensus protocols. Subsequent papers studied this question for other tasks, like approximate consensus [6], k-set consensus [5], renaming [1], and others. A general decision procedure for the above question when $t = 1$ (and n is arbitrary) was given in [2]. For the case where $t > 1$ only partial results are known (e.g., [4, 14]). Notable among the papers which study t-resilient protocols for $t > 1$ are recent results which relate this question to properties of high dimensional topological complexes [3, 8, 13]. In particular, this technique is used there to prove tight impossibility results on the k-set cosensus problem and on the renaming problem. However, a general decision procedure for the above question is not known even for any *fixed* n and t such that $1 < t < n$.

The difficulty of this problem does not seem to depend on the specific model of computation studied (i.e., shared memory or message passing), but more on the inherent difficulty of coordination between processors in a totally asynchronous environment, and in particular on the impossibility to distinguish between faulty processors and processors which are very slow, but in working order. Consequently, it is possible to have a t-resilient protocol for a given task, with the following unpleasant property: The number of steps that may be executed by the protocol, when started from a certain initial configuration, before it fulfills its task, is unbounded.

In this paper we propose an approach for analyzing asynchronous protocols which avoids the difficulty mentioned above. In this approach, we restrict the set of *admissible* runs, for which the protocol is required to behave correctly, to a set of a simple structure, which we call "closed". A closed set of runs has the property that if a protocol is guaranteed to fulfill some task in each run in it, then it is guaranteed to fulfill that task within a fixed number of steps. We use this approach to provide an alternative proof technique for the impossibility of t-resilient consensus protocols in various models. Specific applications of this technique, some of which generalize the classical impossibility result of [7] in an interesting way, appear in [11]. We believe that the closed sets constructed here capture the fundamental properties possessed by the sets of *all* admissible runs, and hence can be used for proving other properties of t-resilient protocols.

1.1 Protocols and runs

A *distributed system* consists of a set of n ($n \geqq 2$) asynchronous processors $\{p_1, \ldots, p_n\}$, modeled as (not necessarily finite) state machines, and of some means of communication among the processors (e.g., shared memory or message passing).

Each processor p acts according to a deterministic transition function t_p. The transition function is described by the set of *atomic steps* which can be taken by the processor. An atomic step consists of a possible change of the processor's state, and of reading and/or writing from the communication means. A *protocol* for a given distributed system is a set of n transition functions, one per processor.

A *configuration* of the system is a description of the system at some moment. It consists of the internal state of each processor and of the contents of the communication means. An *initial configuration* is one in which each processor is in an *initial state*, and the communication means contains some default initial values.

For each processor p and for each configuration c, there is a (finite) set of atomic steps that can be taken by p from the configuration c. A *run* of a protocol is an *infinite* sequence of atomic steps that can be taken in turn starting from some (initial) configuration c. Each atomic step is performed by one of the processors, and brings the system to a subsequent configuration. We say that a run r is *applicable* to a configuration c if it is a run that may start from the configuration c. If r is applicable to c, then for every (finite) prefix r' of r, the configuration resulted from applying r' to c is denoted by $\sigma(c, r')$.

1.2 Closed sets of admissible runs

A distributed protocol is required to fulfill a certain task w.r.t. a specified set of runs, which we call the set of *admissible runs*. Thus, the correctness of a protocol depends not only on the task it should accomplish, but also on the set of admissible run which are assumed. For example, there are protocols which are correct in a synchronous environment but not in an asynchronous one, and there are protocols which are correct when all processors are non-faulty but are incorrect when processors are subject to failures. In both these examples, protocols which are correct for a restricted set of admissible runs become incorrect when the set of admissible runs is extended.

Let R be a set of runs, and c be a given configuration. We denote by R^c the set of all runs in R which are applicable to c. R^c defines an infinite directed tree, $T(R^c)$, in a natural way: the root of $T(R^c)$ is the empty run, and each vertex in it represents a finite prefix of a run in R^c; a vertex u in $T(R^c)$ has a son v iff v represents a prefix of a run in R^c, which extends u by one atomic step. When there is no ambiguity, we will identify vertices in $T(R^c)$ with the prefixes of runs they represent.

For an infinite tree T, $Paths(T)$ denotes the set of infinite directed paths in T. Note that for each set of runs R and for each configuration c, $Paths(T(R^c))$ is a set of runs which are applicable to c, and $Paths(T(R^c)) \supseteq R^c$. However $Paths(T(R^c))$ may contain runs which are not in R^c. For instance, it is possible that for every $r \in R^c$, every processor takes an atomic step infinitely often in r, but $Paths(T(R^c))$ contains a run in which only one processor is activated forever.

A set R of runs is *closed* iff for every possible configuration c, each path in $T(R^c)$ is a run in R^c, i.e.: $Paths(T(R^c)) = R^c$. Closed sets of runs appear to be much easier to analyze than other sets of runs, since they have the simple combinatorial structure possessed by the set of paths of an infinite tree of bounded degree. One specific useful property which is possessed by such sets, is the following: if it is given that each run in R^c eventually satisfies certain property, then it is guaranteed that this property is achieved within a constant number of steps. This property is proved in the following lemma:

Lemma 1.1. *Let R be a closed set of runs of some protocol PR. Assume that for some predicate Pred and for some configuration c, every run $r \in R^c$ has a prefix r' which satisfies Pred. Then there is a constant M_c, such that every run $r \in R^c$ has a prefix of length at most M_c which satisfies Pred.*

Proof. Let $T = T(R^c) = (V, E)$. Define:

$V' = \{v \in V \mid \text{each prefix } v' \text{ of } v \text{ does not satisfy } Pred\}$

$E' = \{e = (v, u) \in E \mid v, u \in V'\}$

By the definition, $T' = (V', E')$ is a subgraph of T, and for each $v \in V'$ the directed path in T from the root to v is in T'. Hence T' is a directed tree. If $|V'| < \infty$, then $M_c = 1 + \max\{depth(v) | v \in V'\}$ satisfies the requirement of the lemma. Otherwise, T' is an infinite tree, the degree of its vertices is bounded, so by König's Infinity Lemma [9] there is an infinite directed path r in T'. This means that r is a run in R^c, all whose prefixes do not satisfy *Pred*, a contradiction. □

Unfortunately, in many cases the set of admissible runs which is of interest is not closed. The most notable example is probably the sets of admissible runs for t-resilient protocols, which must guarantee correct behavior in all runs in which at most t processors are subject to crash (fail-stop) failures. Admissible runs of such protocols are runs which are *fair* with respect to at least $n - t$ processors. The exact definition of "fair" depends on the specific model studied, but under all common definitions, the set of all $n - t$ *fair* runs of a given protocol is not closed for $0 \leqq t \leqq n - 2$.

In this paper we suggest a unified method for proving impossibility results concerning t-resilient protocols, and exemplify this technique on consensus protocols. In this method, we prove the impossibility result with respect to a proper subset of the set of all $n - t$ fair runs, which is closed, by using Lemma 1.1 above. The definition of this subset is based on a purely combinatorial construction, which is independent on the specific model studied. In [11] we demonstrate our technique by using it to prove impossibility of t-resilient consensus protocols in some variants of the shared memory model and of the message passing model, some of which are non-trivial generalization of the fundamental impossibility result of [7].

1.3 Summary of results

In the next section we define the consensus problem and present a general, model-independent, proof of non-existence of consensus protocols. This proof assumes the existence of closed sets of runs which satisfy certain properties. In Sect. 3 we provide a combinatorial construction of closed *schedulers*, which are the main tool we use to construct the closed sets of runs needed for our proofs, and in Sect. 4 we describe the way this construction is applied to specific models of asynchronous computations. An example of applying this general technique for proving impossibility of 1-resilient read/write consensus protocol in the shared memory model is given in Sect. 5.

2 Consensus protocols

A *consensus protocol* is a protocol in which each processor p has a binary input register in_p and an output register out_p. The initial content of the output register is $\perp$. A consensus protocol is correct w.r.t. a given set of admissible runs R, if in each run $r \in R$, some non-faulty processor decides on a binary value v, by writing it in the output register, such that

1) consistency: all the processors which decide, decide on the same value v.
2) nontriviality: v is the input of at least one of the processors.

A t-resilient consensus protocol is a protocol which is correct w.r.t. the set of all $n - t$-fair runs (i.e., at most t processors are faulty in them), which are applicable to some initial configuration.

Let PR be a consensus protocol, R^c be a set of runs of PR applicable to an initial configuration c, and $T(R^c)$ be the tree associated with R^c as described in Sect. 1.2. Each vertex $v \in T(R^c)$ represents a finite prefix r' of some run r in R^c.

Let u be a vertex in $T(R^c)$, and let D_u be the set of decision values of the runs in R^c which are extensions of u. u is *bivalent in* $T(R^c)$ if $|D_u| = 2$. u is *univalent in* $T(R^c)$ if $|D_u| = 1$, and we say that u is 0-*valent in* $T(R^c)$ or 1-*valent in* $T(R^c)$ according to the corresponding decision value. Note that if PR is a t-resilient protocol and all the runs in R^c are $n - t$-fair runs, then each vertex in $T(R^c)$ is either bivalent or univalent in $T(R^c)$. When the tree $T(R^c)$ is obvious from the context, we will not mention it in the terms univalent, bivalent and 0(1)-valent.

2.1 Proving-impossibility of consensus by using closed sets of runs

In this subsection we present a model-independent impossibility proof of t-resilient consensus protocols, for $t \geqq 1$, which is based on the existence of closed sets of $n - t$-fair runs, which satisfy certain properties. We start with some definitions.

Throughout the paper, Q denotes a subset of $\{1, \ldots, n\}$, and for such a Q, P_Q denotes the set of processors $\{p_i \mid i \in Q\}$. For sequences x and y, $x \cdot y$ denotes the concatenation of x and y.

Definitions. A P_Q-*run* is a run in which the set of non-faulty processors is included in P_Q. Runs r_1 and r_2 are P_Q-*equivalent* if for each $p \in P_Q$, p makes the same sequence of atomic steps in r_1 and in r_2.

Let $T_1 = T(R^{c_1})$ and $T_2 = T(R^{c_2})$ be the trees of the sets of admissible runs applicable to configurations c_1 and c_2 resp. Let v_1 be a vertex in T_1 and let v_2 be a vertex in T_2. We say that v_1 and v_2 are P_Q-*similar* if there exists P_Q-runs r_1 and r_2 which are P_Q-equivalent, such that $(v_1 \cdot r_1)$ is in

$Paths(T_1)$ and $(v_2 \cdot r_2)$ is in $Paths(T_2)$ (recall that $v_i \cdot r_i$ denotes the concatenation of the finite sequence v_i with r_i).

In our proof, we define for a given t-resilient consensus protocol PR and for each initial configuration c, a subset of the set of $n-t$-fair runs of PR starting from c, denoted as $R^c_{n,t}$, and we let $R_{n,t}$ be the union $\bigcup_c R^c_{n,t}$, taken over all initial configurations c. $R_{n,t}$ is a closed set of $n-t$-fair runs, and it satisfies the following properties:

initial similarity: Let c_1, c_2 be initial configurations, and let $Q \subseteq \{1, \ldots, n\}$, s.t. $|Q| \geqq n-t$. If each processor $p \in P_Q$ has the same input in c_1 and in c_2, then the roots of $T(R^{c_1}_{n,t})$ and of $T(R^{c_2}_{n,t})$ are P_Q-similar.

siblings similarity: Let c be an initial configuration, and let $u, v, w \in T(R^c_{n,t})$ s.t. v and w are sons of u. Then for some $Q \subseteq \{1, \ldots, n\}$, $|Q| \geqq n-t$, there is a descendant v' of v and a descendant w' of w such that v' and w' are P_Q-similar.

Theorem 2.1. *Let $t \geqq 1$ be a given integer. Then there is no consensus protocol which is correct w.r.t. a closed set of $n-t$-fair runs $R_{n,t}$ which satisfies the* initial similarity *and* siblings similarity *properties.*

Proof. Assume by the way of contradiction, that PR is a consensus protocol which is correct w.r.t. a set of runs $R_{n,t}$ which satisfies the above properties, where $t \geqq 1$.

We derive a contradiction in three steps:

Step 1. Proof of the existence of an initial configuration c_0, s.t. the root $T(R^{c_0}_{n,t})$ is bivalent. Assume by the way of contradiction that for each initial configuration c, the root of $T(R^c_{n,t})$ is univalent. Let c_0 be the initial configuration in which the value of each input register in_p is 0, and c_1 be the initial configuration in which the value of each input register in_p is 1. By the nontriviality property for consensus protocol, the root of $T(R^{c_0}_{n,t})$ is 0-valent and the root of $T(R^{c_1}_{n,t})$ is 1-valent. Hence, there must be initial configurations c_a and c_b which differ only in the initial value in_{p_i} of a single processor p_i, the root v_a of $T(R^{c_a}_{n,t})$ is 0-valent and the root v_b of $T(R^{c_b}_{n,t})$ is 1-valent. Let $Q = \{1, \ldots, i-1, i+1, \ldots, n\}$. Since $t \geqq 1$ and $|Q| = n-1$, the *initial similarity* property implies that there are P_Q-runs $r_a \in R^{c_a}_{n,t}$ and $r_b \in R^{c_b}_{n,t}$, which are P_Q-equivalent. r_a is an $n-t$-fair P_Q-run, and hence there must be $p \in P_Q$ s.t. p eventually reaches a decision state in r_a. Since v_a is 0-valent, p must decide on 0 in r_a. r_a and r_b are P_Q-equivalent, so p takes on r_b the same steps as in r_a, and therefore p decides on 0 also in r_b. This contradicts the 1-valency of v_b. Therefore, there exists an initial configuration c_0, s.t. the root of $T(R^{c_0}_{n,t})$ is bivalent.

Step 2. Proof of the existence of vertices $u, v, w \in T = T(R^{c_0}_{n,t})$, v and w are sons of u, s.t. v is 0-valent and w is 1 valent.

For $v \in T$, we define $Pred(v)$ to be *true* if v is univalent and *false* if v is bivalent. Since every run $r \in R^{c_0}_{n,t}$ is $n-t$-fair, each such run r has a prefix r' s.t. the vertex representing r' in $T(R^{c_0}_{n,t})$ satisfies *Pred*. By Lemma 1.1, there is a constant M_c s.t. every vertex of depth $\geqq M_c$ in T satisfies *Pred*. Assume that M_c is as small as possible. Since by *Step 1* the root of $T(R^{c_0}_{n,t})$ is bivalent (i.e. does not satisfy *Pred*), $M_c \geqq 1$ and hence there exists a bivalent vertex, u, of maximal possible depth. This implies that u has one son v which is 0-valent in T and another son w which is 1-valent in T.

Step 3. Let v and w be as in Step 2. By the *siblings similarity*, there is a vertex v' which is a descendant of v, and a vertex w' which is a descendant of w, s.t. for some $Q \subseteq \{1, \ldots, n\}$, $|Q| \geqq n-t$, v' and w' are P_Q-similar. Then there are P_Q-runs r_1 and r_2 which are P_Q-equivalent, such that both $(v' \cdot r_1)$ and $(w' \cdot r_2)$ are in $Paths(T)$. Like in *Step 1*, since v' is 0-valent, some processor $p \in P_Q$ decides on 0 in r_1. Since p takes the same steps in both runs, p decides on 0 also in r_2. But this is a contradiction, since w' is 1-valent. $\square$

In order to apply Theorem 2.1 to prove the non-existence of t-resilient consensus protocols in specific models, we have to construct closed sets of $n-t$-fair runs $R_{n,t}$, which satisfy the *initial similarity* and *siblings similarity* properties. This construction is carried out in two steps: First, we define and construct combinatorial objects, called *closed schedulers*, and prove that they satisfy certain properties. Then we describe how these closed schedulers are used to construct the sets of runs $R_{n,t}$ in various models.

3 Closed schedulers

Let I be a (finite) set of integers. A *schedule* $s = (s_1, s_2, \ldots)$ over I, denoted *I-schedule*, is an infinite sequence of integers from I; $s^{(l)} = (s_1, \ldots, s_l)$ denotes the prefix of the first l elements of s ($s^{(0)} = \varepsilon$). A schedule s is *fair* for an integer i, if i appears in it infinitely often. s is fair for a subset Q of I if it is fair for every $i \in Q$. s is *m-fair* for $1 \leqq m \leqq n$ if it is fair for a subset Q where $|Q| \geqq m$. Note that each schedule is 1-fair.

A *scheduler* S over I is a set of schedules as above. S is *m-fair* if all the schedules in it are *m-fair*.

Each scheduler S defines an infinite directed tree $T(S)$ in a natural way, as follows: The vertices of $T(S)$ are all the finite prefixes of schedules in S, and a vertex u is the father of a vertex v iff $v = u \cdot (i)$ for some i. The edge (u, v) is marked with i. In this way, each schedule $s \in S$ is an infinite path in $T(S)$.

Let $Paths(T(S))$ be, as before, the set of infinite paths in $T(S)$. Note that $Paths(T(S))$ is a scheduler, and that for each scheduler S, $Paths(T(S)) \supseteq S$. A scheduler S is *closed* if $Paths(T(S)) = S$, i.e. all the infinite paths in $T(S)$ are in S.

Examples

- For each $n \in N$, the set S_n of all 1-fair schedules over $\{1, \ldots, n\}$ (which is the set of all schedules over $\{1, \ldots, n\}$) is closed.
- For each $n \geqq 2$, $0 \leqq t \leqq n-2$, let $S_{n,t}$ denote the set of all $n-t$-fair schedules over $\{1, \ldots, n\}$. $S_{n,t}$ is not closed: $\forall i \in N$ the schedule $(\underbrace{1, \ldots, 1}_{i \text{ times}}, 1, \ldots, n-t, 1, \ldots, n-t, \ldots)$ is $n-t$-fair, so the vertex $(\underbrace{1, \ldots, 1}_{i \text{ times}})$ is in $T(S_{n,t})$. This implies that the schedule $(1, 1, 1, \ldots)$, which

is not $n-t$-fair, is in $Paths(T(S_{n,t}))$. In fact, $T(S_{n,t}) = T(S_n)$ for all $n \geqq 2$, $0 \leqq t \leqq n-2$.
- Each finite scheduler (i.e. a finite set of schedules) is closed.
- Let T be an infinite directed tree with no leaves whose vertices are finite sequences of integers from $\{1, \ldots, n\}$, and a vertex u is the father of a vertex v iff $v = u \cdot (i)$ for some i (the edge (u, v) is marked with i). Then the scheduler $S = Paths(T)$ is closed.

3.1 *Construction of closed and fair schedulers*

In this subsection we define for each $n \geqq 2$, $0 \leqq t \leqq n-1$, a tree $T_{n,t}$ s.t. each infinite path in $T_{n,t}$ is $n-t$-fair. So the scheduler $\mathscr{S}_{n,t} = Paths(T_{n,t})$ is $n-t$-fair and closed. In the next subsection we prove some combinatorial properties of $T_{n,t}$, which are used in the impossibility proofs based on our construction. For $t = n-1$, $T_{n,n-1} = T(S_n)$, where S_n is the set of all $\{1, \ldots, n\}$-schedules. Below we present the construction of $T_{n,t}$ for $0 \leqq t \leqq n-2$.

Each vertex in $T_{n,t}$ will have either $t+1$ or $t+2$ sons. Informally, the sons of a vertex $u \in T_{n,t}$ are determined by the suffix of the last $n-t$ elements in (the sequence representing) u, denoted as $suf_{n-t}(u)$. In order to generalize the definition also for sequences of length $< n-t$, we take $suf_{n-t}(u)$ to be the sequence of the last $n-t$ elements in the sequence $(1, \ldots, n-t) \cdot u$ (i.e., the sequence $(1, \ldots, n-t)$ concatenated with u). Also, when there is no ambiguity, we will omit the subscript $n-t$ and denote this suffix by $suf(u)$. For a finite sequence s', we denote by $SUF(s')$ the set of elements in $suf(s')$.

For $0 \leqq t \leqq n-2$, the tree $T_{n,t}$ is defined inductively as follows:

1. The empty sequence ε is the root of $T_{n,t}$.
2. Let u be a vertex in $T_{n,t}$, and assume that $suf(u) = (s_1, \ldots, s_{n-t})$, where $s_i \neq s_j$ for $i \neq j$ (that is: all the elements in $suf(u)$ are distinct). A vertex u with this property is said to be *normal*. Let $i_1, \ldots, i_t$ be the integers in $\{1, \ldots, n\} \setminus SUF(u)$. Then the sons of u are $u \cdot (i_1), \ldots, u \cdot (i_t), u \cdot (s_1), u \cdot (s_2)$.
3. Let u be a vertex in $T_{n,t}$, and assume that $suf(u) = (s_1, \ldots, s_{n-t-1}, s_1)$, where $s_i \neq s_j$ for $i \neq j$ (that is: the first and last elements are equal and all the others are distinct). A vertex u with this property is said to be *special*. Let $i_1, \ldots, i_{t+1}$ be the integers in $\{1, \ldots, n\} \setminus SUF(u)$. Then the sons of u are $u \cdot (i_1), \ldots, u \cdot (i_{t+1})$.

For the above definition to be complete, we need to show that every vertex in $T_{n,t}$ must be either normal or special. This follows from the fact that $suf(\varepsilon) = (1, \ldots, n-t)$, and hence ε is a normal vertex, and from Lemma 3.1 below.

Lemma 3.1. *Each normal vertex in $T_{n,t}$ has $t+2$ sons, exactly one of which is special and the others are normal, and each special vertex in $T_{n,t}$ has $t+1$ sons which are all normal.*

Proof. Follows immediately from the definition of $T_{n,t}$. □

The definition of $T_{n,t}$ guarantees that for each schedule s in $Paths(T_{n,t})$, in each subsequence of $n-t+1$ consecutive elements of s, at least $n-t$ elements are distinct. This implies that the closed scheduler $S_{n,t} = Paths(T_{n,t})$ *is* $n-t$-fair.

Example. Let $n=5$, $t=2$. The vertex $u_1 = (1,2,3)$ is a normal vertex in $T_{5,2}$, and its sons are $(1,2,3,4)$, $(1,2,3,5)$, $(1,2,3,1)$ and $(1,2,3,2)$. The vertex $u_2 = (1,2,3,2)$ is a special vertex (the only special son of u_1) and its sons are $(1,2,3,2,1)$, $(1,2,3,2,4)$, $(1,2,3,2,5)$, which are all normal vertices.

3.2 *Similarity properties*

In this subsection we prove that the trees $T_{n,t}$ defined above satisfy certain properties, which are needed to guarantee that the *initial similarity* and *siblings similarity* properties are satisfied by the sets of runs constructed in the various models.

Definition. For each $v \in T_{n,t}$, $T_{n,t}(v)$ is given by:

$$T_{n,t}(v) = \{u \mid v \cdot u \in T_{n,t}\}$$

i.e. $T_{n,t}(v)$ is the subtree of $T_{n,t}$ which consists of v and all its descendants, when omitting the prefix v from all the vertices. Note that for each $v \in T_{n,t}$, the scheduler $Paths(T_{n,t}(v))$ is $n-t$ fair and closed.

Lemma 3.2. *Let u be a vertex in $T_{n,t}$, and let $Q \subseteq \{1, \ldots, n\}$, be of cardinality $\geqq n-t$. Then $Paths(T_{n,t}(u))$ contains a Q-schedule.*

Proof. Let $Q = \{i_1, \ldots, i_m\}$, and let $suf(u) = (s_1, \ldots, s_{n-t})$. Assume that the elements in Q are ordered so that for each k, $1 \leqq k < m$, if i_k is in $suf(u)$, then for every l s.t. $k < l \leqq m$, i_l also appears in $suf(u)$, and the last occurrence of i_k in $suf(u)$ precedes the last occurrence of i_l in $suf(u)$[1]. Since $m \geqq n-t$, this implies that for $1 \leqq k \leqq m$, i_k does not occur in $(s_{k+1}, \ldots, s_{n-t})$. Hence, every $n-t$ successive elements in the sequence $(s_2, \ldots, s_{n-t}, i_1, \ldots, i_m)$ are distinct. It follows that the periodical schedule $(i_1, \ldots, i_m, i_1, \ldots)$ is in $Paths(T_{n,t}(u))$. □

Definitions. Vertices u and v in $T_{n,t}$ are *equivalent* iff $T_{n,t}(u) = T_{n,t}(v)$. u and v are *Q-similar*, $Q \subseteq \{1, \ldots, n\}$, $|Q| \geqq n-t$, iff there exists a Q-schedule s s.t. $s \in Paths(T_{n,t}(v)) \cap Paths(T_{n,t}(u))$.

The existence of pairs of vertices which are Q-similar for some $Q \subseteq \{1, \ldots, n\}$ is used in all our impossibility proofs.

Lemma 3.3. *For each $u, v \in T_{n,t}$:*

(a) *If $suf(u) = suf(v)$, then u and v are equivalent.*

(b) *Let $Q \subseteq \{1, \ldots, n\}$, $|Q| \geqq n-t$. If there exists a sequence $s' \in Q^{n-t}$, s.t. both $u \cdot s'$ and $v \cdot s'$ are in $T_{n,t}$, then u and v are Q-similar.*

Proof

(a) We have to show that for each schedule s, s is in $T_{n,t}(u)$ iff it is in $t_{n,t}(v)$. Let $s = (s_1, s_2, \ldots)$, be given. An

[1] Note that if u is special, then one integer appears twice in $suf(u)$

easy induction shows that for each $l \geqq 0$, the prefix $(s_1, \ldots, s_l)$ of s is in $T_{n,t}(u)$ iff it is in $T_{n,t}(v)$. This proves (a).

(b) By Lemma 3.2, $Paths(T_{n,t}(u \cdot s'))$ contains a Q-schedule, say s. Hence $Paths(T_{n,t}(u))$ contains the Q-schedule $s' \cdot s$. By (a) above and the fact that $suf(u \cdot s') = suf(v \cdot s') = s'$, $s' \cdot s$ is also in $Paths(T_{n,t}(v))$. This proves (b). □

The next technical claim follows directly from the inductive definition of $T_{n,t}$, and its proof is left to the reader.

Claim 3.4. *For each $n \geqq 2, 0 \leqq t \leqq n-2$, let v_n be a normal vertex in $T_{n,t}$, $suf(v_n) = (s_1, \ldots, s_{n-t})$ and v_s be a special vertex in $T_{n,t}$, $suf(v_s) = (t_1, \ldots, t_{n-t-1}, t_1)$.*

(a) *Let $s' = suf(v_n)$. Then, $v_n \cdot s'$ is a normal vertex in $T_{n,t}$.*

(b) *Let $s' = (s_1, \ldots, s_{n-t-2}, s_{n-t}, s_{n-t-1})$ (i.e. s' is obtained by switching the last two elements in $suf(v_n)$). Then, $v_n \cdot s'$ is a normal vertex in $T_{n,t}$.*

(c) *Let $s' = (l, s_1, \ldots, s_{n-t-2}, s_{n-t})$ where $l \in \{1, \ldots, n\} \setminus SUF(v_n)$. Then, $v_n \cdot s'$ is a normal vertex in $T_{n,t}$.*

(d) *Let $s' = (l, t_2, \ldots, t_{n-t-1}, t_1)$ where $l \in \{1, \ldots, n\} \setminus SUF(v_s)$. Then, $v_s \cdot s'$ is a normal vertex in $T_{n,t}$.*

(e) *Let v be a vertex in $T_{n,t}$ and s' a sequence of length $n-t$ s.t. $v \cdot s'$ is a normal vertex in $T_{n,t}$. Let s'' be a sequence obtained by replacing elements in s' by distinct integers from $\{1, \ldots, n\} \setminus \{SUF(v) \cup SUF(v \cdot s')\}$. Then, $v \cdot s''$ is normal vertex in $T_{n,t}$.*

Lemma 3.5. *Let $n \geqq 2$, $0 \leqq t \leqq n-1$. Then for each $u \in T_{n,t}$, and for each i, j s.t. both $u \cdot (i)$ and $u \cdot (j)$ are in $T_{n,t}$, it holds that both $u \cdot (i,j)$ and $u \cdot (j,i)$ are in $T_{n,t}$, and for each $Q \subseteq \{1, \ldots, n\}$, $|Q| \geqq n-t$,*

1. *$u \cdot (i,j)$ and $u \cdot (j,i)$ are Q-similar.*
2. *If $t \geqq 1$ and $i \notin Q$ then $u \cdot (i,j)$ and $u \cdot (j)$ are Q-similar.*
3. *If $t \geqq 2$ and $i, j \notin Q$ then $u \cdot (i)$ and $u \cdot (j)$ are Q-similar.*

Proof. If $t = n-1$ then $T_{n,t}$ is the complete n-ary tree, in which all the vertices are equivalent, and hence the lemma holds trivially. Thus, we assume that $0 \leqq t \leqq n-2$. Let u be a vertex in $T_{n,t}$, $Q \subseteq \{1, \ldots, n\}$, $|Q| \geqq n-t$. It is easy to see that if $u \cdot (i)$ and $u \cdot (j)$ are vertices in $T_{n,t}$, then $u_1 = u \cdot (i,j)$ and $u_2 = u \cdot (j,i)$ are normal vertices in $T_{n,t}$, where the only difference between $suf(u_1)$ and $suf(u_2)$ is the order of the last two elements. We now prove each of the three claims in the lemma.

1. Let $u_1 = u \cdot (i,j)$ and $u_2 = u \cdot (j,i)$. By Lemma 3.3(b) it suffices to show that there is a sequence $s' \in Q^{n-t}$, s.t. both $u_1 \cdot s$ and $u_2 \cdot s'$ are in $T_{n,t}$. Assume first that $suf(u_1) \in Q^{n-t}$. In this case, we take $s' = suf(u_1)$. Then, by Claim 3.4(a), $u_1 \cdot s'$ is a normal vertex in $T_{n,t}$, and by Claim 3.4(b), $u_2 \cdot s'$ is a normal vertex in $T_{n,t}$.

If $suf(u_1) \notin Q^{n-t}$, we let s' be the sequence obtained from $suf(u_1)$ by replacing the elements in $suf(u_1)$ which are not in Q by distinct elements from $Q \setminus SUF(u_1)$ (this is possible since $|Q| \geqq |SUF(u_1)|$). Since $SUF(u_1) = SUF(u_1 \cdot suf(u_1)) = SUF(u_2 \cdot suf(u_1))$, by Claim 3.4(e), both $u_1 \cdot s'$ and $u_2 \cdot s'$ are normal vertices in $T_{n,t}$. Hence, by Lemma 3.3(b), u_1 and u_2 are Q-similar.

2. Let $u_1 = u \cdot (j)$, $u_2 = u \cdot (i)$ and let $u_3 = u \cdot (i,j)$. As in 1. above, it suffices to show that there is a sequence $s' \in Q^{n-t}$, s.t. both $u_1 \cdot s'$ and $u_3 \cdot s'$ are in $T_{n,t}$. Let $suf(u_1) = (s_1, \ldots, s_{n-t-1}, j)$ and thus $suf(u_2) = (s_1, \ldots, s_{n-t-1}, i)$. Assume first that $SUF(u_1) \setminus \{i\} \subseteq Q$. In constructing s' we distinguish between two cases:

Case 1. Both u_1 and u_2 are normal. In this case $i \notin SUF(u_1)$, and hence $SUF(u_1) \subseteq Q$. Let $s' = suf(u_1)$. Then $u_1 \cdot s'$ is in $T_{n,t}$ by Claim 3.4(a), and $u_3 \cdot s'$ is in $T_{n,t}$ by Claim 3.4(c).

Case 2. Not case 1. Then $s_1 \in \{i,j\}$, and hence $|SUF(u_1) \cup SUF(u_2) \setminus \{i\}| < n-t \leqq |Q|$. Since $i \notin Q$, there is an $l \in Q$, which is not in $SUF(u_1) \cup SUF(u_2)$. In this case we let $s' = (l, s_2, \ldots, s_{n-t-1}, j)$. Then $u_3 \cdot s'$ is in $T_{n,t}$ by Claim 3.4(c). If $s_1 = j$ then $u_1 \cdot s'$ is in $T_{n,t}$ by Claim 3.4(d), else $u_1 \cdot s'$ is in $T_{n,t}$ by application of (a) and then (e) of Claim 3.4.

Assume now that $SUF(u_1) \setminus \{i\} \nsubseteq Q$. We replace in each of the two cases above the sequence s' by a sequence s'', which is obtained by replacing the elements in s' which are not in Q by distinct elements from Q which are not in s' (again, this is possible since $|Q| \geqq n-t$). Since i does not occur in s'' and $[SUF(u_3) \setminus \{i\}] \subseteq [SUF(u_1) \setminus \{i\}] \subseteq SUF(u_1 \cdot s') = SUF(u_3 \cdot s')$, we have by Lemma 3.4(e) that both $u_1 \cdot s''$ and $u_3 \cdot s''$ are normal vertices of $T_{n,t}$, and by Lemma 3.3(b) they are Q-similar.

3. Let $u_1 = u \cdot (i)$ and $u_2 = u \cdot (j)$. Again, it suffices to show that there is a sequence $s' \in Q^{n-t}$, s.t. both $u_1 \cdot s'$ and $u_2 \cdot s'$ are in $T_{n,t}$. Let $suf(u_1) = (s_1, \ldots, s_{n-t-1}, i)$ and let $suf(u_2) = (s_1, \ldots, s_{n-t-1}, j)$. Assume first that $SUF(u_1) \setminus \{i,j\} \subseteq Q$. At least one out of u_1, u_2 is normal, so assume that u_1 is normal. We let $s' = (m, \ldots, s_{n-t-1}, l)$, where (i) $l \in Q \setminus SUF(u_1)$, and (ii) if $s_1 \neq j$ then $m = s_1$, else $m \neq l$, and $m \in Q \setminus SUF(u_1)$. Then by Claim 3.4(e), both $u_1 \cdot s'$ and $u_2 \cdot s'$ are in $T_{n,t}$.

Assume now that $SUF(u_1) \setminus \{i,j\} \nsubseteq Q$. As in the previous cases, we construct a sequence s'' by replacing the elements in s' above which are not in Q by distinct elements from Q. By Claim 3.4(e) both $u_1 \cdot s''$ and $u_2 \cdot s''$ are normal vertices in $T_{n,t}$, and by Lemma 3.3(b) they are Q-similar. □

4 Mapping schedules to runs

In this section we describe a general technique which, for a given distributed model dm and a t-resilient protocol PR in dm, use the closed schedulers constructed in the previous section, to define a closed set of runs of PR which satisfies the *initial similarity* and *siblings similarity* properties.

Let PR be a protocol for n processors in the given model dm. Then we define a mapping:

$$M_{dm}: C \times S_n \to R$$

where C is the set of configurations of PR, S_n is the set of schedules over $\{1, \ldots, n\}$, R is the set of runs of PR, and for each $c \in C$ and $s \in S_n$, $M_{dm}(c, s)$ is a run which is applicable to c. Intuitively, this mapping maps each occurrence of an integer i in a schedule s to an atomic event $e(i)$, associated with processor P_i. This mapping should guarantee that if s is fair for i, then the run $M_{dm}(c, s)$ is fair for p_i. By varying the way in which $e(i)$ depends on P_i, various impossibility results are obtained (see [11]).

Let $s = (s_1, \ldots, s_i, \ldots)$. First, we define by induction for each finite prefix $s' = (s_1, \ldots, s_i)$ of s, $M_{dm}(c, s')$ to be a sequence of i steps. Then, $M_{dm}(c, s)$ is defined to be the infinite sequence of steps obtained by this induction.

For each initial configuration c_0, we map the canonical (n, t) scheduler $\mathscr{S}_{n,t}$ to a set of runs $R^{c_0}_{n,t}$ of PR:

$$R^{c_0}_{n,t} = \{M_{dm}(c_0, s) \mid s \in \mathscr{S}_{n,t}\}$$

and we consider the tree associated with $R^{c_0}_{n,t}$ as described in Sect. 1.1, $T(R^{c_0}_{n,t})$.

In this way, $T_{n,t}$ and $T(R^{c_0}_{n,t})$ are isomorphic. The $n-t$-fairness of schedules in $\mathscr{S}_{n,t}$ induces $n-t$-fairness of the runs in $R^{c_0}_{n,t}$, and therefore $R^{c_0}_{n,t}$ is a closed set of $n-t$-fair runs. The results in Sect. 3.2, and the isomorphism of $\mathscr{S}_{n,t}$ and $T(R^{c_0}_{n,t})$ guarantees that the *initial similarity* and *siblings similarity* properties are satisfied by $T(R^{c_0}_{n,t})$. In the next section we demonstrate how an impossibility proof for a specific model, using the set of runs constructed here, is carried out.

5 Impossibility of read/write 1-resilient consensus protocols

In this section we demonstrate how to apply the proof technique of Sect. 2 to prove impossibility of t-resilient consensus protocols in a specific model. In [11] we apply the technique to some variations of the shared memory and message passing models. Here, we prove the impossibility of 1-resilient consensus protocols in the read/write shared memory model.

We consider the standard read/write model, as defined in [10], where processes communicate via a set of shared registers. A process may atomically read or atomically write a shared register in one atomic step. A process is non-faulty in a run if it takes infinitely many steps in it (and it is faulty otherwise).

Let PR be a protocol for n processes in the shared memory model sm. Then, we define a mapping:

$$M_{sm}: C \times \mathscr{S}_{n,t} \to R$$

which maps each pair of a configuration $c \in C$ and a schedule $s = (s_1, s_2, \ldots) \in \mathscr{S}_{n,t}$ to a run $M_{sm}(c, s)$, in which, for each $i \in N$, the i-th step is taken by p_{s_i}. Note that in the model considered, given a configuration c, an atomic step is completely determined by specifying the active process p. This guarantees that M_{sm} is well defined.

For each initial configuration c, $R^c_{n,t} = \{M_{sm}(c, s) \mid s \in \mathscr{S}_{n,t}\}$. The mapping M_{sm} defines isomorphism from $T_{n,t}$ onto $T(R^c_{n,t})$. We denote the image of a vertex $u \in T_{n,t}$ under this isomorphism by u_c.

By Theorem 2.1 it suffices to prove that $R_{n,1}$ is a closed set of $n-1$-fair runs which satisfies the initial similarity and siblings similarity properties. By the definition of the mapping M_{sm} and the fact that $\mathscr{S}_{n,t}$ is an $n-t$-fair scheduler, the set $R_{n,1}$ is a closed set of $n-1$-fair runs. In order to prove that it satisfies also the *initial similarity* and *siblings similarity* properties, we need one more definition and lemma:

Definition. Configurations c_1 and c_2 are *P_Q-equivalent*, for $Q \subseteq \{1, \ldots, n\}$, if all the shared registers have the same values in c_1 and in c_2, and each processor $p \in P_Q$ is in the same internal state in c_1 and in c_2.

Lemma 5.1. *Let c_1 and c_2 be P_Q-equivalent configurations, and let s be a Q-schedule. Then the runs $r_1 = M_{sm}(c_1, s)$ and $r_2 = M_{sm}(c_2, s)$ are P_Q-runs which are P_Q-equivalent.*

Proof. First, observe that if c and d are P_Q-equivalent configurations, then for every $l \in \mathbb{N}$, $M_{sm}(c, s^{(l)}) = M_{sm}(d, s^{(l)})$, and the configurations $\sigma(c, M_{sm}(c, s^{(l)}))$ and $\sigma(d, M_{sm}(d, s^{(l)}))$ are P_Q-equivalent.

For each integer l, let $r_1^{(l)} = M_{sm}(c_1, s^{(l)})$ and $r_2^{(l)} = M_{sm}(c_2, s^{(l)})$. Using the above observation, a straightforward induction on l shows that the configurations $\sigma(c_1, r_1^{(l)})$ and $\sigma(c_2, r_2^{(l)})$ are P_Q-equivalent. It follows that the runs r_1 and r_2 are P_Q-runs which are P_Q-equivalent (and, in fact, are identical). $\square$

The *initial similarity* property follows from Lemma 5.1, Lemma 3.2 and the definition of the mapping M_{sm}. The next lemma proves the *siblings similarity* property.

Lemma 5.2. *Let $u, v = u \cdot (i)$, $w = u \cdot (j)$ be vertices in $T_{n,1}$, and let u_c, v_c, w_c be the corresponding vertices in $T(R^c_{n,1})$. Then there is a descendant v'_c of v_c and a descendant w'_c of w_c such that v'_c and w'_c are P_Q-similar, for some $Q \subseteq \{1, \ldots, n\}$, $|Q| \geqq n-1$.*

Proof. By Lemma 5.1, it suffices to find a descendant v' of v and a descendant w' of w s.t. v' and w' are Q-similar and $\sigma(c, v'_c)$ and $\sigma(c, w'_c)$ are P_Q-equivalent.

There is an atomic step a taken by p_i, and an atomic step b taken by p_j, such that $v_c = u_c \cdot (a)$ and $w_c = u_c \cdot (b)$. Let reg_1 be the register that p_i accesses in a, and reg_2 be the register that p_j accesses in b.

Case 1. One of the two steps a, b is a read step.
Suppose w.l.o.g. that the step a taken by p_i is a read step. Let $Q = \{1, \ldots, n\} \setminus \{i\}$, $w' = w = u \cdot (j)$ and $v' = v \cdot (j)$. Then, $\sigma(c, w'_c)$ and $\sigma(c, v'_c)$ are P_Q-equivalent, and by Lemma 3.5(2), $u \cdot (j)$, $v \cdot (j)$, are Q-similar. Therefore w'_c and v'_c are P_Q-similar.

Case 2. Both steps a, b are write steps, $reg_1 \neq reg_2$.
Let $w' = w \cdot (i)$, $v' = v \cdot (j)$, and let $Q = \{1, \ldots, n\}$. Then, $\sigma(c, w'_c)$ and $\sigma(c, v'_c)$ are P_Q-equivalent, and by Lemma 3.5(1), $w \cdot (i)$ and $v \cdot (j)$ are Q-similar. Therefore w'_c and v'_c are P_Q-similar.

Case 3. Both steps a, b are write steps, $reg_1 = reg_2$.
Let $w' = w = u \cdot (j)$, $v' = v \cdot (j)$, and let $Q = \{1, \ldots, n\} \setminus \{i\}$. Then, $\sigma(c, w'_c)$ and $\sigma(c, v'_c)$ are P_Q-equivalent, and like in *Case 1*, w'_c and v'_c are P_Q-similar. $\square$

Thus, we conclude the following:

Theorem 5.3. [10] *There is no 1-resilient read/write consensus protocol.*

6 Conclusion and further research

In this paper we introduced the concept of closed sets of runs, which are sets of runs that can be described as the paths of an infinite tree of bounded degree. Then we introduced the concept of closed schedulers, and presented a unified, model independent technique to construct closed sets of runs of t-resilient protocols by using closed schedulers.

The sets constructed by our technique preserve many of the properties possessed by the sets of *all* runs of t-resilient protocols; in particular, given any finite prefix of a run in this set, it is still impossible to distinguish between faulty and slow processes in this run. We believe that this makes these sets a convenient tool for proving properties of such protocols. To demonstrate this, we used these sets to provide unified proofs of the impossibility of t-resilient consensus protocols.

The full applicability of closed sets of runs, and in particular of the sets constructed by the closed schedulers $\mathcal{S}_{n,t}$ introduced in this paper, is yet to be explored. It is anticipated that the simple combinatorial structure of these schedulers will make them a useful tool for studying further problems related to t-resilient protocols.

Acknowledgement. We would like to thank an anonymous referee for his insightful comments, which helped in improving the presentation of our results.

References

1. Attiya H, Bar-Noy A, Dolev D, Peleg D, Reischuk R: Renaming in an asynchronous environment. J ACM 37(3): 524–548 (1990)
2. Biran O, Moran S, Zaks S: A combinatorial characterization of the distributed 1-solvable tasks. J Algorithm 11: 420–440 (1990) A preliminary versions appeared in Proc 7th ACM Symp on Principles of Distributed Computing, August 1988
3. Borowski E, Gafni E: Generalized FLP impossibility result for t-resilient asynchronous computations. In: Proc 25th ACM Symp on Theory of Computing, May 1993
4. Bridgland M, Watro R: Fault-tolerant decision making in totally asynchronous distributed systems. In: Proc 6th ACM Symp on Principles of Distributed Computing, pp 52–63, 1987
5. Chaudhuri S: More choices allow more faults: set consensus problems in totally asynchronous systems. Inf Comput 105(1): 132–158 (1993)
6. Fekete A: Asynchronous approximate agreement. In: Proc 6th ACM Symp on Principles of Distributed Computing, 149–170, 1987
7. Fischer MJ, Lynch NA, Paterson MS: Impossibility of distributed consensus with one faulty process. J ACM 32(2): 374–382 (1985)
8. Herlihy M, Shavit N: The asynchronous computability theorem for t-resilient tasks. In: Proc 25th ACM Symp on Theory of Computing, May 1993
9. König D: Theorie der endlichen und unendlichen graphen. Leipzig, 1936 (reprinted by Chelsea, 1950)
10. Loui CM, Abu-Amara H: Memory requirements for agreement among unreliable asynchronous processes. Adv Comput Res 4: 163–183 (1987)
11. Lubitch R, Moran S: Closed schedulers: constructions and applications of consensus protocols. In: Proceedings of 6-th International Workshop on Distributed Algorithms, pages 11–34, November 1992. Revised Version in TR #796, Department of Computer Science, Technion, January 1994
12. Moran S, Wolfsthal Y: An extended impossibility result for asynchronous complete networks. Inf. Process Lett 26: 141–151 (1987)
13. Saks M, Zaharoglou F: Wait-free k-set agreement is impossible: the topology of public knowledge. In: Proc 25th ACM Symp on Theory of Computing, May 1993
14. Taubenfeld G, Katz S, Moran S: Impossibility results in the presence of multiple faulty processes. Inf Comput, 1994. Preliminary version appeared in: Veni Madhavan CE (eds) 9th FCT-TCS Conference, Bangalore, India, December, 1989, Lect Notes Comput Sci, vol 405, pp 109–120, Springer, Berlin Heidelberg New York 1989