Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Self-stabilizing mutual exclusion on directed graphs

D. Alstein, J.H. Hoepman, B.E. Olivier and P.I.A. van der Put

Computer Science/Department of Algorithmics and Architecture

CS-R9513 1995

Report CS-R9513
ISSN 0169-118X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Self-Stabilizing Mutual Exclusion on Directed Graphs

Dick Alstein
wsinda@win.tue.nl
Eindhoven University of Technology

Jaap-Henk Hoepman
jhh@cwi.nl
CWI Amsterdam

Bryan E. Olivier
olivier@fwi.uva.nl
University of Amsterdam

Pascale I. A. van der Put
pascale@cs.ruu.nl
Utrecht University

Abstract

This paper investigates the complexity of self-stabilizing mutual exclusion protocols for distributed systems, where processors communicate through shared memory according to a strongly connected directed communication graph. Tchuente's approach of covering a network with one directed cycle is taken as point of departure. This protocol requires $O(n^{2n})$ states per processor together with some preprocessing. By coalescing states a protocol requiring only $O(n^2)$ states per processor—still requiring preprocessing—is derived. Finally two protocols based on spanning trees are considered. Combining these protocols with a self-stabilizing spanning tree protocol yields two $O(n^3m)$—where m is the maximal degree of a processor—states per processor protocols that require knowledge of processor identities. This report concludes with a full proof of the coalesced states protocol in Lamport's Temporal Logic of Actions.

CR Subject Classification (1991): C.2.4, C.4, D.1.3, D.2.1, D.2.4, D.4.1, D.4.4, D.4.5, F.3.1
Keywords & Phrases: self-stabilization, mutual exclusion, distributed control, fault tolerance, shared memory, directed communication graphs, TLA, formal correctness proofs
Note: The last three authors were partially supported by the Dutch foundation for scientific research (NWO) through NFI Project ALADDIN, under contract number NF 62-376

1. Introduction

In a distributed system multiple processors co-operate to perform certain tasks. A prerequisite for achieving co-operation are protocols that implement distributed control, i.e. protocols that reach or maintain a global objective despite the fact that processors can only access partial, local information. Because distributed control protocols are extensively used to implement

distributed systems, many researchers have tried to implement them efficiently. Mutual exclusion protocols constitute a well-know example. These protocols pass a single privilege fairly among the processors in the system. A processor requiring exclusive access to a critical resource must request the privilege and wait for it to arrive. After a processor is done with the privilege, it forwards the privilege to some other processor requesting it. Many solutions to the mutual exclusion problem exist, but most of them assume that the system operates flawlessly at all times.

One can add failure resilience to distributed control protocols by making them self-stabilizing. In abstract terms a self-stabilizing protocol will, when started on a system in an arbitrary initial configuration, reach a certain desirable—legitimate—configuration after a finite number of steps. The legitimate configurations characterize the global objective that has to be maintained in the system. Once such a legitimate configuration is reached, the protocol will keep the system in a legitimate configuration forever. For a mutual exclusion protocol for instance, the legitimate configurations would be those in which at most one processor holds the privilege. In practice this means that a self-stabilizing protocol is resilient to transient errors that change the state of some processors, but will leave the processors themselves in working order. Since, as far as a self-stabilizing protocol is concerned, the erroneous state just after an error might also have been its initial state, self-stabilizing protocols will recover from such errors.

Most research on self-stabilizing protocols has focussed on systems in which part of the state of a processor can be read by other, neighbouring, processors as if they communicated through shared memory. An important complexity measure in this model is the number of states per processor used by such a protocol. The distributed system is—as usual—modelled by a communication graph containing the processors as vertices and a directed edge between two processors if the state of the first can be read by the second. This graph is considered undirected if for every edge there exists an edge between the same processors in the reverse direction as well. It is assumed that this communication graph is strongly connected.

Self-stabilization was introduced by Dijkstra [Dij74, Dij82] in the context of the mutual exclusion problem. Dijkstra gives an $O(n)$-state[1] self-stabilizing mutual exclusion (SSME) protocol for directed rings, a four-state protocol for an undirected chain, and a three-state protocol for undirected rings. All his protocols are non-uniform: the protocol of at least one processor differs from the protocol of the other processors to break the symmetry of the system. He aslo proved that for rings of non-prime size no uniform SSME protocol can exist. Burns and Pachl [BP89] complemented this result with a uniform SSME protocol for rings of prime size.

Elaborating on Dijkstra's results, Tchuente [Tch81] gives some lower bounds on the number of states per processor for any self-stabilizing mutual exclusion (SSME) protocol on a chain or a ring. He also extends Dijkstra's results to SSME protocols for arbitrary graphs in two ways: by covering a graph with a ring[2], and by covering a graph with chains and rings that all share at least one common node. The resulting protocols require knowledge of the coverings,

[1] Unless stated otherwise, an x-state protocol should be read as x-state-per-processor protocol.

[2] Here a ring is a cyclic path in the graph that may visit nodes more than once, and may traverse edges more than once.

and therefore require some preprocessing.

For undirected graphs, many SSME protocols have been developed. There the focus has been on optimizing the time needed to reach a legitimate configuration and the time needed to pass the privilige from one processor to the other. Of special interest to us is a SSME protocol for arbitrary undirected graphs based on spanning trees, by Dolev et. al. [DIM93]. They introduce the notion of a fair protocol combination to combine a self-stabilizing spanning tree protocol with a SSME algorithm for tree-shaped graphs to obtain a SSME protocol for arbitrary graphs. The resulting protocol only requires one a priori special node and can even handle topological changes, as long as the diameter and the maximal degree of the graph do not exceed the limits assumed by the protocol.

To summarize: space efficient self-stabilizing mutual exclusion protocols for undirected graphs exist, but for arbitrary directed graphs only Tchuente's covering with a directed ring can be used. This requires $O(n^{2n})$ states per processor in the worst case. Our contribution is twofold. First we reduce the number of states needed to cover a graph with a virtual ring to $O(n^2)$ by coalescing states on the processors. Recall that this cover still needs to be computed beforehand. Secondly, we present two new protocols using roughly $O(n^2)$ states per per processor that do not require preprocessing, but do require that processors know their identities. Instead of a virtual ring, this protocol uses a spanning tree which can be maintained by a separate self-stabilising protocol requiring an additional $O(mD)$—where m is the maximum degree of any node in the graph and D is the diameter of the graph—states per processor. Merging both protocols using fair protocol combination yield two $O(n^2Dm)$ states per processor protocols for self-stabilizing mutual exclusion on arbitrary, strongly connected, directed graphs.

The structure of our paper is as follows. We begin by describing the model, give a precise statement of the problem and introduce some notation in section 2. Section 3 discusses Tchuente's results from [Tch81], and is followed by section 4, in which we improve his solution of covering a graph with a ring. Our protocols based on a spanning tree are presented in section 5. Finally, section 6 discusses our results, and suggests ways of further research. Due to space considerations, the proofs in this paper will only be sketched.

2. Model & notation

We consider arbitrary strongly connected distributed systems modelled by a directed communication graph G=(V,E), with *nodes* $v \in V = \{0, \ldots, n-1\}$ and directed *edges* $uv \in E \subseteq V \times V$. If $uv \in E$, then u is an *in-neighbour* of v and v an *out-neighbour* of u. Internally, nodes can distinguish between neighbours. We define the set of in-neighbours $\mathsf{In}(v) = \{u \mid uv \in E\}$. Nodes in the system communicate with each other by reading each others state. Node v can read the state of node u if and only if $uv \in E$.

A *configuration* $C \in \mathcal{C}$ of the system is the Cartesian product $\prod_{v \in V} S_v$ of *states* $S_v \in \mathcal{S}_v$ of all nodes $v \in V$. Here $\mathcal{S}_v$ is the set of all possible states of node v, and $\mathcal{C}$ is the set of all possible configurations. We write $C[v]$ for the state of node v in configuration C. A *program* for a node v describes the steps it can perform, based on the state of v and the state of its in-neighbours $\mathsf{In}(v)$. We model a program for a node v as a deterministic state-transition function $\delta_v : \mathcal{S}_v \times \prod_{u \in \mathsf{In}(v)} \mathcal{S}_u \mapsto \mathcal{S}_v$. A *step* of v in configuration C changes the

state of v to $\delta_v(C) = \delta_v(C[v], (C[u])_{u \in \mathsf{In}(v)})$, yielding configuration C' with $C'[v] = \delta_v(C)$ and $C'[u] = C[u]$ for all $u \neq v$. We write $C \xrightarrow{v} C'$. A *protocol* consists of a program for each node $v \in V$.

All protocols assume the existence of a *central daemon* [Dij74]. In this model, nodes take a step one at a time according to a fair *schedule* $(v_0 v_1 \ldots)$, such that each $v \in V$ occurs infinitely often in the schedule. An *initial configuration* C_0 and a schedule $(v_i)_{i \geq 0}$ induce an *execution* $(C_0 C_1 \ldots)$ such that $C_i \xrightarrow{v_i} C_{i+1}$ for all $i \geq 0$. This execution is fair iff the schedule is fair. We consider normalized executions from which all void steps $C_i \xrightarrow{v_i} C_{i+1}$ with $C_i = C_{i+1}$ are removed.

Let $\mathcal{L}$ be the set of *legitimate configurations*, i.e. the set of configurations the distributed systems should be in. Then a protocol is *self-stabilizing* to $\mathcal{L}$ if for all executions $C_0 C_1 \ldots$ there exists an i such that $C_i \in \mathcal{L}$, and moreover, for all $C' \in \mathcal{L}$ and $C'' \in \mathcal{C}$ with $C' \rightarrow C''$, we have $C'' \in \mathcal{L}$ as well. In other words, a protocol is self-stabilizing to $\mathcal{L}$ if in every execution a configuration in $\mathcal{L}$ is reached, and if $\mathcal{L}$ is closed under transitions of the protocol.

The performance of a self-stabilizing protocol is usually measured by the number of states per processor used by the protocol, and the rate of convergence, i.e. the worst case number of non-void steps needed to reach a legitimate configuration. The state-per-processor measure does not count any overhead incurred by communicating the state to the neighbour. This cost is considered part of the topology. Of course, any critical assessment of a self-stabilizing protocol should also consider the complexity measures related to the original problem to be solved by the protocol. For self-stabilizing mutual exclusion protocols this would include a bound on the time a processor may have to wait before it receives the privilege.

The problem to be solved can now be stated as follows: design a protocol, self-stabilizing to a set $\mathcal{L}$ of legitimate configurations in which at most one node is privileged. Each node should be able to determine whether it is privileged or not based on its own state and that of its in-neighbours. A further requirement is that during an execution of the protocol, each node gets privileged infinitely often.

We use the following notational conventions. The state of a processor is split into several named fields. A field *name* of the state of v is written $name[v]$. Its value in a configuration C is denoted $C.name[v]$. The program δ_v for a node v is denoted as a sequence of statements to transform the state of v to its new state. Assignment is denoted by :=, alternatives by **if -then -else** statements.

3. Tchuente's approach

Tchuente considers undirected (network) graphs whereas we consider directed graphs. Some of Tchuente's ideas are applicable to directed graphs (as will be shown).

3.1 Virtual ring

Dijkstra's protocol for mutual exclusion applies to a unidirectional ring. A variant of this protocol applies to an arbitrary strongly connected directed graph.

For each node v there exists a path from v to any other node in an arbitrary strongly connected directed graph G. So it is possible to find a walk in G which contains all nodes in

G. [3] Such a walk forms a *virtual ring*. Each node in G simulates (i.e. executes the program of) one or more nodes in the (unidirectional) virtual ring G^r. And each virtual node in G^r is simulated by one node in G. So mutual exclusion in the virtual ring implies mutual exclusion in the (corresponding original) arbitrary strongly connected directed graph.

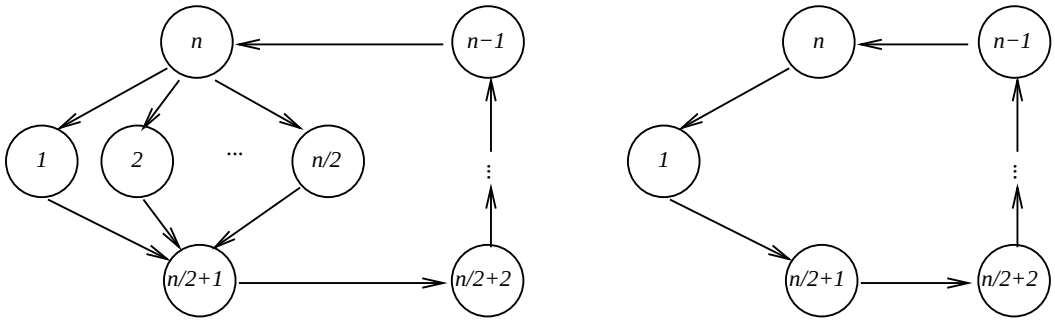


Figure 1: Virtual Ring and Construction from Subgraphs (Worst case)

The best case example occurs when the original graph is a simple ring. The virtual ring (with minimal number of nodes) in G is G itself. A worst case example is presented in figure 1. The virtual ring in G is $(1, n/2+1, n/2+2, \ldots, n, 2, n/2+1, n/2+2, \ldots, n, \ldots, n/2, n/2+1, n/2+2, \ldots, n)$. The virtual ring contains about $n^2/4$ nodes. It can be shown that there exists a virtual ring $G^r = (V^r, E^r)$ for an arbitrary strongly connected directed graph $G = (V, E)$, where $|V| = n$, s.t. $n \leq |V^r| \leq n^2$.

The state of a node v in graph G should represent the states of all nodes in G^r which v simulates. Tchuente's representation is simple. The state s_v of node v is a tuple. This tuple has as fields the states s_u of all (distinct) nodes u simulated by v. In worst case n nodes simulate $\Theta(n^2)$ nodes. Each simulated node u has a state consisting of a variable which has $\Theta(n^2)$ values in worst case. Since an arbitrary node v simulates $O(n)$ nodes u this results in $O(n^{2n})$ states per processor ($(n^2)^n = n^{2n}$).

3.2 Construction from subgraphs

We describe another idea of Tchuente as follows. The mutual exclusion problem in a subgraph is solved by means of a subprotocol. An protocol for the entire graph is constructed by combining the subprotocols. Combining the subprotocols requires a so called *central node* which is common to all subgraphs. Each node in G simulates one or more nodes in one or more subgraphs G^c. And each node in a G^c is simulated by one node in G. The central node chooses which subprotocol is 'active'. So mutual exclusion in and among the subgraphs implies mutual exclusion in the (corresponding original) arbitrary strongly connected directed graph. (Hierarchies of subprotocols may be considered.) Tchuente's subprotocols are variants of Dijkstra's protocol. Each node in a subgraph has a state consisting of a single variable which has a constant number of values (instead of about n' values where n' is the number of

[3] Consider the nodes v_1 upto including v_n. There exist a path from v_1 to v_2, from v_2 to v_3 and a path from v_n to v_1, forming a (non-minimal) walk.

nodes in the subgraph).

The best case example occurs when the original graph as a whole resembles a subgraph. The protocol amounts to the corresponding subprotocol. Figure 1 can be viewed as a worst case example. Node n is the central node. A subgraph in the form of a ring is presented in the same figure. There exist $\Omega(n)$ subgraphs of $\Omega(n)$ length in this example. In worst case n nodes simulate $\Theta(n^2)$ nodes in a directed graph. Since an arbitrary node v simulates $O(n)$ nodes u this results in $O(c^n)$ states per processor. The worst case may be less bad in an undirected graph. In worst case n nodes simulate an amount of nodes ranging between $\Omega(n \cdot \log n)$ and $O(n^2)$.

4. Reducing the number of states

As in Tchuentes approach we start with a walk through the directed strongly connected graph $G = (V, E)$ visiting all nodes, each node is visited at most $n-1$ ($n = |V|$) times and the walk returns where it started. We will call each visit of a node an *occurrence* of this node. Some occurrence of some node is selected as the start of the walk and is assigned a number 0. It will be called the *root*. If the length of the walk is n_R then all occurrences are assigned a number in $V_R \triangleq \{0, \ldots, n_R - 1\}$ increasing along the walk. The function $f : V_R \to V$ gives for each occurrence its corresponding node.

$$\begin{array}{rcl} \mathsf{Sim}(v) & \triangleq & \{v_R \in V_R | f(v_R) = v\} \\ \mathsf{First}(v) & \triangleq & \min \mathsf{Sim}(v) \\ \mathsf{Last}(v) & \triangleq & \max \mathsf{Sim}(v) \end{array}$$

$\mathsf{Sim}(v)$ is the set of occurrences of node v, $\mathsf{First}(v)$ is the first and $\mathsf{Last}(v)$ is the last occurrence of node v.

In Tchuentes protocol each occurrence runs the program of a processor in Dijkstra's protocol, ending up with a total number of states per processor of $O(n_R^n)$ It is our goal to reduce this number to $O(n^2)$. In our protocol we also have a program for each occurrence of a node, but all occurrences of a node share two variables t and d, each with $O(n)$ values. Variable t, called *ticket*, corresponds to Dijkstra's variable. Instead of associating a ticket with each occurrence, all occurrences of a node share the ticket of the node. The variable d, called *distance*, is used to tell which occurrences have this ticket and which don't. The distance at a node v is equal to one of the occurrences of node v: all occurrences with a smaller or equal number as this distance have the ticket, the others do not. Formally occurrence v_R *has the ticket* ($t[f(v_R)]$) iff $d[f(v_R)] \geq v_R$. It should be noted that the number of values of distance is $O(n)$, because a node has at most $n-1$ occurrences.

The first occurrences of nodes follow Dijkstra's protocol, the others take care of propagating the tickets. The root will generate new tickets when its own ticket returns at its *predecessor* (in the walk), i.e.

The program for $v_R = 0$:
if $t[f(n_R - 1)] = t[f(0)] \wedge d[f(n_R - 1)] \geq n_R - 1$ **then**
    $t[f(0)]] \;:=\; t[f(0)] + 1 \bmod (n+1)$; $d[f(0)] \;:=\; 0$
end if

The other first occurrences will wait for their predecessor to have a different ticket and then copy this ticket

The program for $v_R \neq 0$ and $v_R = \mathsf{First}(v)$ for some $v \in V$:
if $t[f(v_R - 1)] \neq t[f(v_R)] \wedge d[f(v_R - 1)] \geq v_R - 1$ **then**
$\quad t[f(v_R)]] \;:=\; t[f(v_R - 1)];\; d[f(v_R)] \;:=\; v_R$
end if

The non-first occurrences just propagate their ticket by increasing the distance to include themselfses

The program for $v_R \neq 0$ and $v_R > \mathsf{First}(v)$ for some $v \in V$:
if $t[f(v_R - 1)] = t[f(v_R)] \wedge d[f(v_R - 1)] \geq v_R - 1 \wedge d[f(v_R)] < v_R$ **then**
$\quad d[f(v_R)] \;:=\; v_R$
end if

The behaviours must satisfy the following fairness requirement: if the central daemon gives a turn to a processor, this processor selects one of its occurrences and executes the corresponding action, as described above. A processor is required to eventually execute an occurrence that is *enabled*, i.e. its guard holds. The daemon is required to eventually select a processor that has an occurrence enabled.

In the legitimate configurations the walk can be divided in two parts. In the first part all occurrences have the same ticket as the root, we call them Green. In the second part all occurrences either have the previous ticket of the root or are waiting for the ticket of the root.

$$
\begin{array}{rcl}
\mathsf{Green}(v_R) & \triangleq & t[f(v_R)] = t[f(0)] \wedge d[f(v_R)] \geq v_R \\
\mathsf{Red}(v_R) & \triangleq & (d[f(v_R)] \geq v_R \Rightarrow t[f(v_R)] = t[f(0)] - 1 \bmod (n+1)) \wedge \\
 & & (d[f(v_R)] < v_R \Rightarrow t[f(v_R)] = t[f(0)]) \\
\mathsf{Legal}(v_R) & \triangleq & \Big(\forall u_R : u_R \in V_R \wedge u_R \leq v_R :: \mathsf{Green}(v_R)\Big) \wedge \\
 & & \Big(\forall u_R : u_R \in V_R \wedge u_R > v_R :: \mathsf{Red}(v_R)\Big) \\
\mathsf{Legitimate} & \triangleq & \Big(\exists v_R : v_R \in V_R :: \mathsf{Legal}(v_R)\Big)
\end{array}
$$

In a legitimate configuration, if $\mathsf{Legal}(v_R)$ holds, only the occurrence $v_R+1 \bmod n_R$ is enabled (theorem B.11). It will therefore eventually be executed (theorem B.12). After performing its action $\mathsf{Legal}(v_R + 1 \bmod n_R)$ holds (theorem B.18). A processor can be privileged if any of its occurrences is enabled. Or, if we want all processors to be privileged in a more round-Robin way, if its first occurrence is enabled. Or, if we want control over the sequence of privileges, the first occurrence is enabled and its predecessor has a ticket with the number of this node. From the above it follows that legitimate configurations are preserved and the privilege is passed on. If all occurrences, except the root (number 0), are disabled, then all occurrences are Green (induction on V_R), therefore $\mathsf{Legal}(n_R - 1)$ holds (lemma B.20) and the root is enabled. So in any configuration some occurrence is enabled (corollary B.21).

The stabilization period consists of two phases. In the first phase the root will generate new tickets until no other occurrence has the same ticket. This starts the second phase, in

which the root will not generate a new ticket until all occurrences have copied its ticket, at which moment $\mathsf{Legal}(n_R - 1)$ holds. For both phases we will define a function that decreases with the execution of any enabled occurrence, unless we can start the next phase. Consider the legitimate configurations as the third phase. The ranges of these bounding functions are finite and therefore eventually a legitimate configuration is reached. The sum of the sizes of these ranges is an upper-bound on the number of changes of configuration before reaching a legitimate configuration. For this reason we try to keep the ranges small.

First we exhibit a function Chaos, decreasing with the execution of any enabled occurrence except the root. This function will be used in both phases. For convenience the root will not be considered an occurrence in the following and execution of any occurrences implies it is enabled. We write $\big(\#v : D(v) :: P(v)\big)$ to denote the number of v s.t. $D(v)$ and $P(v)$ (domain and predicate) hold.

$$
\begin{array}{rcl}
W(v) & \triangleq & f(\mathsf{First}(v) - 1 \bmod n_R) \\
\mathsf{Weight}(v) & \triangleq & 1 + \big(\sum u : u \in V \setminus \{f(0)\} \wedge W(u) = v :: \mathsf{Weight}(u)\big) \\
\mathsf{Distance}(v) & \triangleq & \big(\#v_R : v_R \in \mathsf{Sim}(v) :: v_R < d[v]\big) \\
M & \triangleq & \big(\max\, v : v \in V :: |\,\mathsf{Sim}(v)|\big) \\
\mathsf{Rank}(v) & \triangleq & -\,\mathsf{Distance}(v) + \begin{cases} 0 & \text{if } t[v] = t[W(v)] \vee v = f(0) \\ M \cdot \mathsf{Weight}(v) & \text{otherwise} \end{cases} \\
\mathsf{Chaos} & \triangleq & \big(\sum v : v \in V :: \mathsf{Rank}(v)\big)
\end{array}
$$

Chaos consists of the sum of contributions, $\mathsf{Rank}(v)$, of the nodes. This contribution in turn consists of two parts: its distance and its weight. $-\,\mathsf{Distance}(v)$ decreases with the execution of a non-first occurrence of v (lemma B.29) and increases with at most $M - 1$ if the first occurrence is executed (lemma B.30). The sum of the weight-parts decreases with at least M with the execution of a first occurrence (lemma B.28) and stays unchanged otherwise (lemma B.27). From this it follows that Chaos decreases with at least 1 by the execution of an occurrence (lemma B.32).

It can be shown that $0 \leq \mathsf{Distance}(v) < |\,\mathsf{Sim}(v)|$ (lemma B.22) and therefore it follows that $n - n_R \leq \big(\sum v : v \in V :: -\,\mathsf{Distance}(v)\big) \leq 0$ (lemma B.23). It can also be shown that $0 \leq \big(\sum v : v \in V \wedge v \neq f(0) :: \mathsf{Weight}(v)\big) \leq \frac{1}{2}n(n-1)$ (lemma B.26), concluding (lemma B.31)

$$n - n_R \leq \mathsf{Chaos} \leq M \cdot \frac{n(n-1)}{2}$$

Configurations in the second phase look like legitimate configurations, the walk can also be split in two parts and all occurrences in the first part are Green. The occurrences in the second part however need not be Red as long as they are not Green.

$$\mathsf{Stable}(v_R) \quad \triangleq \quad \big(\forall u_R : u_R \in V_R :: u_R \leq v_R \Leftrightarrow \mathsf{Green}(u_R)\big)$$

The second phase starts with $\mathsf{Stable}(0)$ and is ended with $\mathsf{Stable}(n_R - 1)$ being equivalent with $\mathsf{Legal}(n_R - 1)$ (lemma B.44). If $\mathsf{Stable}(v_R)$ $(v_R < n_R - 1)$ holds then execution of

occurrence $v_R + 1$ will establish $\mathsf{Stable}(v_R + 1)$. Furthermore all occurrences u_R (including the root) before v_R ($u_R \leq v_R$) are not enabled. Execution of an occurrence u_R after $v_R + 1$ ($u_R > v_R + 1$) preserves $\mathsf{Stable}(v_R)$ (lemma B.46). We conclude that if the second phase is indeed started with $\mathsf{Stable}(0)$ it will run through all v_R to be Stable at v_R, because the function Chaos does not allow infinitely many executions of non-root occurrences. Eventually $\mathsf{Stable}(n_R - 1)$ holds, ending the second phase.

The first phase starts in an arbitrary configuration and ends with $\mathsf{Stable}(0)$, the start of the second phase. As long as the second phase can't start, the function $\mathsf{Phase1}$, as defined below, will decrease.

$$\begin{aligned}
\mathsf{Missing}(s) &\triangleq \Big(\forall v : v \in V \setminus \{f(0)\} :: t[v] \neq s\Big) \wedge (t[f(0)] = s \Rightarrow d[f(0)] = 0) \\
&\quad \text{for } s \in \{0, \ldots, n\} \\
\mathsf{Miss} &\triangleq \Big(\min\, s : s \in \{0, \ldots, n\} \wedge \mathsf{Missing}(s) :: (s - t[f(0)]) \bmod (n+1)\Big) \\
\mathsf{Phase1} &\triangleq M \cdot n \cdot \mathsf{Miss} + \mathsf{Chaos}
\end{aligned}$$

Since there only n nodes there can be at most n tickets so there is at least one $\mathsf{Missing}$ and therefore Miss is well defined in any configuration (lemma B.37). If $\mathsf{Miss} = 0$ also $\mathsf{Stable}(0)$ holds (lemma B.43) (no occurrence is Green), so during the first phase $\mathsf{Miss} > 0$. Let's assume $\mathsf{Miss} > 0$. If the root executes, Miss decreases with 1 (lemma B.40). Though not trivial to prove, the increase of Chaos is at most $M \cdot n - 1$ (lemma B.33) and therefore $\mathsf{Phase1}$ decreases with at least 1 (lemma B.42).

The other occurrences only copy or preserve tickets so missing tickets stay missing, unless they copy from $f(0)$. If $\mathsf{Miss} > 0$ the ticket of the root wasn't missing anyway. Therefore they don't increase Miss (lemma B.39) and as already shown they decrease Chaos. Since the range of $\mathsf{Phase1}$ is finite, Miss will eventually equal 0, starting the second phase.

The stabilization time can in general be bounded by $O(M \cdot n^2) \subseteq O(n^3)$. In case of a ring and the obvious walk through it: $M \in O(1)$ and the protocol is almost equivalent with Dijkstra's protocol. He uses n tickets where we require $n+1$ tickets. It is possible to compute the walk in a self-stabilizing way, by first computing the total topology in every node and then have each node compute a ring in such a way that they will all compute the same ring. It would be interesting whether it could also be done more space efficient. Unlike the other protocols in this paper the read/write synchronization is limited to two processors, a processor only reads the state of *one* other processor at the time followed by a write of its own state.

5. Two Protocols Based On Spanning Trees

Although the protocol of the previous section is very space-efficient, it has one major drawback in that it requires preprocessing to compute the covering of the graph. In this section we lift this restriction at the expense of space-efficiency, and present two $O(n^2Dm)$ state per processor SSME protocols for arbitrary, strongly connected, directed graphs with n nodes, diameter D and maximal degree m. In the following two sections we will describe two SSME protocols, assuming that we are given a stable spanning tree for the graph. Both protocols require roughly $O(n^2)$ states per processor. Several self-stabilizing spanning-tree protocols

have been published, for instance by Afek et. al. [AKY90] for undirected graphs, and Dolev et. al. [DIM93] for arbitrary communication graphs. Combining this second protocol with one of the SSME protocols described below using fair protocol combination [DIM93] will yield a SSME protocol for arbitrary strongly connected directed graphs. This composition increases the state complexity by a factor of $O(mD)$, hence the complete protocol requires $O(n^2Dm)$ states per processor. For details on both the spanning-tree protocol and fair protocol combination we refer to section 5.3 and Dolev et. al. [DIM93]. Throughout this section we assume that each node has access to its identity $u \in \{0, \ldots, n-1\}$ which is stored in some storage resilient to transient errors.

5.1 The first protocol

The central idea in this first SSME protocol is to use the root of the spanning tree as coordinator. This root will continually generate numbers in the range $\{0, \ldots, n-1\}$ in cyclic order. Such a number indicates the next node to become privileged. All non-root nodes cooperate to pass this number down the spanning tree. Whenever a non-root node discovers that its identity is equal to the number held by its parent, it becomes privileged until it stores this number in its own state.

To ensure that only one node is privileged at a time, the root is only allowed to generate a new number if it can be sure that all nodes hold the same number. Note that in that case the node with identity equal to this number has already used its privilege. Inspection of the numbers held by all incoming nodes of the root alone will not guarantee this, because the root may not be able to read all leaves of the spanning tree directly. To allow the root to make the correct decision, all nodes are required to express their trust in the number they hold. For a node holding a certain number, its trust roughly corresponds to the length of the shortest path to this node starting from an arbitrary node not holding this number. Then, once all the incoming nodes of the root express the maximal trust $(D-1)$, the root may be allowed to generate the next number. The root always expresses the maximal trust for any number it holds. Then once the root sees this number on all its incoming nodes, all with maximal trust, it is certain that all nodes hold the same number.

5.1.1 The implementation In the protocol we assume that one node r is the root of the spanning tree. The identity of this node is determined beforehand, for example picking the node with identity 0. All non-root nodes u have access to their parent $\mathsf{P}(u)$ in the tree. For the purpose of this section it is assumed that this information is also stored in some non-volatile storage, but it will actually be maintained by a separate self-stabilizing spanning tree protocol (see above).

The mutual exclusion protocol appears in figure 2. In this protocol the externally visible state—i.e. the one readable by neighbouring processors—of each processor u is divided into two fields: $num[u]$ with values in $\{0, \ldots, n-1\}$, and $trust[u]$ with values in $\{0, \ldots, D-1\}$. A node is privileged iff the corresponding predicate is true in the current configuration. A privileged node can choose to access its critical resource before taking the next step in the SSME protocol, in which case it loses its privilege again.

Node r : *Privileged* if $(\forall v \in \mathsf{In}(r) : num[v] = num[r] = r \wedge trust[v] = D - 1)$
  if $(\forall v \in \mathsf{In}(r) : num[v] = num[r] \wedge trust[v] = D - 1)$
    then $num[r] := (num[r] + 1) \bmod n$
  $trust[r] := D - 1$

Node $u \neq r$: *Privileged* if $num[\mathsf{P}(u)] = u \neq num[u]$
  $num[u] := num[\mathsf{P}(u)]$
  if $(\forall v, w \in \mathsf{In}(u) : num[v] = num[w])$
    then $trust[u] := \min(\{D - 1\} \cup \{trust[v] + 1 \mid v \in \mathsf{In}(u)\})$
    else $trust[v] := 0$

Figure 2: The First Mutual Exclusion Protocols

5.1.2 Proof of correctness To prove correctness we proceed as follows. First we give a precise characterization of the legitimate configurations. We show that in these configurations the mutual exclusion property is satisfied, and that the privilege is passed fairly among all processors in the system. Then we show that once the system has reached a legitimate configuration, it will remain in a legitimate configuration as long as no further errors occur. We prove that the system is indeed self-stabilizing, by showing that the system will reach a legitimate configuration when started from an arbitrary initial configuration.

The legitimate configurations C must satisfy the following three conditions for each node u. First, the number held by u must equal the one held by r, or one less. If u holds the same number as r, then so must its parent and therefore all other nodes on the path from r to u in the spanning tree.

$$\begin{aligned}\mathsf{Proper}(C,u) \quad \equiv \quad & C.num[u] \in \{C.num[r], (C.num[r] - 1) \bmod n\} \\ & \wedge \Big[(u \neq r \wedge C.num[u] = C.num[r]) \Rightarrow C.num[\mathsf{P}(u)] = C.num[r]\Big]\end{aligned}$$

Second, if u and all its incoming nodes hold the same number as the root r, then the trust expressed by u should not be exaggerated.

$$\begin{aligned}\mathsf{Modest}(C,u) \quad \equiv \quad & \Big[(\forall v \in \mathsf{In}(u) : C.num[v] = C.num[r]) \wedge u \neq r \wedge C.num[u] = C.num[r]\Big] \\ & \Rightarrow C.trust[u] \leq \min_{v \in \mathsf{In}(u)} C.trust[v] + 1\end{aligned}$$

Third, if u holds the same number as the root r while some of its incoming nodes do not, the trust expressed by u should be 0.

$$\begin{aligned}\mathsf{Honest}(C,u) \quad \equiv \quad & \Big[(\exists v \in \mathsf{In}(u) : C.num[v] \neq C.num[r]) \wedge u \neq r \wedge C.num[u] = C.num[r]\Big] \\ & \Rightarrow C.trust[u] = 0\end{aligned}$$

Observe that according to these definitions $\mathsf{Proper}(C,r)$, $\mathsf{Modest}(C,r)$ and $\mathsf{Honest}(C,r)$ hold for the root r in any configuration C. Combining these three predicates we define the legitimate configurations C as exactly those that satisfy the predicate

$$\mathsf{Legitimate}(C) \equiv (\forall u \in V : \mathsf{Proper}(C,u) \wedge \mathsf{Modest}(C,u) \wedge \mathsf{Honest}(C,u))$$

We start with an important proposition, stating that if the root advances in a legitimate configuration, all nodes must hold the same value just prior to that.

Proposition 1 *If* $(\forall v \in \mathsf{In}(r) : C.num[v] = C.num[r] \wedge C.trust[v] = D-1)$ *for a legitimate configuration* C*, then* $(\forall v \in V : C.num[v] = C.num[r])$.

Proof: By contradiction. Suppose there is a node $u \in V$ with $C.num[u] \neq C.num[r]$. Clearly $u \notin \mathsf{In}(r)$. Consider the shortest path $(u, v_1, \cdots, v_k, r)$ from u to r for some k, $1 \leq k \leq D-1$. This path exists because the graph is strongly connected. Let $v_0 = u$, and take the largest $i \geq 0$ such that $C.num[v_i] \neq C.num[r]$. Then by $\mathsf{Honest}(C, v_{i+1})$ we have $C.trust[v_{i+1}] = 0$, and for all $j > i+1$ we have $C.trust[v_{j+1}] \leq C.trust[v_j] + 1$ by $\mathsf{Modest}(C, v_{j+1})$. Then $C.trust[v_k] < k \leq D-1$. By the definition of the path, $v_k \in \mathsf{In}(r)$, and so for some $v \in \mathsf{In}(r)$ we have $C.trust[v] \neq D-1$. This is a contradiction, and the proposition follows. □

That our protocol is indeed a mutual exclusion protocol in legitimate configurations follows from the next lemma.

Lemma 2 (safety) *In any legitimate configuration* C*, at most one processor is privileged.*

Proof: Take any node $u \neq r$. If u is privileged in C, then $C.num[u] \neq C.num[\mathsf{P}(u)] = u$. Since C is legitimate we have $\mathsf{Proper}(C,u)$ and so $C.num[u] = C.num[r]$ or $C.num[u] = (C.num[r]-1) \bmod n$. But in the second case we must have $C.num[\mathsf{P}(u)] = C.num[r]$, again by $\mathsf{Proper}(C,u)$. We conclude $C.num[u] \neq C.num[r]$ and thus $u = C.num[r]$ for a privileged node in a legitimate configuration C.

This implies that no other node $v \neq r$ can be privileged in C. Also, if the root r were privileged, we have $(\forall v \in \mathsf{In}(r) : C.num[v] = C.num[r] \wedge C.trust[v] = D-1)$. Then by proposition 1 we have $(\forall v \in V : C.num[v] = C.num[r])$. But this contradicts the fact that $C.num[u] \neq C.num[r]$ from which we conclude that the root too cannot be privileged if $u \neq r$ is privileged. □

The next lemma implies that the protocol is deadlock-free and all its executions are infinite.

Lemma 3 (no-deadlock) *In any configuration* C *at least one processor can take a non-void step.*

Proof: Let node u hold the minimal value in its *trust* field in configuration C. Then for all $v \in \mathsf{In}(u)$ $trust[v] \geq trust[u]$. If $trust[u] < D-1$, then if u, whether it is equal to r or not, takes a step $C \rightarrow C'$, then $C.trust[u] \neq C'.trust[u]$ and u would be able take a non-void step. If $trust[u] = D-1$ then, by minimality, for all nodes v $trust[v] = D-1$. If for some node u, $num[u] \neq num[\mathsf{P}(u)]$ then u is able to take a non-void step. If for all nodes $u \neq r$, $num[u] = num[\mathsf{P}(u)]$, we see that for all nodes u, $num[u] = num[r]$. Together with $trust[u] = D-1$ we see that now the root r must be able to take a non-void step. □

Because we consider only fair executions, all processors take infinitely many steps in such an infinite execution. That the protocol is also fair with respect to passing the privilege among the processors is established by the next lemma.

Lemma 4 (fairness) *If in a legitimate configuration C node u is privileged, then in any execution, the next different privileged node will be* $(u+1) \bmod n$.

Proof: In the proof of the safety-property we saw that if in a legitimate configuration C node u is privileged, then $u = C.num[r]$. Consider the execution starting at C. If the root advances for the first time after C, say in configuration C_1, then for all configurations C' with $C \Rightarrow C' \Rightarrow C_1$ we know that only u can be privileged. Similarly, if the root advances once more, say in configuration C_2, then for all configurations C'' with $C_1 \stackrel{r}{\rightarrow} C_1' \Rightarrow C'' \Rightarrow C_2$ only node $(u+1) \bmod n$ can be privileged in C''. Then it remains to be shown that node $(u+1) \bmod n$ eventually becomes privileged for at least one such C''. This is guaranteed by the fact that according to proposition 1 in configuration C_1 we have $(\forall v \in V : C_1.num[v] = C_1.num[r])$ and in configuration C_2 we have $(\forall v \in V : C_2.num[v] = C_2.num[r])$. As $C_1.num[r] = u$ and $C_2.num[r] = (u+1) \bmod n$ we must have an intermediate configuration in which node $(u+1) \bmod n$ changes its value from u to $(u+1) \bmod n$. If $(u+1) \bmod n \neq r$ then in this configuration $num[\mathsf{P}((u+1) \bmod n)] = (u+1) \bmod n$ while $num[(u+1) \bmod n] = u$, implying that node $(u+1) \bmod n$ is privileged as required. If $(u+1) \bmod n = r$, then in C_2 r is privileged. □

Note that we can weaken the privilege-condition of r to

$$(\forall v \in \mathsf{In}(r) : num[v] = num[r] \wedge trust[v] = D-1)$$

without violating the safety-property. In this case the root will be privileged every time it generates a new number, i.e. between the time that u was privileged and the time that $(u+1) \bmod n$ was privileged, the root will have been privileged as well (unless $(u+1) \bmod n$ happens to equal r).

The next lemma states that once the system reaches a legitimate configuration, it will remain in a legitimate configuration for as long as no transient errors occur.

Lemma 5 (closure) *For any legitimate configuration C, if $C \rightarrow C'$ in any execution of the protocol, then C' is legitimate.*

Proof: We split the proof in two cases

Node r takes a step: If $C.num[r] = C'.num[r]$ clearly $\mathsf{Proper}(C', u)$ and $\mathsf{Honest}(C', u)$ still hold for all $u \in V$, and $\mathsf{Modest}(C', v)$ still holds for all v with $r \notin \mathsf{In}(v)$. For those w with $r \in \mathsf{In}(w)$, notice that r must have changed its *trust* to $n-1$, and so still $C'.trust[w] \leq C'.trust[r]$ which shows $\mathsf{Modest}(C', w)$ for all those w as well.

If $C.num[r] \neq C'.num[r]$, then $C'.num[r] = (C.num[r] + 1) \bmod n$ and also $(\forall v \in \mathsf{In}(r) : num[v] = num[r] \wedge trust[v] = D-1)$. Using proposition 1 we obtain

$(\forall v \in V : C.num[v] = C.num[r])$. Then $(\forall v \neq r : C'.num[v] = (C'.num[r] - 1) \bmod n)$ and thus $\mathsf{Proper}(C', u)$ for all $u \in V$ and trivially $\mathsf{Honest}(C', u)$ and $\mathsf{Modest}(C', u)$ for all $u \in V$.

Node $v \neq r$ takes a step: It is easily checked that $\mathsf{Proper}(C', u)$ for all $u \in V$ and $\mathsf{Modest}(C', u)$ and $\mathsf{Honest}(C', u)$ for all u with $v \notin \mathsf{In}(u)$ or $C.num[u] \neq C.num[r]$. So it remains to show that also $\mathsf{Modest}(C', u)$ and $\mathsf{Honest}(C', u)$ for all u with $v \in \mathsf{In}(u)$ and $C.num[u] = C.num[r]$.

If $(\exists w \in \mathsf{In}(u) : C.num[w] \neq C.num[r])$, then by $\mathsf{Honest}(C, u)$ we have $C.trust[u] = 0$. Then also $C'.trust[u] = 0$ and trivially $\mathsf{Honest}(C', u)$ and $\mathsf{Modest}(C', u)$.

If $(\forall w \in \mathsf{In}(u) : C.num[w] = C.num[r])$, then by $\mathsf{Modest}(C, u)$ and the fact that $v \in \mathsf{In}(u)$ we have $C.trust[u] = C'.trust[u] \leq C.trust[v]$. Moreover, also $C.num[v] = C.num[r]$ and thus $C.num[\mathsf{P}(v)] = C.num[r]$ by $\mathsf{Proper}(C, v)$, so $C'.num[v] = C'.num[r]$ by the protocol. This, in turn, implies $(\forall w \in \mathsf{In}(u) : C'.num[w] = C'.num[r])$ and thus $\mathsf{Honest}(C', u)$. By $\mathsf{Modest}(C, v)$ and the protocol we see $C'.trust[v] \geq C.trust[v]$ and thus $\mathsf{Modest}(C', u)$.

□

Lemma 6 (self-stabilization) *For any execution of the protocol starting in an arbitrary configuration C, the execution will eventually reach a legitimate configuration C'.*

Proof: Start the protocol in configuration C, and wait until the root takes a step for the first time in configuration C_1: i.e. $C \Rightarrow C_1 \xrightarrow{r} C_2$. In C_1 all $v \in \mathsf{In}(r)$ have $num[v] = num[r]$. Because there are n nodes and at least one such v, at least one value in $\{0, \ldots, n-1\}$ cannot occur on any node in C_1. Take the first such value following $num[r]$ in the cyclic ordering, and call this value a. Continue the protocol until r sets $num[r] = a$ reaching configuration C_3. Because non-root nodes only copy existing values, in C_3 the root is the only node with $num[v] = a$.

Now continue the execution until the root advances once more in configuration C_4, i.e. $C_4 \xrightarrow{r} C'$, setting $num[r] = (a + 1) \bmod n$. We will show that $\mathsf{Legitimate}(C')$ holds. First observe that $(\forall v \in \mathsf{In}(r) : C_4.num[v] = a \wedge C_4.trust[v] = D - 1)$. Also observe that, since in C_3 no non-root node has $num[v] = a$, all nodes with $num[v] = a$ must have taken a step somewhere between C_3 and C_4. Now let $C_4.num[u] \neq a$ for some node u. Then for any configuration $C_3 \Rightarrow C \Rightarrow C_4$ we must have $C.num[u] \neq a$. Again, like the proof of proposition 1, consider the shortest path $u, v_1, \ldots, v_k, r$, with $k \leq D - 1$ and all intermediate nodes holding number a. Because $num[v_1] = a$, it must have taken a step, setting $trust[v_1] = 0$. Then $trust[v_2] \leq 1$ because v_2 also took a step, and, inductively, $trust[v_k] < k \leq D - 1$. But then for some $u \in \mathsf{In}(r)$ we have $C_4.trust[u] < D - 1$, contrary to assumption.

We conclude that $(\forall v \in V : C_4.num[v] = a)$ and thus in C' we have $num[r] = (a+1) \bmod n$ and $(\forall v \neq r \in V : C'.num[v] = a)$, from which $\mathsf{Legitimate}(C')$ easily follows. □

5.2 The second protocol

In this section we present a second protocol, also based on a spanning tree. The main difference is that in legitimate configurations, it does not necessarily require the cooperation

of all nodes in order to transfer the privilege to another node. In other words, it has a less strict synchronization. The downside of the comparison is that it is slightly less space-efficient ($2n^2$ states per node).

Like the protocol in the previous section, this protocol works by distributing a number through a spanning tree. The number indicates the identity of the node that should get the privilege. When that node has completed its critical section, it makes this fact known by setting a (boolean) flag. This flag can be seen as an acknowledge. When the root node finds that the flag has been set, it chooses the next number.

The number is distributed by letting each non-root node copy the number from its parent in the tree. When a node "sees" that its parent carries a number that destines the privilege to itself, it first executes the critical section. On exiting the critical section, it copies the number and sets its flag.

Unlike the number, the value of the flag is not spread via the edges of the tree. Instead, each node looks at the flags of *all* of its in-neighbours nodes. If there is a neighbour that has the same number, it takes the logical *or* of this flag and its own value. By this mechanism, a flag value of T is diffused through the graph, and eventually reaches the root. When the root finds that one of its in-neighbours has set its flag (and has the same number), it chooses a new number and sets its flag to F.

Thus, it is not necessary for all nodes to take steps before the root can choose a new number. All that is needed is that the number be copied from the root down the path in the tree to the privileged node. When that node has finished the critical section, the flag value of F must be copied along *some* path from that node to the root. A node that is not in one of those two paths does not need to update its state.

The root chooses its new number in a round-robin fashion. In contrast with the previous protocol, the numbers do not range from 0 to $n-1$ but from 0 to n^2-1. The necessity for this will become clear in the proof. A node i is privileged only if the number of its parent equals $i (\mathrm{mod} n)$.

5.2.1 The implementation We first introduce some notational definitions: The state of a node consists of a number $num \in \{0, 1, 2, \ldots, n^2 - 1\}$, and a boolean $flag \in \{T, F\}$. For a node v, $\mathsf{P}(v)$ denotes its parent, and $\mathsf{Anc}(v)$ the set of ancestors of v, i.e. those nodes $u \neq v$ for which there is a directed path within the tree from u to v. The function Dest indicates for a certain node the destined privileged node: $\mathsf{Dest}(v) = num[v] \bmod n$. The protocol is presented in figure 3.

5.2.2 Proof of correctness To prove the correctness of the protocol, we first define the set of legitimate configurations. We then show that the legitimate configurations satisfy the properties of mutual exclusion, and that progress and fairness are guaranteed. Next we show that the set of legitimate configurations is closed under the steps taken by the nodes, and finally we prove self-stabilization.

Node r : *Privileged* if $\mathsf{Dest}(r) = r \vee (\exists v \in \mathsf{In}(r) : num[v] = num[r] \wedge flag[v])$
  if $\mathsf{Dest}(r) = r \vee (\exists v \in \mathsf{In}(r) : num[v] = num[r] \wedge flag[v])$
    then $num[r] := num[r] + 1 \bmod n^2$
  $flag[r] := F$

Node $v \neq r$: *Privileged* if $\mathsf{Dest}(\mathsf{P}(v)) = v \wedge num[\mathsf{P}(v)] \neq num[v]$
  if $\mathsf{Dest}(\mathsf{P}(v)) = v$
    then $flag[v] := T$
    else $flag[v] := (\exists u \in \mathsf{In}(v) \cup \{v\} : num[u] = num[\mathsf{P}(v)] \wedge flag[u])$
  $num[v] := num[\mathsf{P}(v)]$

Figure 3: The Second Mutual Exclusion Protocol

Definition 7 *A configuration C is* legitimate *iff it satisfies*

$$\begin{aligned}
\mathsf{L1} &\equiv (\forall v \in V, u \in \mathsf{Anc}(v) : num[v] \preceq num[u]) \\
\mathsf{L2} &\equiv (\forall v \in V : (\forall x : num[v] \prec x \prec num[r] \;\Rightarrow\; x \neq v \bmod n)) \\
\mathsf{L3} &\equiv num[\mathsf{Dest}(r)] \neq num[r] \;\Rightarrow\; ((\forall v \in V : num[v] = num[r] \;\Rightarrow\; flag[v] = F)) \\
\mathsf{L4} &\equiv num[\mathsf{Dest}(r)] = num[r] \wedge \mathsf{Dest}(r) \neq r \;\Rightarrow\; flag[\mathsf{Dest}(r)] = T
\end{aligned}$$

In L1, the ordering $\preceq$ on numbers is defined as

$$x \preceq y \equiv (y - x \bmod n^2) \leq n$$

and $x \prec y$ in L2 denotes $(x \preceq y) \wedge (x \neq y)$, as might be expected.

Definition 7 can be intuitively understood as follows: since the numbers are distributed downward through the tree, the ancestors of a node hold "more recent" numbers. The root increases the number, so nodes on higher levels should hold "higher" numbers. Since the root needs the cooperation of the destined privileged node in order to increase the number, the number of a node can never "lag behind" more than n (L1). Also, it is prohibited that the root increases the number before the privileged node has seen this number. Suppose that node v has a number that is different from the one that the root holds. The ancestors of v hold numbers in the range from $num[v]$ to $num[r]$. The numbers that lie strictly "in between" must not destine the privilege to v: if there is an ancestor u of v for which $\mathsf{Dest}(u) = v$ yet $num[u] \neq num[v]$, then it must be that $num[r] = num[u]$, otherwise u might overwrite this number (by copying the number from its parent) before v sees it. L2 excludes these configurations. Lastly, the flags must express an "acknowledge" by node $\mathsf{Dest}(r)$. If this node has set its number to $num[r]$ (i.e. completed the critical section), then its flag must be T (L4). If it has not yet done so, then the flags of all nodes that do hold $num[r]$ should be F (L3).

In the following lemmas, L denotes an arbitrary legitimate configuration, C is an arbitrary configuration (not necessarily legitimate), and d is the node that is destined to get the privilege according to the root: $d = \mathsf{Dest}(r)$.

Lemma 8 *If, in L, the root r is privileged, then*

1. $num[d] = num[r]$
2. *if* $d \neq r$ *then* $flag[d] = T$

Proof: If r is privileged, then by definition of the protocol

$$((\exists i \in V : (num[i] = num[r]) \wedge (flag[i] = T))) \vee (d = r)$$

Combined with L3, the first part of the lemma follows. Applying L4 then yields the second part. □

Lemma 9 *If, in L, a non-root node v with parent $p = \mathsf{P}(v)$ is privileged, then*

1. $num[p] = num[r]$
2. $v = d$

Proof: If v is privileged, then by definition of the protocol

$$\mathsf{Dest}(p) = v \wedge num[v] \neq num[p]$$

Since by L1 $num[v] \preceq num[p] \preceq num[r]$ we can use L2 to obtain $num[p] = num[r]$. The second part of the lemma follows directly from this. □

With lemmas 8 and 9 it is also easy to prove that in a legitimate configuration a node can not lose its privilege while it is executing the critical section.

Theorem 10 (safety) *In any legitimate configuration, at most one node is privileged.*

Proof: If some non-root node v is privileged, then by lemma 9 $num[v] \neq num[\mathsf{P}(v)] = num[r]$. According to the protocol $v = \mathsf{Dest}(r)$, hence the root node is not privileged (as that would contradict the first part of lemma 8). Furthermore, if any other node u is privileged, then $u = v$, by the second part of lemma 9. □

Theorem 11 (progress) *In an arbitrary execution, let the system be in configuration C. Eventually, the system will reach a configuration C' in which the root is privileged.*

Proof: We may assume that $\mathsf{Dest}(r) \neq r$, and that r does not take a step (otherwise r becomes privileged and we are ready).

Consider the set $A = \{v |\ num[v] = num[r] \wedge flag[v] = T\}$. If A is empty in C, then the number held by r will be distributed through the tree, node $d = \mathsf{Dest}(r)$ will become privileged and set its flag to T. So eventually, the system reaches a configuration where A is non-empty. In the configurations reachable from this configuration, A will become larger, until it includes an incoming node of r. At this point, r becomes privileged. □

Theorem 12 (fairness) *In an arbitrary execution, let the system be in a legitimate configuration L. For any node v, the system will eventually reach a configuration L' in which v is privileged.*

Proof: By repeatedly applying theorem 11, we know that the system will reach a legitimate configuration L'', in which $\mathsf{Dest}(r) = v$. Since the root was privileged in the configuration just prior to L'', there is no node i for which $num[i] = num[r]$. Hence $num[v] \neq num[r]$, the number will be distributed through the tree and eventually, v will be privileged. □

Theorem 13 (closure) *If the system goes from a configuration C to configuration C', and C is legitimate, then C' is legitimate as well.*

Proof: The transition from C to C' must be one of the following:

1. The root takes a step (increases its number)
2. The privileged node $d \neq r$ takes a step (copies its number from its parent and sets its flag to T)
3. A non-root node $v \neq d$ takes a step (copies the number from its parent, and copies the flag from an incoming node that has the same number)

We show that in all cases the requirements for a legitimate configuration remain satisfied in C'.

1. Suppose that L1 is violated in C' for a certain node i (i.e. $num[r] - num[i] > n \bmod n^2$). Since L1 held in C and $num[r]$ was increased by 1 in the step to C', in C $num[r] = num[i] + n \pmod{n^2}$. But since L2 also holds in C, all numbers x strictly between num[i] and num[r] have $\mathsf{Dest}(x) \neq i$. Since there are $n-1$ of those numbers it must be that $\mathsf{Dest}(r) = \mathsf{Dest}(i) = i$. Applying lemma 8 results in $num[i] = num[r]$, which is a contradiction.

   Suppose L2 were violated in C' for a node i. Since L2 held in C and r increased its num by 1, it must be that in C num[i]=num[r]. Hence in C' there is no x for which $num[i] \prec x \prec num[r]$, contradicting the assumption.

   L3 and L4 must hold in C', because there is no $v \neq r$ for which $num[v] = num[r]$ (since by L1 r held the "largest" number in C).

2. In C', $num[d] = num[r]$ (lemma 9), so d satisfies L2. It is easily verified that L1, L3 and L4 are also satisfied in C'.

3. As in the previous case, L1 and L4 trivially hold in C'. L2 and L3 must also be satisfied, otherwise they would already have been violated in C, by the node that v copied the state from.

□

We must now prove self-stabilization. According to theorem 11, the root will keep on increasing its number. We can view an execution as consisting of *rounds*, separated by a step

in which the root chooses a new number. In a round, the non-root nodes may take steps, but the state of the root does not change.

The proof that the system stabilizes from any initial configuration is based on the presence of what may be called *unoccupied* numbers. Define $\mathsf{Occ} = \{x |\ (\exists v \in V, v \neq r : num[v] = x)\}$, the set of *occupied* numbers, i.e. numbers that are present in some non-root node. Suppose that the system starts a new round, and that the new value of $num[r]$ is not occupied. The root can not increase its num again until node $\mathsf{Dest}(r)$ has been privileged. The number held by the root is distributed through the tree until it reaches this node. After the critical section has been completed, the flag is set and the 'acknowledge' is diffused through the graph. At the end of this round, node $\mathsf{Dest}(r)$ has a num equal to $num[r]$.

The proof is split in two parts. First we prove that from any configuration, the system will reach a configuration from which in the next n rounds, the root will choose an unoccupied value. We then show that the configuration reached after these n rounds is legitimate.

Theorem 14 (self-stabilization) *In an arbitrary execution, let the system be in configuration C.*

1. *Eventually, the system will reach a configuration C' in which*

$$\neg\,(\exists v \in V : num[r] \prec num[v])$$

2. *From C', the system will eventually reach a legitimate configuration*

Proof:

1. First, observe that in any configuration, there exist at least n consecutive unoccupied numbers (since the set of numbers has size n^2, and the non-root nodes hold at most $n-1$ different numbers). In C, let $u_1, u_2, \ldots, u_n$ be such a row of consecutive numbers. Second, consider the way in which Occ changes during an execution. The only steps in which new numbers are added to Occ is when the root chooses a new number. Thus, in any execution from C, the numbers remain unoccupied until the round in which $num[r]$ is equal to u_1. Call the configuration just prior to the start of this round C'. In this configuration $num[r] = u_1 - 1 \bmod n^2$, so the only numbers x for which $num[r] \prec x$ are precisely the ones $u_1, u_2, \ldots, u_n$.

2. In configuration C', the root changes its number to u_1. Since u_1 is unoccupied, eventually node $v = u_1 \bmod n^2$ will become privileged. At the end of this round, $num[r]$ is set to u_2, which is still an unoccupied number. We can repeat this argument for $u_2, u_3, \ldots, u_n$. Call the configuration that is reached after these n rounds C''. Between C' and C'', each non-root node has been privileged exactly once, and exactly one node was privileged in each round.

   We can now show that C'' is a legitimate configuration:

- Let v be an arbitrary non-root node, with parent p. Immediately after v was privileged, $num[v] = num[p]$. In all steps thereafter, if $num[v]$ or $num[p]$ changed, then either $num[v]$ was set equal to $num[p]$, or $num[p]$ was set to the number of p's parent, which is a higher number (which can be proved by induction on p's depth in the tree). Hence $num[v] \preceq num[p]$ in C''. Also, in C'' all numbers held by a node are within the range $u_1, u_2, \ldots, u_n$. Together this yields L1.
- Immediately after some node v was privileged, $num[v] = num[r]$ and $\mathsf{Dest}(r) = v$. From then on until C'', $num[r]$ is incremented at most n times. Thus, L2 must hold in C''.
- In C'', there is no non-root node that has the same number as the root, hence L3 and L4 trivially hold.

□

5.3 About spanning trees & combining protocols

Several self-stabilizing spanning-tree protocols have been published, for instance by Afek et. al. [AKY90] for undirected graphs, and Dolev et. al. [DIM93] for arbitrary communication graphs. Since this last result matches our needs, we briefly sketch the (adapted) spanning tree protocol of Dolev et. al.

Node r :
$dist[r] \leftarrow 0$
$\mathsf{P}(r) \leftarrow nil$

Node $u \neq r$:
$dist[u] \leftarrow 1 + \min\{dist[v] \mid v \in \mathsf{In}(u)\}$
$\mathsf{P}(u) \leftarrow$ deterministically select $v \in \mathsf{In}(u)$ with $dist[v] = \min\{dist[v] \mid v \in \mathsf{In}(u)\}$

Figure 4: Spanning Tree Protocols

The self-stabilizing spanning tree protocol is presented in figure 4. It will build a breadth-first-search tree using the distance to the root of the spanning tree. The range of values that can be assumed by field $dist[u]$ should equal $\{0, \ldots, D\}$, where D is an upperbound on the diameter of the graph. To break the symmetry, one of the nodes is selected root a-priori and forces its distance to 0. The other nodes read the distance of their neighbours, and add 1 to the minimal value found. A selection among the nodes with minimal distance must be made to determine the parent in the spanning tree. This selection must be deterministic to ensure that once a complete spanning tree is computed, it remains stable provided no further errors occur.

A subtle point that should not go unmentioned is related to storing the value for $\mathsf{P}(u)$. Technically speaking the identity of any of the incoming nodes cannot be determined, because the identity of a node is not part of the state readable by other processors. But in fact, we are not even interested in the *identity* of such a node: we only need to know which of the incoming nodes is the desired node. So in fact we only have to store a 'port-address', or

something similar, in $\mathsf{P}(u)$. If m equals the maximal in-degree of any node in the graph, then storing $\mathsf{P}(u)$ requires $O(m)$ values. The proof of the protocol is straightforward, and for more details we refer to [DIM93].

Like [DIM93]—where the notion of fair protocol composition is formally derived—we compose both protocols to obtain a SSME protocol for arbitrary directed graphs. By observing that the mutual exclusion part does not alter the part of the state used by the spanning tree protocol, it is easily seen that any proof of this spanning tree protocol still holds for the combined case. After stabilization of the spanning-tree part of the combined protocol, this part will not influence the mutual exclusion part any more: the values stored in the $\mathsf{P}(u)$ will not change. Then after stabilization of the spanning-tree part, the proof of the mutual exclusion part can be used unaltered to prove stabilsation and correctness of the whole protocol.

The overall space-complexity of the protocol is determined by multiplying the number of values stored in $num[u]$, $trust[u]$, $dist[u]$, and $\mathsf{P}(u)$. This gives $n \cdot D^2 \cdot m$ states per node. A factor m can be saved by observing that $\mathsf{P}(u)$ is really a function of the state of u and is only used and maintained locally. This means that $\mathsf{P}(u)$ does not have to be stored explicitly.

6. Conclusions & Further Research

In this paper we investigated the complexity, in number of states per processor, of achieving self-stabilizing mutual exclusion on strongly connected directed graphs. Starting of with Tchuente's approach, reducing the number of states per processor needed to $O(n^2)$, we presented two protocols using $O(n^2 D)$ states per processor. It would be interesting to see whether the second protocol can be modified to work with $O(n)$ numbers, reducing its complexity to $O(nD)$. In any case, we have shown an exponential improvement in space-complexity over the best known previous protocols for mutual exclusion on directed graphs.

7. Acknowledgements

The authors wish to express their utmost gratitude to Ted Herman for the advice and guidance he has given while he was visiting Utrecht University. Without his help this paper might never have been written.

References

[Aba90] Abadi, M. An axiomatization of Lamport's temporal logic of actions. In *Concur '90. Theories of Concurrency: Unification and Extension* (1990), J. Baeten and J. Klop (Eds.), Springer-Verlag, pp. 57–69. volume 458.

[AKY90] Afek, Y., Kutten, S., and Yung, M. Memory-efficient self stabilizing protocols for general graphs. In *4th Int. Workshop on Distributed Algorithms* (1990), pp. 15–28.

[BP89] Burns, J. E., and Pachl, J. Uniform self-stabilizing rings. *ACM Transactions on Programming Languages and Systems* **11**, 2 (1989), 330–344.

[Dij74] Dijkstra, E. W. Self-stabilizing systems in spite of distributed control. *Communications of the ACM* **17**, 11 (1974), 643–644.

[Dij82] Dijkstra, E. W. Self-stabilization in spite of distributed control. In *Selected Writings on Computing: A Personal Perspective*. Springer-Verlag, New York, 1982,

pp. 41–46.

[DIM93] DOLEV, S., ISRAELI, A., AND MORAN, S. Self-stabilization of dynamic systems assuming only read/write atomicity. *Distributed Computing* **7**, 1 (1993), 3–16.

[Lam91] LAMPORT, L. The temporal logic of actions. DEC-SRC-report 79, 1991.

[Tch81] TCHUENTE, M. Sur l'autostabilisation dans un réseau d'ordinateurs. *RAIRO Informatique Théoretique* **15**, 1 (1981), 47–66.

A. A correctness proof in TLA

A.1 Introduction

In this section we will substantiate our claims about the protocol, using the Temporal Logic of Actions (TLA). A readable and extensive introduction to this proof system can be found in [Lam91]. A complete axiomatization of its proposition part can be found in [Aba90], though it should be noted that axiom 16 should be changed a little. Here I give a small introduction especially intended to understand the following proof.

In TLA we have four types of formulas: (temporal) formulas, actions, state-predicates and state-functions.

(Temporal) formulas A statement in TLA is a *temporal formula* and its validity is evaluated on a single execution (a reflexive, transitive, linear and discrete Kripke frame). If a formula is valid we also say that it *holds*. A formula is a tautology (true) if it is valid in any execution. We will write $\models \phi$ if the formula *phi* is a tautology and write $\vdash \phi$ if we can also derive it. Both specifications and properties of protocols are formulas. We use Greece symbols for formulas. The capital symbols are used to denote specifications and properties and the non-capitals are used to denote arbitrary formulas. All boolean combinations of formulas are again formulas, with their obvious semantics. If ϕ is a formula then also $\Box\phi$ and $\Diamond\phi$ are formulas. $\Box\phi$ means that ϕ is valid at any state in the execution. $\Diamond\phi$ is equivalent with $\neg\Box\neg\phi$ and means that ϕ holds at some state in the execution (possibly the current state).

Logical implication is used to express *implementation*. If Φ is a specification and Π is a property then proving that $\Phi \Rightarrow \Pi$ is a tautology shows that Φ satisfies the property Π, implements a property (possibly a higher-level specification).

Actions Actions describe *state transitions*, i.e. their semantics is a binary relation on states. Actions constitute the building blocks of TLA. Proving properties of specifications mostly involves proving properties of actions, glued together with temporal logic. We use calligraphic capitals to denote actions. We write $\models_{\mathcal{A}} \mathcal{B}$ if $\mathcal{B}$ is a tautology and write $\vdash_{\mathcal{A}} \mathcal{B}$ if we can also give a derivation. Boolean combinations of actions are again actions, with their obvious meaning. If $\mathcal{A}$ is an action then $[\mathcal{A}]$ and $\langle\mathcal{A}\rangle$ are formulas. $[\mathcal{A}]$ means that the next *state-change* (a state transition from one state to a *different* state) satisfies $\mathcal{A}$ or there is no state-change at all. $\langle\mathcal{A}\rangle$ is equivalent to $\neg[\neg\mathcal{A}]$ and means that there is a state-change and it satisfies $\mathcal{A}$.

As a consequence $[\bot]$ (where $\bot$ denotes the empty relation) means there will be no state-change at all, and we associate this with termination. Likewise $\langle\top\rangle$ (where $\top$ is the complement of the empty relation) denotes non-termination, i.e. there will be a state-change. $\Box[\mathcal{A}]$ now means that all state-changes satisfy $\mathcal{A}$. Furthermore $[\mathcal{A}]$ is equivalent with $\langle\mathcal{A}\rangle \vee [\bot]$. Since $[\top]$ is trivially a tautology, proving an action $\mathcal{A}$ equivalent with $\top$ also proves $[\mathcal{A}]$ is a tautology. However it does not prove $\langle\mathcal{A}\rangle$ a tautology, because in addition this implies non-termination.

Predicates Predicates, also calles *state-predicates*, can be evaluated in a single state. Boolean combinations of predicates are again predicates, with their obvious meaning. We use capitals to denote arbitrary predicates or the italic font if the name consist of more than one symbol. A predicate is also a formula (using an implicit cast), meaning

that the predicate holds in the first state of the execution. If predicate P is a tautology we will write $\models_{\mathcal{A}} P$ and write $\vdash_{\mathcal{A}} P$ if we can also give a derivation.

For example a typical specification looks like $Init \wedge \Box[\mathcal{A}]$, the first state satisfies the initialization predicate $Init$ and all state-changes satisfy $\mathcal{A}$. A predicate P is also (using an implicit cast) an action, meaning a state transition from a state satisfying P to some arbitrary state. Also a *primed* predicate P' is an action, meaning a state transition from an arbitrary state to one satisfying P. As a typical example of proving a safety property, we might prove $P \wedge \mathcal{A} \Rightarrow P'$ and $Init \Rightarrow P$ (as tautologies on actions) and conclude $Init \wedge \Box[\mathcal{A}] \Rightarrow \Box P$ is a tautology, i.e. the specification $Init \wedge \Box[\mathcal{A}]$ satisfies property $\Box P$ (predicate P holds at any state in an execution satisfying the specification).

If $\mathcal{A}$ is an action then $*\mathcal{A}$ and $\circ\mathcal{A}$ are predicates. $*\mathcal{A}$ holds in a state s_1 if there is a state s_2 (not necessary different from s_1) such that the state-transition (s_1, s_2) satisfies $\mathcal{A}$. $*\mathcal{A}$ is the *enabled predicate* of $\mathcal{A}$, if $*\mathcal{A}$ holds then $\mathcal{A}$ is enabled. $\circ\mathcal{A}$ is equivalent to $\neg * \neg\mathcal{A}$, but is not used very often. The most important axiom concerning the enabled predicate is $\mathcal{A} \Rightarrow *\mathcal{A}$: if $\mathcal{A}$ is executed then $\mathcal{A}$ was enabled. There is not yet any reference to the axiomatization of the enabled predicate, except for my personal notes. Furthermore it is not clear whether this axiomatization is complete.

State-functions Just as propositional logic is connected to the real world mathematics by the use of functions, relations and quantifiers, so is TLA. Variables and constants (also called rigid variables) are the basic functions, in a state they may have some value, a mathematical object like a natural number. To avoid obvious and obscuring formalities, we will show their use with an example.

Let $\mathcal{A}$ be specified by $x' = x + 1$. x is a program variable and $\mathcal{A}$ specifies that x' (the value of x after the state transition) is equal to $x+1$ (the value of $x+1$ before the state transition). A state transition therefore only satisfies $\mathcal{A}$ if x is incremented by 1. Now consider the specification

$$x = 0 \wedge \Box[x' = x + 1]$$

Obviously this specification satisfies the property $\Box(x \geq 0)$. How do we prove this?

$$\frac{\begin{array}{ll} Init \Rightarrow P & x = 0 \Rightarrow x \geq 0 \\ P \wedge \mathcal{A} \Rightarrow P' & x \geq 0 \wedge x' = x + 1 \Rightarrow (x \geq 0)' \end{array}}{\begin{array}{ll} Init \wedge \Box[\mathcal{A}] \Rightarrow \Box P & x \geq 0 \wedge \Box[x' = x + 1] \Rightarrow \Box(x \geq 0) \end{array}}$$

We will use a monad structure to depict quantifiers, set operations and aggregates.

Quantifiers $\Big(\forall j : j \in \{1, \ldots, n\} :: P(i, j)\Big)$ means for all $j = 1, \ldots, n$, $P(i, j)$ holds, note that i is a free variable in this expression. Other possible quantifiers are $\exists$ and $\exists!$.

Set operations $\Big(\bigcup i : i \in \{1, \ldots, n\} :: S(i)\Big)$ is the union of all sets $S(i)$ for $i = 1, \ldots, n$. The other set operation is $\bigcap$.

Aggregates $\Big(\sum i : i \in \{1, \ldots, n\} :: f(i)\Big)$ is the sum of all $f(i)$ for $i = 1, \ldots, n$. Other aggregates are $\prod$, max, min and # (number of).

The outer parentheses will be laid out around the whole formula inside, even if this covers more than one line. It would be more consequent if the set-notation also was changed to explicitly denote its bound variables, however I do not have a definite choice yet, so I sticked to the traditional notation.

Let $\{\mathcal{A}(i)|1 \leq i \leq n\}$ be a set of actions describing some components then a standard asynchronous composition consists of their disjunction

$$\Big(\bigvee i : i \in \{1, \ldots, n\} :: \mathcal{A}(i)\Big)$$

If component i has a set of local variables $L(i)$ then all other components have to assert not to use these variables, i.e.

$$\Big(\bigwedge i, j : i, j \in \{1, \ldots, n\} :: i \neq j \Rightarrow \mathit{Unchanged}\ (L(i))\Big)$$

where $\mathit{Unchanged}\ (L(i)) = \Big(\bigwedge v : v \in L(i) :: \mathit{Unchanged}\ (v)\Big)$ and $\mathit{Unchanged}\ (v) = (v = v')$.

The proof structure we use is basically a list $\phi_1, \ldots, \phi_n$ of formulas (actions or predicates), that are all true statements. Each statement follows from previous statements using a valid derivation rule (or are evidently true by them selfses).

To ease down the requirement of the statements to be really true, we can put all statements in a context of hypotheses. Generalisation with a context of hypotheses is a risky business. In this paper we use the word 'arbitrary' in our assumption to announce generalisation and generalise in the prove-part (following the assumptions). So proving

$$\Big(\forall i : i \in D :: P(i)\Big)$$

will look like

$\langle 1\rangle 1.\ \Big(\forall i : i \in D :: P(i)\Big)$
LET: $i \in D$ arbitrary
PROVE: $P(i)$
$\langle 2\rangle 1.\ \ldots$
$\langle 2\rangle 2.\ \ldots$
$\langle 2\rangle 3.$ Q.E.D. by $\langle 2\rangle 1$ and $\langle 2\rangle 2$

In this way the correctness of the application of generalisation can be easily checked at the beginning of the proof. A similar (and closely related) problem arises in modal logic. However there will be very few modal proofs here, so we will not digress on it.

A statement can also be derived by starting a seperate proof, possibly involving new hypotheses. We will not repeat already stated hypotheses at the start of these (nested) proofs. In this paper we don't refer to hypotheses by labels (as we do with the statements). If the hypothesis is short, like $v_R < n_R - 1$, we write it just like that. If it is too long we may write 'by assumption' or 'by let of v_R'.

In using lemmas sometimes the used substitution may become confusing. In these cases we give it explicitly, like

$$\text{by lemma } 1.1[v_R, u_R := n_R - 1, v_R]$$

Only substitution of variables bound by the *outer universal quantifier* is allowed this way. Often this is followed by references to previous proven statements to fill in assumptions of

the lemma. In these cases we stick to the order in the lemma (as much as possible). We apply the existential quantifier ($\exists$-elimination) by

$$\text{Let } i \in \{1,\ldots,n\} \text{ s.t. } P(i), \text{ such } i \text{ exists by } \texttt{reference}$$

where we refer to a statement

$$\Big(\exists i : i \in \{1,\ldots,n\} :: P(i)\Big)$$

The conclusion we get from this should (of course) not have i as a free variable.

When exhibiting a case analysis (or natural induction) we announce its use with 'by case analysis on ... and ...' and start a seperate proof for each case, possibly preceded by statements used in several cases. Some time we refer to the '*op*-mono of ...', where *op* is one of $[], \langle\rangle, \Box, \Diamond$. This refers to the monotonicity of these operations. If we can derive
(*) $\vdash_{\mathcal{A}} \Big(\forall i : i \in I :: \mathcal{A} \Rightarrow \mathcal{B}\Big)$
then we can also derive
$\vdash \Big(\forall i : i \in I :: [\mathcal{A}] \Rightarrow [\mathcal{B}]\Big)$ by []-mono on (*)
$\vdash \Big(\forall i : i \in I :: \langle\mathcal{A}\rangle \Rightarrow \langle\mathcal{B}\rangle\Big)$ by $\langle\rangle$-mono on (*)
In case of $\Box$ and $\Diamond$ we only allow the $\forall i$-quantor in case i is a rigid variable (i.e. a variable not changed by the protocol).

In formulas we may use *bulleted* conjunctions and disjunctions. The purpose of this notation is to use indentation instead of parentheses. So e.g. the following formula

$$\begin{array}{l} \wedge\ \alpha \\ \wedge\ \vee\ \beta \\ \quad\ \vee\ \gamma \\ \wedge\ \delta \end{array}$$

is the same as

$$\alpha \wedge (\beta \vee \gamma) \wedge \delta$$

B. The TLA proof

Let us start with some obvious facts about the function f and the associated notions.

(F1) $\vdash_{\mathcal{A}} \Big(\forall v_R : v_R \in V_R :: v_R \in \mathit{Sim}(f(v_R))\Big)$

(F2) $\vdash_{\mathcal{A}} \Big(\forall v_R : v_R \in V_R :: \mathit{First}(f(v_R)) \leq v_R \leq \mathit{Last}(f(v_R))\Big)$

Next we define the protocol in TLA style.

Definition B.1

$$
\begin{array}{lcl}
\mathcal{A}(v_R) & \triangleq & \wedge \Big(\forall v : v \in V \setminus \{f(v_R)\} :: \mathit{Unchanged}\ (t[v], d[v])\Big) \\
& & \wedge\ \textbf{if } v_R = 0 \\
& & \quad \textbf{then } \wedge\, t[f(n_R - 1)] = t[f(0)] \\
& & \qquad\quad \wedge\, d[f(n_R - 1)] = n_R - 1 \\
& & \qquad\quad \wedge\, t[f(0)]' = t[f(0)] + 1 \bmod Q \\
& & \qquad\quad \wedge\, d[f(0)]' = 0 \\
& & \quad \textbf{elseif } v_R = \mathit{First}(f(v_R)) \\
& & \quad \textbf{then } \wedge\, t[f(v_R - 1)] \neq t[f(v_R)] \\
& & \qquad\quad \wedge\, d[f(v_R - 1)] \geq v_R - 1 \\
& & \qquad\quad \wedge\, t[f(v_R)]' = t[f(v_R - 1)] \\
& & \qquad\quad \wedge\, d[f(v_R)]' = v_R \\
& & \quad \textbf{elseif } v_R > \mathit{First}(f(v_R)) \\
& & \quad \textbf{then } \wedge\, t[f(v_R - 1)] = t[f(v_R)] \\
& & \qquad\quad \wedge\, d[f(v_R - 1)] \geq v_R - 1 \\
& & \qquad\quad \wedge\, d[f(v_R)] < v_R \\
& & \qquad\quad \wedge\, t[f(v_R)]' = t[f(v_R - 1)] \\
& & \qquad\quad \wedge\, d[f(v_R)]' = v_R \\
& & \quad \textbf{end if} \\
\mathcal{S} & \triangleq & \Big(\exists v_R : v_R \in V_R :: \mathcal{A}(v_R)\Big) \\
\\
\Phi & \triangleq & \Box[\mathcal{S}] \wedge \mathrm{WF}(\mathcal{S})
\end{array}
$$

Let us first conclude some (obvious) facts about the protocol.

$$
\begin{array}{ll}
\text{(F3)} & \vdash_{\mathcal{A}} \Big(\forall v_R : v_R \in V_R :: \mathcal{A}(v_R) \Rightarrow d[f(v_R)]' = v_R\Big) \\
\text{(F4)} & \vdash_{\mathcal{A}} \Big(\forall v_R : v_R \in V_R \setminus \{0\} :: \mathcal{A}(v_R) \Rightarrow t[f(v_R)]' = t[f(v_R - 1)]\Big) \\
\text{(F5)} & \vdash_{\mathcal{A}} \Big(\forall v_R : v_R \in V_R \setminus \{0\} :: \mathcal{A}(v_R) \Rightarrow t[f(0)]' = t[f(0)]\Big)
\end{array}
$$

$*\mathcal{A}(v_R)$ is predicate that holds in configurations where the action $\mathcal{A}(v_R)$ can be executed, i.e. its guards hold. Since we will often use $*\mathcal{A}(v_R)$, we give it explicitly.

$$
\begin{array}{rcl}
*\mathcal{A}(v_R) \Leftrightarrow \wedge\, v_R = 0 \Rightarrow \wedge\, t[f(0)] = t[f(n_R - 1)] & & \\
\qquad\qquad\qquad\qquad\qquad\quad \wedge\, d[f(n_R - 1)] = n_R - 1 & & \\
\wedge\, v_R \neq 0 \wedge v_R = \mathit{First}(f(v_R)) \Rightarrow \wedge\, t[f(v_R)] \neq t[f(v_R - 1)] & & \\
\qquad\qquad\qquad\qquad\qquad\quad \wedge\, d[f(v_R - 1)] \geq v_R - 1 & & \\
\wedge\, v_R \neq 0 \wedge v_R \neq \mathit{First}(f(v_R)) \Rightarrow \wedge\, t[f(v_R - 1)] = t[f(v_R)] & & \\
\qquad\qquad\qquad\qquad\qquad\quad \wedge\, d[f(v_R - 1)] \geq v_R - 1 & & \\
\qquad\qquad\qquad\qquad\qquad\quad \wedge\, d[f(v_R)] < v_R & &
\end{array}
$$

Note that in the third conjunct we have $v_R \neq \mathit{First}(f(v_R))$ instead of $v_R > \mathit{First}(f(v_R))$. By (F1) these are equivalent.

B.1 Domain constraints

In selfstabilizing protocols it is very customary to assume *domain constraints* on the variables used. If a is a variable then a domain constraint for a has the form $a \in A$, where A is its domain. If Φ is the protocol then it must satisfy $\vdash a \in A \wedge \Phi \Rightarrow \Box(a \in A)$. In general a selfstabilizing protocol has no initialization predicate. A domain constraint however may be

assumed initially, because we can change the protocol s.t. the domain constraints are satisfied after a step of each proces. We change the program of each processor, such that it will run its program only when the domain constraints on its own variables is satisfied. Otherwise it will set its own variables to some arbitrary value in their domain. When each processor has taken at least one step, the domain constraints will hold.

Definition B.2 $\mathit{Domain} \triangleq \Big(\forall v : v \in V :: t[v] \in \{0, \ldots, Q-1\} \wedge d[v] \in \mathit{Sim}(v)\Big)$

Lemma B.1 $\vdash_{\mathcal{A}} \mathit{Domain} \wedge \mathcal{S} \Rightarrow \mathit{Domain}'$

Proof:

ASSUME: $\mathit{Domain} \wedge \mathcal{S}$
LET: $v_R \in V_R$ s.t. $\mathcal{A}(v_R)$ such v_R exists by $\mathcal{S}$
LET: $v \in V$ arbitrary
PROVE: $t[v]' \in \{0, \ldots, Q-1\} \wedge d[v]' \in \mathit{Sim}(v)$
$\langle 1 \rangle 1$. Q.E.D. by case analysis on $v \neq f(v_R)$ and $v = f(v_R)$
CASE: $v \neq f(v_R)$
  $\langle 2 \rangle 1$. $t[v]' = t[v]$ by $\mathcal{A}(v_R)$ and $v \neq f(v_R)$
  $\in \{0, \ldots, Q-1\}$ by *Domain*
  $\langle 2 \rangle 2$. $d[v]' = d[v]$ by $\mathcal{A}(v_R)$ and $v \neq f(v_R)$
  $\in \mathit{Sim}(v)$ by *Domain*
  $\langle 2 \rangle 3$. Q.E.D. by $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$
CASE: $v = f(v_R)$
  $\langle 2 \rangle 1$. $d[v]' = d[f(v_R)]'$ by $v = f(v_R)$
  $= v_R$ by (F3)$[v_R := v_R]$
  $\in \mathit{Sim}(f(v_R))$ by (F1)$[v_R := v_R]$
  $= \mathit{Sim}(v)$ by $v = f(v_R)$
  $\langle 2 \rangle 2$. $t[v]' \in \{0, \ldots, Q-1\}$ by case analysis on $v_R = 0$ and $v_R \neq 0$
  CASE: $v_R = 0$ so $\mathcal{A}(0)$ and $v = f(0)$
    $\langle 3 \rangle 1$. $t[v]' = t[f(0)]'$ by $v = f(0)$
    $= t[f(0)] + 1 \bmod Q$ by $\mathcal{A}(0)$
    $\in \{0, \ldots, Q-1\}$
  CASE: $v_R \neq 0$
    $\langle 3 \rangle 1$. $t[v]' = t[f(v_R)]'$ by $v = f(v_R)$
    $= t[f(v_R - 1)]$ by (F4)$[v_R := v_R]$
    $\in \{0, \ldots, Q-1\}$ by *Domain*
  $\langle 2 \rangle 3$. Q.E.D. by $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$

■

Corollary B.2 $\vdash \mathit{Domain} \wedge \Phi \Rightarrow \Box \mathit{Domain}$

Proof: Follows from lemma B.1 by applying INV1. ■

Corollary B.3
$\vdash \mathit{Domain} \wedge \Phi \Leftrightarrow \mathit{Domain} \wedge \Box[\mathcal{T}] \wedge \mathrm{WF}(\mathcal{T})$
where $\mathcal{T} \triangleq \mathcal{S} \wedge \mathit{Domain} \wedge \mathit{Domain}'$

Proof: Follows from lemma B.1 by applying our version of INV2: if $\vdash_{\mathcal{A}} I \wedge \mathcal{N} \Rightarrow I'$ then $\vdash \begin{array}{l} I \wedge \Box[\mathcal{N}] \wedge \mathrm{WF}(\mathcal{N}) \\ I \wedge \Box[\mathcal{M}] \wedge \mathrm{WF}(\mathcal{M}) \end{array}$. ■

Corollary B.4 $\vdash_{\mathcal{A}} \mathit{Domain} \Rightarrow \Big(\forall v : v \in V :: \mathit{First}(v) \leq d[v] \leq \mathit{Last}(v)\Big)$

Proof: *Domain* implies for any $v \in V$, $d[v] \in \mathit{Sim}(v)$. Therefore $d[v] = v_R$ for some v_R s.t. $f(v_R) = v$, the corollary now follows from fact (F2). ■

B.2 Privilige in a legitimate configuration
In this subsection we define the legitimate configurations and prove that only one occurrence is enabeled in such a configuration.

$$\begin{array}{rcl} \mathit{Green}(v_R) & \triangleq & t[f(v_R)] = t[0] \wedge d[f(v_R)] \geq v_R \\ \mathit{Red}(v_R) & \triangleq & \wedge\ d[f(v_R)] \geq v_R \Rightarrow t[f(v_R)] = t[f(0)] - 1 \bmod Q \\ & & \wedge\ d[f(v_R)] < v_R \Rightarrow t[f(v_R)] = t[f(0)] \\ \mathit{Legal}(v_R) & \triangleq & \left(\begin{array}{r} \forall u_R : u_R \in V_R :: \wedge\ u_R \leq v_R \Rightarrow \mathit{Green}(u_R) \\ \wedge\ u_R > v_R \Rightarrow \mathit{Red}(u_R) \end{array}\right) \\ \mathit{Legitimate} & \triangleq & \Big(\exists v_R : v_R \in V_R :: \mathit{Legal}(v_R)\Big) \end{array}$$

Lemma B.5 $\vdash_{\mathcal{A}} \mathit{Green}(0)$

Proof:

$\langle 1 \rangle 1.$ $t[f(0)] = t[f(0)]$ — why not?
$\langle 1 \rangle 2.$ $d[f(0)] \geq 0$ — by corollary B.4
$\langle 1 \rangle 3.$ Q.E.D. — by $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$

■

Lemma B.6 $\vdash_{\mathcal{A}} \Big(\forall v_R : v_R \in V_R \setminus \{0\} :: \mathit{Green}(v_R - 1) \wedge \mathit{Green}(v_R) \Rightarrow \neg * \mathcal{A}(v_R)\Big)$

Proof:

LET: $v_R \in V_R \setminus \{0\}$ arbitrary
ASSUME: $\mathit{Green}(v_R - 1) \wedge \mathit{Green}(v_R)$
PROVE: $\neg * \mathcal{A}(v_R)$ by case analysis on $v_R = \mathit{First}(f(v_R))$ and $v_R \neq \mathit{First}(f(v_R))$
CASE: $v_R = \mathit{First}(f(v_R))$
  $\langle 2 \rangle 1.$ $t[f(v_R - 1)]$
    $= t[f(0)]$ by $\mathit{Green}(v_R - 1)$
    $= t[f(v_R)]$ by $\mathit{Green}(v_R)$
  $\langle 2 \rangle 2.$ Q.E.D. — by $v_R = \mathit{First}(f(v_R))$ and $\langle 2 \rangle 1$
CASE: $v_R > \mathit{First}(f(v_R))$
  $\langle 2 \rangle 1.$ $d[f(v_R)] \geq v_R$ — by $\mathit{Green}(v_R)$
  $\langle 2 \rangle 2.$ Q.E.D. — by $v_R > \mathit{First}(f(v_R))$ and $\langle 2 \rangle 1$

■

Lemma B.7 $\vdash_{\mathcal{A}} \Big(\forall v_R : v_R \in V_R \setminus \{0\} :: \mathit{Red}(v_R - 1) \wedge \mathit{Red}(v_R) \Rightarrow \neg * \mathcal{A}(v_R)\Big)$

Proof:

LET: $v_R \in V_R \setminus \{0\}$ arbitrary
ASSUME: 1. $\mathit{Red}(v_R - 1) \wedge \mathit{Red}(v_R)$
2. $*\mathcal{A}(v_R)$ for the sake of contradiction
PROVE: $\bot$
$\langle 1\rangle 1$. $d[f(v_R - 1)] \geq v_R - 1$ by $*\mathcal{A}(v_R)$
$\langle 1\rangle 2$. $t[f(v_R - 1)] = t[f(0)] - 1 \bmod Q$ by $\langle 1\rangle 1$ and $\mathit{Red}(v_R - 1)$
$\langle 1\rangle 3$. $v_R = \mathit{First}(f(v_R))$
$\Leftrightarrow$ $t[f(v_R)] \neq t[f(v_R - 1)]$ by $*\mathcal{A}(v_R)$ and $v_R \neq 0$
$\Leftrightarrow$ $t[f(v_R)] \neq t[f(0)] - 1 \bmod Q$ by $\langle 1\rangle 2$
$\langle 1\rangle 4$. $d[f(v_R)] < v_R$
$\Rightarrow$ $t[f(v_R)] = t[f(0)]$ by $\mathit{Red}(v_R)$
$\Rightarrow$ $v_R = \mathit{First}(f(v_R))$ by $\langle 1\rangle 3$
$\Rightarrow$ $d[f(v_R)] \geq v_R$ by *Domain*
$\langle 1\rangle 5$. $d[f(v_R)] \geq v_R$
$\Rightarrow$ $t[f(v_R)] = t[f(0)] - 1 \bmod Q$ by $\mathit{Red}(v_R)$
$\Rightarrow$ $v_R > \mathit{First}(f(v_R))$ by $\langle 1\rangle 3$ and fact (F2)
$\Rightarrow$ $d[f(v_R)] < v_R$ by $*\mathcal{A}(v_R)$ and $v_R \neq 0$
$\langle 1\rangle 6$. Q.E.D. by $\langle 1\rangle 4$ $d[f(v_R)] \geq v_R$ and by $\langle 1\rangle 5$ $d[f(v_R)] < v_R$

■

Lemma B.8
$$\vdash_{\mathcal{A}} \left(\begin{array}{l} \forall v_R : v_R \in V_R \setminus \{0\} :: \\ \quad \Big(\forall u_R : u_R \in V_R \wedge u_R < v_R :: \mathit{Green}(u_R)\Big) \wedge \neg \mathit{Green}(v_R) \Rightarrow * \mathcal{A}(v_R) \end{array}\right)$$

Proof:

LET: $v_R \in V_R \setminus \{0\}$ arbitrary
ASSUME: 1. $\neg \mathit{Green}(v_R)$
2. $\Big(\forall u_R : u_R \in V_R \wedge u_R < v_R :: \mathit{Green}(u_R)\Big)$
PROVE: $*\mathcal{A}(v_R)$
$\langle 1\rangle 1$. $d[f(v_R - 1)] \geq v_R - 1$ by $v_R > 0$, $v_R - 1 < v_R$ so $\mathit{Green}(v_R - 1)$
$\langle 1\rangle 2$. Q.E.D. by case analysis on $v_R = \mathit{First}(f(v_R))$ and $v_R > \mathit{First}(f(v_R))$
CASE: $v_R = \mathit{First}(f(v_R))$ so $d[f(v_R)] \geq v_R$ by corollary B.4
$\langle 2\rangle 1$. $t[f(v_R)] \neq t[f(0)]$ by $\neg \mathit{Green}(v_R)$ and $d[f(v_R)] \geq v_R$
$= t[f(v_R - 1)]$ by $\mathit{Green}(v_R - 1)$
$\langle 2\rangle 2$. Q.E.D. by $v_R \neq 0$, $v_R = \mathit{First}(f(v_R))$, $\langle 2\rangle 1$ and $\langle 1\rangle 1$
CASE: $v_R > \mathit{First}(f(v_R))$
$\langle 2\rangle 1$. $t[f(v_R)] = t[f(\mathit{First}(f(v_R)))]$ by $f(v_R) = f(\mathit{First}(f(v_R)))$
$= t[f(0)]$ by $\mathit{First}(f(v_R)) < v_R$ so $\mathit{Green}(\mathit{First}(f(v_R)))$
$\langle 2\rangle 2$. $t[f(v_R)] = t[f(v_R - 1)]$ by $\langle 2\rangle 1$ and $\mathit{Green}(v_R - 1)$
$\langle 2\rangle 3$. $d[f(v_R - 1)] \geq v_R - 1$ by $\mathit{Green}(v_R - 1)$

$\langle 2\rangle 4$. $d[f(v_R)] < v_R$ by $\langle 2\rangle 1$ and $\neg Green(v_R)$
$\langle 2\rangle 5$. Q.E.D. by $v_R \neq 0$, $v_R > First(f(v_R))$, $\langle 2\rangle 2$, $\langle 2\rangle 3$, $\langle 1\rangle 1$ and $\langle 2\rangle 4$

■

Lemma B.9 $\vdash_{\mathcal{A}} \Big(\forall v_R : v_R \in V_R :: Red(v_R) \Rightarrow \neg Green(v_R)\Big)$

Proof:

LET: $v_R \in V_R$ arbitrary
ASSUME: $Red(v_R) \wedge d[f(v_R)] \geq v_R$
PROVE: $t[f(v_R)] \neq t[f(0)]$
$\langle 1\rangle 1$. $t[f(v_R)] = t[f(0)] - 1 \bmod Q$ by $Red(v_R)$ and $d[f(v_R)] \geq v_R$
$\neq t[f(0)]$

■

Lemma B.10 $\vdash_{\mathcal{A}} Green(n_R - 1) \Leftrightarrow *\mathcal{A}(0)$

Proof:

$\langle 1\rangle 1$. $d[f(n_R - 1)]$
$\leq$ $Last(f(n_R - 1))$ by corollary B.4
$=$ $n_R - 1$ $\leq$ by $Last(v) \leq n_R - 1$, $\geq$ by $n_R - 1 \in Sim(f(n_R - 1))$
$\langle 1\rangle 2$. $*\mathcal{A}(0)$
$\Leftrightarrow$ $t[f(n_R - 1)] = t[f(0)] \wedge d[f(n_R - 1)] = n_R - 1$ by def. of $\mathcal{A}(0)$
$\Leftrightarrow$ $t[f(n_R - 1)] = t[f(0)] \wedge d[f(n_R - 1)] \geq n_R - 1$ by $\langle 1\rangle 1$
$\Leftrightarrow$ $Green(n_R - 1)$

■

Theorem B.11
$$\vdash_{\mathcal{A}} \left(\begin{array}{l}\forall v_R : v_R \in V_R :: Legal(v_R) \Rightarrow \wedge\ *\mathcal{A}(v_R + 1 \bmod n_R) \\ \qquad\qquad\qquad\qquad\qquad\qquad \wedge \Big(\forall u_R : u_R \in V_R \setminus \{v_R + 1 \bmod n_R\} :: \neg * \mathcal{A}(u_R)\Big)\end{array}\right)$$

Proof:

LET: $v_R \in V_R$ arbitrary
ASSUME: $Legal(v_R)$ i.e. $\wedge \Big(\forall u_R : u_R \in V_R \wedge u_R \leq v_R :: Green(u_R)\Big)$
$\wedge \Big(\forall u_R : u_R \in V_R \wedge u_R > v_R :: Red(u_R)\Big)$
$\langle 1\rangle 1$. $*\mathcal{A}(v_R + 1 \bmod n_R)$ by case analysis on $v_R = n_R - 1$ and $v_R < n_R - 1$
CASE: $v_R = n_R - 1$ so $Green(n_R - 1)$ and by lemma B.10 $*\mathcal{A}(0)$
CASE: $v_R < n_R - 1$ so $v_R + 1 \bmod n_R = v_R + 1$
$\langle 2\rangle 1$. $v_R + 1 \neq 0$ by $v_R \in V_R$ so $v_R \geq 0$
$\langle 2\rangle 2$. $\Big(\forall u_R : u_R \in V_R \wedge u_R < v_R + 1 :: Green(u_R)\Big)$ by assumption
$\langle 2\rangle 3$. $Red(v_R + 1)$ by assumption
$\langle 2\rangle 4$. $*\mathcal{A}(v_R + 1)$ by lemma B.9, lemma B.8$[v_R := v_R + 1]$ and $\langle 2\rangle 1 \ldots \langle 2\rangle 3$
$\langle 1\rangle 2$. $\Big(\forall u_R : u_R \in V_R \setminus \{v_R + 1 \bmod n_R\} :: \neg * \mathcal{A}(u_R)\Big)$

LET: $u_R \in V_R$ arbitrary s.t. $u_R \neq v_R + 1 \bmod n_R$
PROVE: $\neg * \mathcal{A}(u_R)$ by case analysis on $u_R = 0$, $0 < u_R \leq v_R$ and $u_R > v_R + 1$
CASE: $u_R = 0$ so $v_R < n_R - 1$ (by $u_R \neq v_R + 1 \bmod n_R$)
$\langle 3 \rangle 1$. $Red(n_R - 1)$ by assumption and $v_R < n_R - 1$
$\langle 3 \rangle 2$. $\neg Green(n_R - 1)$ by $\langle 3 \rangle 1$ and lemma B.9$[v_R := n_R - 1]$
$\langle 3 \rangle 3$. $\neg * \mathcal{A}(0)$ by $\langle 3 \rangle 2$ and lemma B.10
CASE: $0 < u_R \leq v_R$
$\langle 3 \rangle 1$. $Green(u_R - 1) \wedge Green(u_R)$ by assumption
$\langle 3 \rangle 2$. $\neg * \mathcal{A}(u_R)$ by $u_R \neq 0$ and lemma B.6$[v_R := u_R]$
CASE: $v_R + 1 < u_R$
$\langle 3 \rangle 1$. $Red(u_R - 1) \wedge Red(u_R)$ by $u_R - 1 > v_R$ and assumption
$\langle 3 \rangle 2$. $\neg * \mathcal{A}(u_R)$ by $\langle 3 \rangle 1$, $u_R - 1 > v_R \geq 0$ and lemma B.7$[v_R := u_R]$
$\langle 1 \rangle 3$. Q.E.D. by $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$

■

Theorem B.12 $\vdash \Phi \wedge Legal(v_R) \Rightarrow \langle \mathcal{A}(v_R + 1 \bmod n_R) \rangle$

Proof:

$\langle 1 \rangle 1$. $\vdash_{\mathcal{A}} Legal(v_R) \Rightarrow \left(\bigwedge u_R, w_R : u_R, w_R \in V_R \wedge u_R \neq w_R :: * \mathcal{A}(u_R) \Rightarrow \neg * \mathcal{A}(w_R) \right)$
ASSUME: $Legal(v_R)$
LET: $u_R, w_R \in V_R$ arbitrary s.t. $u_R \neq w_R$
ASSUME: $* \mathcal{A}(u_R)$
PROVE: $\neg * \mathcal{A}(w_R)$
$\langle 2 \rangle 1$. $u_R = v_R + 1 \bmod n_R$ by $Legal(v_R)$, $* \mathcal{A}(u_R)$ and lemma B.11
$\langle 2 \rangle 2$. $w_R \neq v_R + 1 \bmod n_R$ by $\langle 2 \rangle 1$ and $u_R \neq w_R$
$\langle 2 \rangle 3$. Q.E.D. by $Legal(v_R)$, $\langle 2 \rangle 2$ and lemma B.11
$\langle 1 \rangle 2$. $\vdash \left(\bigwedge u_R : u_R \in V_R :: Legal(v_R) \wedge [\mathcal{T}] \wedge * \mathcal{A}(u_R) \Rightarrow \langle \mathcal{A}(u_R) \rangle \right)$
by $\langle 1 \rangle 1$ and the following theorem of which proof can obtained from the authors.
If $\vdash_{\mathcal{A}} P \Rightarrow \left(\bigwedge i, j : i, j \in I \wedge i \neq j :: * \mathcal{A}(i) \Rightarrow \neg * \mathcal{A}(j) \right)$
then $\vdash \left(\bigwedge i : i \in I :: P \wedge [A] \wedge \mathrm{WF}(\mathcal{A}) \wedge * \mathcal{A}(i) \Rightarrow [\mathcal{A}(i)] \right)$
where $\mathcal{A} \triangleq \left(\bigvee i : i \in I :: \mathcal{A}(i) \right)$
$\langle 1 \rangle 3$. $\vdash Legal(v_R) \Rightarrow * \mathcal{A}(v_R + 1 \bmod n_R)$ by lemma B.11
$\langle 1 \rangle 4$. $\vdash \Phi \Rightarrow [\mathcal{T}]$ by $\vdash \Box \phi \Rightarrow \phi$
$\langle 1 \rangle 5$. Q.E.D. by $\langle 1 \rangle 4$, $\langle 1 \rangle 3$ and $\langle 1 \rangle 2$

■

B.3 Staying legitimate

We have now proven that in a legitimate configuration only one occurrence is enabled and that it will be executed next. We will now analyse what happens if it is executed, i.e. we prove the next occurrence will become the only one enabled and that we are still in a legitimate configuration.

Lemma B.13 $\vdash_{\mathcal{A}} \left(\forall v_R : v_R \in V_R \setminus \{0\} :: Green(v_R - 1) \wedge \mathcal{A}(v_R) \Rightarrow Green(v_R)' \right)$

Proof:

LET: $v_R \in V_R \setminus \{0\}$ arbitrary
ASSUME: $Green(v_R - 1) \wedge \mathcal{A}(v_R)$
PROVE: $Green(v_R)'$

$\langle 1 \rangle 1.$ $t[f(v_R)]' = t[f(v_R - 1)]$ by $\mathcal{A}(v_R)$ and $v_R \neq 0$
$= t[f(0)]$ by $Green(v_R - 1)$
$= t[f(0)]'$ by (F5), $\mathcal{A}(v_R)$ and $v_R \neq 0$

$\langle 1 \rangle 2.$ $d[f(v_R)]' = v_R$ by $\mathcal{A}(v_R)$
$\langle 1 \rangle 3.$ Q.E.D. by $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$

■

Lemma B.14
$\vdash_{\mathcal{A}} \left(\forall u_R, v_R : u_R, v_R \in V_R \wedge v_R < u_R :: Green(v_R) \wedge \mathcal{A}(u_R) \Rightarrow Green(v_R)'\right)$

Proof:

LET: $u_R, v_R \in V_R$ arbitrary s.t. $v_R < u_R$ so $u_R \neq 0$
ASSUME: $Green(v_R) \wedge \mathcal{A}(u_R)$
PROVE: $Green(v_R)'$ by case analysis on $f(v_R) = f(u_R)$ and $f(v_R) \neq f(u_R)$
CASE: $f(v_R) = f(u_R)$

$\langle 2 \rangle 1.$ $t[f(u_R - 1)] = t[f(0)]$

$\langle 3 \rangle 1.$ $First(f(u_R)) \leq v_R$ by (F2) and $f(v_R) = f(u_R)$
$< u_R$

$\langle 3 \rangle 2.$ $t[f(u_R - 1)] = t[f(u_R)]$ by $\mathcal{A}(u_R)$, $u_R \neq 0$ and $\langle 3 \rangle 1$
$= t[f(v_R)]$ by $f(u_R) = f(v_R)$
$= t[f(0)]$ by $Green(v_R)$

$\langle 2 \rangle 2.$ $t[f(v_R)]' = t[f(u_R)]'$ by $f(v_R) = f(u_R)$
$= t[f(u_R - 1)]$ by (F4), $\mathcal{A}(u_R)$ and $u_R \neq 0$
$= t[f(0)]$ by $\langle 2 \rangle 1$
$= t[f(0)]'$ by (F5), $\mathcal{A}(u_R)$ and $u_R \neq 0$

$\langle 2 \rangle 3.$ $d[f(v_R)]' = d[f(u_R)]'$ by $f(v_R) = f(u_R)$
$= u_R$ by (F3), $\mathcal{A}(u_R)$
$> v_R$

$\langle 2 \rangle 4.$ Q.E.D. by $\langle 2 \rangle 2$ and $\langle 2 \rangle 3$

CASE: $f(v_R) \neq f(u_R)$

$\langle 2 \rangle 1.$ $Unchanged\ (t[f(v_R)], d[f(v_R)])$ by $\mathcal{A}(u_R)$ and $f(u_R) \neq f(v_R)$
$\langle 2 \rangle 2.$ $Unchanged\ (t[f(0)])$ by (F5), $\mathcal{A}(u_R)$ and $u_R \neq 0$
$\langle 2 \rangle 3.$ Q.E.D. by $Green(v_R)$, $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$

■

Lemma B.15
$\vdash_{\mathcal{A}} \left(\forall u_R, v_R : u_R, v_R \in V_R \wedge 0 < u_R < v_R :: Green(u_R - 1) \wedge \mathcal{A}(u_R) \wedge Red(v_R) \Rightarrow Red(v_R)'\right)$

Proof:

LET: $u_R, v_R \in V_R$ arbitrary s.t. $0 < u_R < v_R$
ASSUME: $Green(u_R - 1) \wedge \mathcal{A}(u_R) \wedge Red(v_R)$
PROVE: $Red(v_R)'$ by case analysis on $f(v_R) = f(u_R)$ and $f(v_R) \neq f(u_R)$
CASE: $f(v_R) = f(u_R)$

$\langle 2 \rangle 1.$ $d[f(v_R)]' = d[f(u_R)]'$ by $f(v_R) = f(u_R)$
$= u_R$ by (F3) and $\mathcal{A}(u_R)$
$< v_R$

$\langle 2 \rangle 2.$ $t[f(v_R)]' = t[f(u_R)]'$ by $f(v_R) = f(u_R)$
$= t[f(u_R - 1)]$ by (F4), $\mathcal{A}(u_R)$ and $u_R > 0$
$= t[f(0)]$ by $Green(u_R - 1)$
$= t[f(0)]'$ by (F5), $\mathcal{A}(u_R)$ and $u_R > 0$

$\langle 2 \rangle 3.$ Q.E.D. by $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$

CASE: $f(v_R) \neq f(u_R)$

$\langle 2 \rangle 1.$ *Unchanged* $(t[f(v_R)], d[f(v_R)])$ by $\mathcal{A}(u_R)$ and $f(v_R) \neq f(u_R)$
$\langle 2 \rangle 2.$ *Unchanged* $(t[f(0)])$ by (F5), $\mathcal{A}(u_R)$ and $u_R > 0$
$\langle 2 \rangle 3.$ Q.E.D. by $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$

■

The following lemma will not be used in this subsection, but in analysing the second phase. However, since it is very similar to the previous lemmas, we give it here.

Lemma B.16

$$\vdash_{\mathcal{A}} \left(\begin{array}{l} \forall u_R, v_R : u_R, v_R \in V_R \wedge 0 < u_R :: \\ \qquad (\neg Green(u_R - 1) \vee u_R < v_R) \wedge \mathcal{A}(u_R) \wedge \neg Green(v_R) \Rightarrow \neg Green(v_R)' \end{array} \right)$$

Proof:

LET: $u_R, v_R \in V_R$ arbitrary s.t. $0 < u_R$
ASSUME: 1. $\neg Green(u_R - 1) \vee u_R < v_R$
2. $\mathcal{A}(u_R) \wedge \neg Green(v_R)$
PROVE: $\neg Green(v_R)'$ by case analysis on $f(u_R) \neq f(v_R)$ and $f(u_R) = f(v_R)$
$\langle 1 \rangle 1.$ *Unchanged* $(t[f(0)])$ by (F5), $\mathcal{A}(u_R)$ and $u_R \neq 0$
CASE: $f(u_R) \neq f(v_R)$

$\langle 2 \rangle 1.$ *Unchanged* $(t[f(v_R)], d[f(v_R)])$ by $\mathcal{A}(u_R)$ and $f(v_R) \neq f(u_R)$
$\langle 2 \rangle 2.$ Q.E.D. by $\neg Green(v_R)$, $\langle 1 \rangle 1$ and $\langle 2 \rangle 1$

CASE: $f(u_R) = f(v_R)$

$\langle 2 \rangle 1.$ Q.E.D. by case analysis on $\neg Green(u_R - 1)$ and $u_R < v_R$

CASE: $\neg Green(u_R - 1)$

$\langle 3 \rangle 1.$ $d[f(u_R - 1)] \geq u_R - 1$ by $\mathcal{A}(u_R)$ and $u_R > 0$
$\langle 3 \rangle 2.$ $t[f(u_R - 1)] \neq t[f(0)]$ by $\neg Green(u_R - 1)$ and $\langle 3 \rangle 1$
$\langle 3 \rangle 3.$ $t[f(v_R)]' = t[f(u_R)]'$ by $f(v_R) = f(u_R)$
$= t[f(u_R - 1)]$ by (F4), $\mathcal{A}(u_R)$ and $u_R > 0$
$\neq t[f(0)]$ by $\langle 3 \rangle 2$
$= t[f(0)]'$ by $\langle 1 \rangle 1$

$\langle 3 \rangle 4.$ Q.E.D. by $\langle 3 \rangle 3$

CASE: $u_R < v_R$

$\langle 3\rangle 1.$ $d[f(v_R)]' = d[f(u_R)]'$ by $f(v_R) = f(u_R)$
$= u_R$ by (F3), $\mathcal{A}(u_R)$
$< v_R$
$\langle 3\rangle 2.$ Q.E.D. by $\langle 3\rangle 1$

■

Lemma B.17 $\vdash_{\mathcal{A}} \mathit{Legal}(n_R - 1) \wedge \mathcal{A}(0) \Rightarrow \mathit{Legal}(0)'$

Proof:

ASSUME: $\mathit{Legal}(n_R - 1) \wedge \mathcal{A}(0)$
LET: $v_R \in V_R \setminus \{0\}$ arbitrary, so $\mathit{Green}(v_R)$ by $\mathit{Legal}(n_R - 1)$
PROVE: $\mathit{Red}(v_R)'$ so $\mathit{Legal}(0)'$ by $v_R \in V_R \setminus \{0\}$ arbitrary and $\vdash_{\mathcal{A}} \mathit{Green}(0)$ (lemma B.5).
Proof by case analysis on $f(v_R) = f(0)$ and $f(v_R) \neq f(0)$
CASE: $f(v_R) = f(0)$
$\langle 2\rangle 1.$ $d[f(v_R)]' = d[f(0)]'$ by $f(v_R) = f(0)$
$= 0$ by (F3), $\mathcal{A}(0)$
$< v_R$ by $v_R \neq 0$
$\langle 2\rangle 2.$ $t[f(v_R)]' = t[f(0)]'$ by $f(v_R) = f(0)$
$\langle 2\rangle 3.$ Q.E.D. by $\langle 2\rangle 1$ and $\langle 2\rangle 2$
CASE: $f(v_R) \neq f(0)$
$\langle 2\rangle 1.$ $d[f(v_R)]' = d[f(v_R)]$ by $\mathcal{A}(0)$ and $f(v_R) \neq f(0)$
$\geq v_R$ by $\mathit{Green}(v_R)$
$\langle 2\rangle 2.$ $t[f(v_R)]' = t[f(v_R)]$ by $\mathcal{A}(0)$ and $f(v_R) \neq f(0)$
$= t[f(0)]$ by $\mathit{Green}(v_R)$
$= t[f(0)]' - 1 \bmod Q$ by $\mathcal{A}(0)$
$\langle 2\rangle 3.$ Q.E.D. by $\langle 2\rangle 1$ and $\langle 2\rangle 2$

■

Theorem B.18
$\vdash_{\mathcal{A}} \left(\forall v_R : v_R \in V_R :: \mathit{Legal}(v_R) \wedge \mathcal{A}(v_R + 1 \bmod n_R) \Rightarrow \mathit{Legal}(v_R + 1 \bmod n_R)'\right)$

Proof:

LET: $v_R \in V_R$ arbitrary
ASSUME: $\mathit{Legal}(v_R) \wedge \mathcal{A}(v_R + 1 \bmod n_R)$
PROVE: $\mathit{Legal}(v_R + 1 \bmod n_R)'$ by case analysis on $v_R = n_R - 1$ and $v_R < n_R - 1$
CASE: $v_R = n_R - 1$ by lemma B.17
CASE: $v_R < n_R - 1$ so $v_R + 1 \bmod n_R = v_R + 1$
LET: $u_R \in V_R$ arbitrary
$\langle 2\rangle 1.$ $u_R \leq v_R \Rightarrow \mathit{Green}(u_R)'$
ASSUME: $u_R \leq v_R$
PROVE: $\mathit{Green}(u_R)'$
$\langle 3\rangle 1.$ $u_R < v_R + 1$ by $u_R \leq v_R$
$\langle 3\rangle 2.$ $\mathit{Green}(u_R)$ by $u_R \leq v_R$ and $\mathit{Legal}(v_R)$

$\langle 3\rangle 3.\ \mathcal{A}(v_R+1)$ by $\mathcal{A}(v_R+1 \bmod n_R)$
$\langle 3\rangle 4.$ Q.E.D. by lemma B.14 $[u_R, v_R := v_R+1, u_R]$ and $\langle 3\rangle 1 \ldots \langle 3\rangle 3$
$\langle 2\rangle 2.\ u_R = v_R+1 \Rightarrow \mathit{Green}(u_R)'$
ASSUME: $u_R = v_R+1$
PROVE: $\mathit{Green}(u_R)'$
$\langle 3\rangle 1.\ u_R \neq 0$ by $u_R = v_R+1 > 0$
$\langle 3\rangle 2.\ \mathit{Green}(u_R-1)$ by $u_R - 1 = v_R \leq v_R$ and $\mathit{Legal}(v_R)$
$\langle 3\rangle 3.\ \mathcal{A}(u_R)$ by $\mathcal{A}(v_R+1)$ and $u_R = v_R+1$
$\langle 3\rangle 4.$ Q.E.D. by lemma B.13 $[v_R := u_R]$ and $\langle 3\rangle 1 \ldots \langle 3\rangle 3$
$\langle 2\rangle 3.\ u_R > v_R+1 \Rightarrow \mathit{Red}(u_R)'$
ASSUME: $u_R > v_R+1$
PROVE: $\mathit{Red}(u_R)'$
$\langle 3\rangle 1.\ 0 < v_R+1 < u_R$
$\langle 3\rangle 2.\ \mathit{Green}(v_R)$ by $v_R \leq v_R$ and $\mathit{Legal}(v_R)$
$\langle 3\rangle 3.\ \mathcal{A}(v_R+1)$ by $\mathcal{A}(v_R+1 \bmod n_R)$
$\langle 3\rangle 4.\ \mathit{Red}(u_R)$ by $u_R > v_R+1 > v_R$ and $\mathit{Legal}(v_R)$
$\langle 3\rangle 5.$ Q.E.D. by lemma B.15 $[u_R, v_R := v_R+1, u_R]$ and $\langle 3\rangle 1 \ldots \langle 3\rangle 4$
$\langle 2\rangle 4.$ Q.E.D. by $\langle 2\rangle 1$, $\langle 2\rangle 2$, $\langle 2\rangle 3$ and $u_R \in V_R$ arbitrary.

■

Corollary B.19 $\vdash \Phi \wedge \mathit{Legitimate} \Rightarrow \Box \mathit{Legitimate}$

Proof:

$\langle 1\rangle 1.$ If $\vdash P \wedge \phi \Rightarrow [P']$ then $\vdash P \wedge \Box\phi \Rightarrow \Box P$
$\langle 2\rangle 1.\ \vdash\ P \wedge \Box\phi$
$\Rightarrow\ P \wedge \Box(\phi \wedge (P \Rightarrow [P']))$ by $\vdash P \wedge \phi \Rightarrow [P']$
$\Rightarrow\ P \wedge \Box(P \Rightarrow [P'])$
$\Rightarrow\ \Box P$ by rule TLA1
Rule TLA1 says
If $\vdash_{\mathcal{A}} P \wedge (f' = f) \Rightarrow P'$
then $\vdash \Box P \Leftrightarrow P \wedge \Box[P \Rightarrow P']_f$
Let f be the tuple containing all program variables used (in particular all program variables mentioned in P), then obviously $\vdash_{\mathcal{A}} P \wedge (f' = f) \Rightarrow P'$. Furthermore $\vdash (P \Rightarrow [P']) \Leftrightarrow [P \Rightarrow P']$ and also $\vdash [\mathcal{A}] \Rightarrow [\mathcal{A}]_f$ for any state-function f.
$\langle 1\rangle 2.\ \mathit{Legitimate} \wedge \Phi \Rightarrow [\mathit{Legitimate}']$
ASSUME: $\mathit{Legitimate} \wedge \Phi$
LET: $v_R \in V_R$ s.t. $\mathit{Legal}(v_R)$ such v_R exists by *Legitimate*
PROVE: $\langle \mathit{Legitimate}' \rangle$ so $[\mathit{Legitimate}']$ $(\wedge \langle \top \rangle)$
$\langle 2\rangle 1.\ \langle \mathcal{A}(v_R+1 \bmod n_R) \rangle$ by theorem B.12 and Φ and $\mathit{Legal}(v_R)$
$\langle 2\rangle 2.\ \langle \mathit{Legal}(v_R) \wedge \mathcal{A}(v_R+1 \bmod n_R) \rangle$ by $\langle 2\rangle 1$, $\mathit{Legal}(v_R)$ and $\vdash P \wedge \langle A \rangle \Leftrightarrow \langle P \wedge \mathcal{A} \rangle$
$\langle 2\rangle 3.\ \langle \mathit{Legal}(v_R+1 \bmod n_R)' \rangle$ by $\langle 2\rangle 2$ and $\langle\rangle$-mono of theorem B.18
$\langle 2\rangle 4.$ Q.E.D. by $\langle 2\rangle 3$ a v_R exists s.t. $\langle \mathit{Legal}(v_R)' \rangle$ so $\langle \mathit{Legitimate}' \rangle$
$\langle 1\rangle 3.\ \vdash \mathit{Legitimate} \wedge \Box\Phi \Rightarrow \Box \mathit{Legitimate}$ by $\langle 1\rangle 1$ and $\langle 1\rangle 2$
$\langle 1\rangle 4.\ \vdash \Box\Phi \Leftrightarrow \Phi$ by missing initial prediate

⟨1⟩5. Q.E.D. by ⟨1⟩3 and ⟨1⟩4

■

Lemma B.20 $\vdash_{\mathcal{A}} \left(\forall v_R : v_R \in V_R \setminus \{0\} :: \neg * \mathcal{A}(v_R)\right) \Rightarrow \mathit{Legal}(n_R - 1)$

Proof:

ASSUME: $\left(\forall v_R : v_R \in V_R \setminus \{0\} :: \neg * \mathcal{A}(v_R)\right)$
PROVE: $\left(\forall v_R : v_R \in V_R :: \mathit{Green}(v_R)\right)$ by induction on v_R
CASE: $v_R = 0$. $\mathit{Green}(0)$ by lemma B.5
CASE: $v_R = u_R + 1 < n_R$ so $v_R \neq 0$
  ASSUME: $\left(\forall w_R : w_R \in V_R \wedge w_R < v_R :: \mathit{Green}(w_R)\right)$ induction hypothesis
  PROVE: $\mathit{Green}(v_R)$ by case analysis on $v_R = \mathit{First}(f(v_R))$ and $v_R > \mathit{First}(f(v_R))$
  CASE: $v_R = \mathit{First}(f(v_R))$
    ⟨3⟩1. $d[f(v_R - 1)] \geq v_R - 1$ by $\mathit{Green}(v_R - 1)$
    ⟨3⟩2. $t[f(v_R)]$
      $= t[f(v_R - 1)]$ by $\neg * \mathcal{A}(v_R)$, $v_R \neq 0$, $v_R = \mathit{First}(f(v_R))$ and ⟨3⟩1
      $= t[f(0)]$ by $\mathit{Green}(v_R - 1)$
    ⟨3⟩3. $d[f(v_R)]$
      $\geq \mathit{First}(f(v_R))$ by corollary B.4
      $= v_R$ by case assumption
    ⟨3⟩4. Q.E.D. by ⟨3⟩2 and ⟨3⟩3
  CASE: $v_R > \mathit{First}(f(v_R))$
    ⟨3⟩1. $d[f(v_R - 1)] \geq v_R - 1$ by $\mathit{Green}(v_R - 1)$
    ⟨3⟩2. $t[f(v_R)]$
      $= t[f(\mathit{First}(f(v_R)))]$ by $f(v_R) = f(\mathit{First}(f(v_R)))$
      $= t[f(0)]$ by $\mathit{First}(v_R) < v_R$ so $\mathit{Green}(\mathit{First}(f(v_R))$
    ⟨3⟩3. $t[f(v_R)]$
      $= t[f(0)]$ by ⟨3⟩2
        $t[f(v_R - 1)]$ by $\mathit{Green}(v_R - 1)$
    ⟨3⟩4. $d[f(v_R)] \geq v_R$ by $\neg * \mathcal{A}(v_R)$, $v_R > \mathit{First}(f(v_R))$, ⟨3⟩1 and ⟨3⟩3
    ⟨3⟩5. Q.E.D. by ⟨3⟩2 and ⟨3⟩4

■

Corollary B.21 $\vdash_{\mathcal{A}} \left(\exists v_R : v_R \in V_R :: *\mathcal{A}(v_R)\right)$

Proof: If $\mathcal{A}(v_R)$, for some $v_R \in V_R \setminus \{0\}$, is enabled then the corollary holds. Otherwise it follows by lemma B.20 that $\mathit{Legal}(n_R - 1)$ holds and by theorem B.11 $\mathcal{A}(0)$ is enabled. ■

B.4 The main decreasing function Chaos

In this section we define the function *Chaos* and show that its range is finite and execution of any non-root occurrence decreases this function.

$$
\begin{aligned}
W(v) &\triangleq f(First(v) - 1 \bmod n_R) \\
vTu &\triangleq u \neq f(0) \wedge v = W(u) \\
Weight(v) &\triangleq 1 + \Big(\sum u : u \in V \wedge vTu :: Weight(u)\Big) \\
Distance(v) &\triangleq \Big(\#v_R : v_R \in Sim(v) :: v_R < d[v]\Big) \\
M &\triangleq \Big(\max\, v : v \in V :: |Sim(v)|\Big) \\
SDistance &\triangleq \Big(\sum v : v \in V :: Distance(v)\Big) \\
SWeight &\triangleq \left(\sum v : v \in V :: \left\{ \begin{array}{ll} Weight(v) & \text{if } W(v)Tv \wedge t[v] \neq t[W(v)] \\ 0 & \text{otherwise} \end{array} \right.\right) \\
Chaos &\triangleq M \cdot SWeight - SDistance
\end{aligned}
$$

Lemma B.22 $\vdash_{\mathcal{A}} \Big(\forall v : v \in V :: 0 \leq Distance(v) < |Sim(v)|\Big)$

Proof:

LET: $v \in V$ arbitrary

$\langle 1 \rangle 1$. $Distance(v) \geq 0$ by $\Big(\#v : R(v) :: P(v)\Big) \geq 0$

$\langle 1 \rangle 2$.

$$
\begin{aligned}
& Distance(v) \\
= \;& \Big(\#v_R : v_R \in Sim(v) :: v_R < d[v]\Big) && \text{by definition of } Distance \\
\leq \;& \Big(\#v_R : v_R \in Sim(v) :: v_R < Last(v)\Big) && \text{by } d[v] \leq Last(v) \\
= \;& |Sim(v)| - 1 && \text{by definition of } Last
\end{aligned}
$$

$\langle 1 \rangle 3$. Q.E.D. by $\langle 1 \rangle 1$ and $\langle 1 \rangle 2$

■

Lemma B.23 $\vdash_{\mathcal{A}} n - n_R \leq -SDistance \leq 0$

Proof:

$$
\begin{aligned}
\langle 1 \rangle 1. \quad & 0 \\
\geq \;& \Big(\sum v : v \in V :: -Distance(v)\Big) && \text{by lemma B.22} \\
\geq \;& \Big(\sum v : v \in V :: -|Sim(v)| + 1\Big) && \text{by lemma B.22} \\
= \;& n - n_R && \text{by } |V| = n \text{ and } \Big(\sum v : v \in V :: |Sim(v)|\Big) = n_R
\end{aligned}
$$

$\langle 1 \rangle 2$. Q.E.D. by $\langle 1 \rangle 1$ and def. of $SDistance$

■

Lemma B.24 *The relation T defines a spanning tree on V with root $f(0)$.*

Proof:

$\langle 1 \rangle 1$. $\vdash_{\mathcal{A}} \Big(\forall u, v : u, v \in V \wedge vTu :: First(v) < First(u)\Big)$

LET: $u, v \in V$ arbitrary s.t. vTu

PROVE: $First(v) < First(u)$

$\langle 2 \rangle 1$. $First(u) > 0$ by vTu so $u \neq f(0)$ and $0 \notin Sim(u)$

$\langle 2\rangle 2$. $First(v)$
$= First(W(u))$ by $vTu \Rightarrow v = W(u)$
$= First(f(First(u) - 1 \bmod n_R))$ by def. of W
$= First(f(First(u) - 1))$ by $\langle 2\rangle 1$
$\leq First(u) - 1$ by (F2)
$< First(u)$

$\langle 1\rangle 2$. $\Big(\forall v : v \in V :: \neg W(v)Tv \Rightarrow v = f(0)\Big)$ by def. of relation T

$\langle 1\rangle 3$. Q.E.D.
For every u there is at most one v s.t. vTu, since $v = W(u)$ and W is a function. By $\langle 1\rangle 1$ there are no cycles in the relation. By $\langle 1\rangle 2$ every u has a parent except for $f(0)$. This proofs T makes up a spanning tree of V with root $f(0)$.

■

Define $S(v)$ as the set of nodes in the subtree with root v.

Lemma B.25 $\vdash_{\mathcal{A}} \Big(\forall v : v \in V :: Weight(v) = |S(v)|\Big)$

Proof:

LET: $v \in V$ arbitrary
PROVE: $Weight(v) = |S(v)|$
$\langle 1\rangle 1$. Q.E.D. by induction on the depth of the subtree with root v.
CASE: $depth = 0$ so $\{u \in V | vTu\} = \emptyset$ and $S(v) = \{v\}$
$\langle 2\rangle 1$. $Weight(v)$
$= 1 + \Big(\sum u : u \in V \wedge vTu :: Weight(u)\Big)$
$= 1$ by $\{u \in V | vTu\} = \emptyset$
$= |S(v)|$ by $S(v) = \{v\}$
CASE: $depth = e + 1$ and by induction hypothesis $\Big(\forall u : u \in V \wedge vTu :: Weight(u) = |S(u)|\Big)$
$\langle 2\rangle 1$. $Weight(v)$
$= 1 + \Big(\sum u : u \in V \wedge vTu :: Weight(u)\Big)$ def. of $Weight(v)$
$= |S(v)|$
since $S(v) = \{v\} \cup \Big(\bigcup u : u \in V \wedge vTu :: S(v)\Big)$ and all $S(u), \{v\}$ are disjoint.

■

Lemma B.26 $\vdash_{\mathcal{A}} 0 \leq SWeight \leq \frac{n(n-1)}{2}$

Proof:

$\langle 1\rangle 1$. $\Big(\forall v : v \in V :: Weight(v) \geq 1\Big)$, since $Weight(v)$ is the number of nodes in $S(v)$. Therefore $SWeight \geq 0$.
$\langle 1\rangle 2$. $\Big(\forall v : v \in V :: 2\Big(\sum u : u \in S(v) :: Weight(u)\Big) \leq Weight(v)(Weight(v) + 1)\Big)$
We prove this by induction to the depth d of the subtree with root v.
CASE: $d = 0$
$\langle 3\rangle 1$. $Weight(v) = 1$ and $S(v) = \{v\}$ and the claim follows

CASE: $d = e + 1$

LET: $v_1, \ldots, v_c$ $(c \neq 0)$ be the children of v

LET: $m = \Big(\max\, i : 1 \leq i \leq c :: \mathit{Weight}(v_i)\Big)$ so $m \geq \mathit{Weight}(v_i)$ for $i = 1, \ldots, c$

ASSUME: $\Big(\forall i : 1 \leq i \leq c :: 2\Big(\sum u : u \in S(v_i) :: \mathit{Weight}(u)\Big) \leq \mathit{Weight}(v_i)(\mathit{Weight}(v_i) + 1)\Big)$
by induction hypothesis

$\langle 3 \rangle 1$. $\mathit{Weight}(v) = 1 + \Big(\sum i : 1 \leq i \leq c :: \mathit{Weight}(v_i)\Big)$ by definition of Weight

$\langle 3 \rangle 2$. $m \leq \Big(\sum i : 1 \leq i \leq c :: \mathit{Weight}(v_i)\Big) - (c - 1)$ by $\mathit{Weight}(v_i) \geq 1$ and let of m

$\langle 3 \rangle 3$.
$$\begin{array}{ll}
 & 2\Big(\sum u : u \in S(v) :: \mathit{Weight}(u)\Big) \\
= & 2\,\mathit{Weight}(v) + 2\Big(\sum i : 1 \leq i \leq c :: \Big(\sum u : u \in S(v_i) :: \mathit{Weight}(u)\Big)\Big) \\
 & \text{by definition of } S(v) \\
\leq & 2\,\mathit{Weight}(v) + \Big(\sum i : 1 \leq i \leq c :: \mathit{Weight}(v_i)(\mathit{Weight}(v_i) + 1)\Big) \\
 & \text{by induc. hyp.} \\
\leq & 2\,\mathit{Weight}(v) + m\Big(\sum i : 1 \leq i \leq c :: \mathit{Weight}(v_i) + 1\Big) \\
 & \text{by let of } m \\
= & 2(1 + \Big(\sum i : 1 \leq i \leq c :: \mathit{Weight}(v_i)\Big)) + m\Big(\sum i : 1 \leq i \leq c :: \mathit{Weight}(v_i) + 1\Big) \\
 & \text{by } \langle 3 \rangle 1 \\
\leq & 2\Big(\sum i : 1 \leq i \leq c :: \mathit{Weight}(v_i) + 1\Big) + m\Big(\sum i : 1 \leq i \leq c :: \mathit{Weight}(v_i) + 1\Big) \\
 & \text{by } c \geq 1 \\
= & (m + 2)\Big(\sum i : 1 \leq i \leq c :: \mathit{Weight}(v_i) + 1\Big) \\
\leq & (\Big(\sum i : 1 \leq i \leq c :: \mathit{Weight}(v_i)\Big) + 2 - (c - 1))\Big(\sum i : 1 \leq i \leq c :: \mathit{Weight}(v_i) + 1\Big) \\
 & \text{by } \langle 3 \rangle 2 \\
= & (\mathit{Weight}(v) + 1 - (c - 1))(\mathit{Weight}(v) + (c - 1)) \\
 & \text{by } \langle 3 \rangle 1 \\
\leq & \mathit{Weight}(v)(\mathit{Weight}(v) + 1) \\
 & \text{by calculation}
\end{array}$$

$\langle 1 \rangle 3$.
$$\begin{array}{lll}
 & \mathit{SWeight} & \\
\leq & \Big(\sum u : u \in V \setminus \{f(0)\} :: \mathit{Weight}(u)\Big) & \text{by } \mathit{Weight}(u) > 0 \\
= & \Big(\sum u : u \in S(f(0)) :: \mathit{Weight}(u)\Big) - \mathit{Weight}(f(0)) & \text{by lemma B.24 } S(f(0)) = V \\
\leq & \frac{\mathit{Weight}(f(0))(\mathit{Weight}(f(0))+1)}{2} - \mathit{Weight}(f(0)) & \text{by } \langle 1 \rangle 2 \\
= & \frac{n(n-1)}{2} & \\
 & \text{by lemma B.24 and lemma B.25 } \mathit{Weight}(f(0)) = n &
\end{array}$$

■

Lemma B.27

$\vdash_{\mathcal{A}} \Big(\forall v_R : v_R \in V_R \setminus \{0\} :: \mathcal{A}(v_R) \wedge v_R > \mathit{First}(f(v_R)) \Rightarrow \mathit{SWeight}' = \mathit{SWeight}\Big)$

Proof:

LET: $v_R \in V_R \setminus \{0\}$ arbitrary

ASSUME: $\mathcal{A}(v_R) \wedge v_R > \mathit{First}(f(v_R))$

PROVE: $SWeight' = SWeight$
$\langle 1\rangle 1$. $\Big(\forall v : v \in V :: Unchanged\ (t[v])\Big)$ by $\mathcal{A}(v_R) \wedge v_R > First(f(v_R))$
$\langle 1\rangle 2$. Q.E.D. by $\langle 1\rangle 1$ and $SWeight$ only depends on the $t[]$-values

■

Lemma B.28
$\vdash_{\mathcal{A}} \Big(\forall v_R : v_R \in V_R \setminus \{0\} :: \mathcal{A}(v_R) \wedge v_R = First(f(v_R)) \Rightarrow SWeight' < SWeight\Big)$

Proof:

LET: $v_R \in V_R \setminus \{0\}$ arbitrary
ASSUME: $\mathcal{A}(v_R) \wedge v_R = First(f(v_R))$
PROVE: $SWeight' < SWeight$
LET: $v = f(v_R)$ and

$$(a) = \left(\sum u : u \in V \wedge u \neq f(0) \wedge u \neq v \wedge v \neq W(u) :: \begin{cases} Weight(u) & \text{if } W(u)Tu \wedge t[u] \neq t[W(u)] \\ 0 & \text{otherwise} \end{cases}\right)$$

$$(b) = \left(\sum u : u \in V \wedge vTu :: \begin{cases} Weight(u) & \text{if } W(u)Tu \wedge t[u] \neq t[W(u)] \\ 0 & \text{otherwise} \end{cases}\right)$$

$$(c) = \begin{cases} Weight(v) & \text{if } W(v)Tv \wedge t[v] \neq t[W(v)] \\ 0 & \text{otherwise} \end{cases}$$

$\langle 1\rangle 1$. $v \neq f(0)$ by $v = f(v_R)$ and $First(f(v_R)) = v_R \neq 0$
$\langle 1\rangle 2$. $W(v)Tv$ by $\langle 1\rangle 1$
$\langle 1\rangle 3$. $f(v_R) \neq f(v_R - 1)$ by $v_R = First(f(v_R))$
$\langle 1\rangle 4$. $SWeight = (a) + (b) + (c)$
$\langle 2\rangle 1$. $vTu \Rightarrow First(v) < First(u) \Rightarrow u \neq v$ see proof of lemma B.24
$\langle 2\rangle 2$.
$\wedge\ u \in V$
$\wedge\ \vee\ u \neq f(0) \wedge u \neq v \wedge v \neq W(u)$
$\vee\ vTu$
$\vee\ u = v$
$\Leftrightarrow$ $\wedge\ u \in V$
$\wedge\ \vee\ u \neq f(0) \wedge u \neq v \wedge v \neq W(u)$
$\vee\ u \neq f(0) \wedge u \neq v \wedge v = W(u)$ by $\langle 2\rangle 1$ and $vTu \Leftrightarrow u \neq f(0) \wedge v = W(u)$
$\vee\ u \neq f(0) \wedge u = v$ by $\langle 1\rangle 1$
$\Leftrightarrow$ $u \in V \wedge u \neq f(0)$
$\langle 2\rangle 3$. $u \neq f(0) \wedge u \neq v \wedge v \neq W(u) \Rightarrow v \neq W(u) \Rightarrow \neg vTu$
$\langle 2\rangle 4$. $u \neq f(0) \wedge u \neq v \wedge v \neq W(u) \Rightarrow u \neq v \Rightarrow \neg(u = v)$
$\langle 2\rangle 5$. Q.E.D.
by $\langle 2\rangle 2$ $(a) + (b) + (c)$ exaclty covers all elements summed in $SWeight$. By $\langle 2\rangle 3$ and $\langle 2\rangle 4$, (a) does not cover any element also in (b) or (c) (and vice versa). By $\langle 2\rangle 1$ (b) does not cover any element also in (c) and vice versa.
$\langle 1\rangle 5$. $SWeight' = (a)' + (b)' + (c)'$
The proof is very similar to $\langle 1\rangle 4$ since it only involves reasoning about rigid variables.
$\langle 1\rangle 6$. $(a) = (a)'$

LET: $u \in V$ arbitrary s.t. $u \neq f(0) \wedge u \neq v \wedge v \neq W(u)$
$\langle 2 \rangle 1.\ t[u]' = t[u]$ by $\mathcal{A}(v_R) \wedge f(v_R) = v \neq u$
$\langle 2 \rangle 2.\ t[W(u)]' = t[W(u)]$ by $\mathcal{A}(v_R) \wedge f(v_R) = v \neq W(u)$
$\langle 2 \rangle 3.\ t[u]' = t[W(u)]' \Leftrightarrow t[u] = t[W(u)]$ by $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$
$\langle 2 \rangle 4.$ Q.E.D. by $\langle 2 \rangle 3$ and $Weight(u)$ is fixed for any u
$\langle 1 \rangle 7.\ (b) + (c) \geq Weight(v)$

$\langle 2 \rangle 1.$	$t[v]$	$=$	$t[f(v_R)]$	by $v = f(v_R)$
		$\neq$	$t[f(v_R - 1)]$	by $\mathcal{A}(v_R) \wedge v_R = First(f(v_R)) \neq 0$
		$=$	$t[f(First(v) - 1)]$	by $v_R = First(v)$
		$=$	$t[W(v)]$	by def. of W

$\langle 2 \rangle 2.\ (c) = Weight(v)$ by $\langle 1 \rangle 2$ and $\langle 2 \rangle 1$
$\langle 2 \rangle 3.$ Q.E.D. by $\langle 2 \rangle 2$ and $(b) \geq 0$
$\langle 1 \rangle 8.\ (c)' = 0$

$\langle 2 \rangle 1.$	$t[v]'$	$=$	$t[f(v_R)]'$	by $v = f(v_R)$
		$=$	$t[f(v_R - 1)]$	by $\mathcal{A}(v_R) \wedge v_R = First(f(v_R)) \neq 0$
		$=$	$t[f(v_R - 1)]'$	by $\mathcal{A}(v_R)$ and $\langle 1 \rangle 3$
		$=$	$t[f(First(v) - 1)]'$	by $v_R = First(v)$
		$=$	$t[W(v)]'$	by def. of W

$\langle 2 \rangle 2.$ Q.E.D. by $\langle 2 \rangle 1$

$\langle 1 \rangle 9.$	$(b)'$			
	$\leq$	$\left(\sum u : u \in V \wedge vTu :: Weight(u)\right)$	by $Weight(u) \geq 0$	
	$=$	$Weight(v) - 1$	def. of $Weight(v)$	
	$<$	$Weight(v)$		

$\langle 1 \rangle 10.\ (b)' + (c)' < (b) + (c)$ by $\langle 1 \rangle 8$, $\langle 1 \rangle 9$ and $\langle 1 \rangle 7$

$\langle 1 \rangle 11.$	$SWeight'$	$=$	$(a)' + (b)' + (c)'$	by $\langle 1 \rangle 5$
		$=$	$(a) + (b)' + (c)'$	by $\langle 1 \rangle 6$
		$<$	$(a) + (b) + (c)$	by $\langle 1 \rangle 10$
		$=$	$SWeight$	by $\langle 1 \rangle 4$

$\langle 1 \rangle 12.$ Q.E.D. by $\langle 1 \rangle 11$

■

Lemma B.29
$\vdash_{\mathcal{A}} \left(\forall v_R : v_R \in V_R \setminus \{0\} :: \mathcal{A}(v_R) \wedge v_R > First(f(v_R)) \Rightarrow -SDistance' < -SDistance\right)$

Proof:

LET: $v_R \in V_R \setminus \{0\}$ arbitrary s.t. $\mathcal{A}(v_R) \wedge v_R > First(f(v_R))$ and let $v = f(v_R)$

$\langle 1 \rangle 1.$	$Distance(v)$			
	$=$	$\left(\#u_R : u_R \in Sim(v) :: u_R < d[v]\right)$		
	$<$	$\left(\#u_R : u_R \in Sim(v) :: u_R \leq d[v]\right)$	by $d[v] \in Sim(v)$ $(Domain)$	
	$\leq$	$\left(\#u_R : u_R \in Sim(v) :: u_R < v_R\right)$	by $d[v] = d[f(v_R)] < v_R$	
			by $\mathcal{A}(v_R) \wedge v_R > First(f(v_R))$	
	$=$	$\left(\#u_R : u_R \in Sim(v) :: u_R < d[v]'\right)$	by $d[v]' = v_R$ by $\mathcal{A}(v_R) \wedge v_R > First(f(v_R))$	
	$=$	$Distance(v)'$		

$\langle 1\rangle 2.\ \Big(\forall u : u \in V \wedge u \neq v :: Distance(u)' = Distance\Big)$
by $\mathcal{A}(v_R) \wedge f(v_R) = v \neq u \Rightarrow d[u]' = d[u]$
$\langle 1\rangle 3.$ Q.E.D. by $\langle 1\rangle 1$ and $\langle 1\rangle 2$

■

Lemma B.30
$$\vdash_{\mathcal{A}} \left(\begin{array}{l}\forall v_R : v_R \in V_R \wedge v_R = First(f(v_R)) :: \\ \quad \mathcal{A}(v_R) \Rightarrow SDistance' \geq SDistance - |Sim(f(v_R))| + 1\end{array}\right)$$

Proof:

LET: $v_R \in V_R$ arbitrary s.t. $\mathcal{A}(v_R) \wedge v_R = First(f(v_R))$ and let $v = f(v_R)$

$\langle 1\rangle 1.\ d[v]' = v_R$ by (F3) and $\mathcal{A}(v_R)$
$= First(f(v_R))$

$\langle 1\rangle 2.\ Distance(v)'$
$= \Big(\#v_R : v_R \in Sim(v) :: v_R < d[v]'\Big)$
$= 0$ by $\langle 1\rangle 1$
$\geq Distance(v) - |Sim(v)| + 1$ by lemma B.22

$\langle 1\rangle 3.\ \Big(\forall u : u \in V \wedge u \neq v :: Distance(u)' = Distance(u)\Big)$
by $\mathcal{A}(v_R) \wedge f(v_R) = v \neq u \Rightarrow d[u]' = d[u]$

$\langle 1\rangle 4.$ Q.E.D. by $\langle 1\rangle 2$ and $\langle 1\rangle 3$

■

Lemma B.31 $\vdash_{\mathcal{A}} n - n_R \leq Chaos \leq M \cdot \frac{n(n-1)}{2}$

Proof: By lemma B.23 and lemma B.26 and $Chaos = M \cdot SWeight - SDistance$ ■

Lemma B.32 $\vdash_{\mathcal{A}} \Big(\forall v_R : v_R \in V_R \setminus \{0\} :: \mathcal{A}(v_R) \Rightarrow Chaos' < Chaos\Big)$

Proof:

LET: $v_R \in V_R \setminus \{0\}$ arbitrary s.t. $\mathcal{A}(v_R)$
PROVE: $Chaos' < Chaos$ by case analysis on $v_R = First(f(v_R))$ and $v_R > First(f(v_R))$
CASE: $v_R = First(f(v_R))$

$\langle 2\rangle 1.\ Chaos'$
$= M \cdot SWeight' - SDistance'$
$\leq M \cdot (SWeight - 1) - SDistance'$ by lemma B.28
$\leq M \cdot SWeight - M - SDistance + |Sim(f(v_R))| - 1$ by lemma B.30
$< M \cdot SWeight - SDistance$
by $M \geq |Sim(f(v_R))|$ for any v
$= Chaos$

CASE: $v_R > First(f(v_R))$

$$
\begin{array}{lll}
& Chaos' & \\
= & M \cdot SWeight' - SDistance' & \\
= & M \cdot SWeight - SDistance' & \text{by lemma B.27} \\
< & M \cdot SWeight - SDistance & \text{by lemma B.29} \\
= & Chaos &
\end{array}
$$

■

Lemma B.33 $\vdash_{\mathcal{A}} \mathcal{A}(0) \Rightarrow Chaos' \leq Chaos + M \cdot n - 1$

Proof:

ASSUME: $\mathcal{A}(0)$

PROVE: $Chaos' \leq Chaos + M \cdot n - 1$

LET: $(a) \triangleq \left(\sum v : v \in V \wedge v \neq f(0) \wedge f(0)Tv :: \begin{cases} Weight(v) & \text{if } W(v)Tv \wedge t[v] \neq t[W(v)] \\ 0 & \text{otherwise} \end{cases}\right)$

$(b) \triangleq \left(\sum v : v \in V \wedge v \neq f(0) \wedge \neg f(0)Tv :: \begin{cases} Weight(v) & \text{if } W(v)Tv \wedge t[v] \neq t[W(v)] \\ 0 & \text{otherwise} \end{cases}\right)$

⟨1⟩1. $SWeight = (a) + (b)$ and $SWeight' = (a)' + (b)'$

The elements summed in (a) and (b) exactly cover those summed in $SWeight$ and no element covered by (a) is covered by (b) (and vice versa).

⟨1⟩2. $(b)' = (b)$

LET: $v \in V$ arbitrary s.t. $v \neq f(0) \wedge \neg f(0)Tv$

⟨2⟩1. $t[v]' = t[v]$ by $\mathcal{A}(0)$ and $v \neq f(0)$

⟨2⟩2. $t[W(v)]' = t[W(v)]$ by $\mathcal{A}(0)$ and $v \neq f(0) \wedge \neg f(0)Tv \Rightarrow W(v) \neq f(0)$

⟨2⟩3. Q.E.D. by ⟨2⟩1 and ⟨2⟩2

⟨1⟩3. $(a) \geq 0$ by $Weight(v) \geq 0$ for any v

⟨1⟩4.
$$
\begin{array}{lll}
(a)' & \leq & \left(\sum v : v \in V \wedge v \neq f(0) \wedge f(0)Tv :: Weight(v)\right) \\
& = & Weight(f(0)) - 1 \quad \text{by } f(0)Tv \Rightarrow v \neq f(0) \\
& = & n - 1 \quad \text{by lemma B.24 and lemma B.25}
\end{array}
$$

⟨1⟩5.
$$
\begin{array}{lll}
SWeight' & = & (a)' + (b)' \quad \text{by ⟨1⟩1} \\
& \leq & (a) + (b) + n - 1 \quad \text{by ⟨1⟩2, ⟨1⟩3 and ⟨1⟩4} \\
& = & SWeight + n - 1 \quad \text{by ⟨1⟩1}
\end{array}
$$

⟨1⟩6.
$$
\begin{array}{lll}
-SDistance' & \leq & -SDistance + |Sim(f(0))| - 1 \quad \text{by lemma B.30, } \mathcal{A}(0) \\
& & \quad \text{and } 0 = First(f(0)) \\
& \leq & -SDistance + M - 1 \quad \text{by } |Sim(v)| \leq M \text{ for any } v
\end{array}
$$

⟨1⟩7.
$$
\begin{array}{lll}
Chaos' & = & M \cdot SWeight' - SDistance' \\
& \leq & M(SWeight + n - 1) - SDistance + M - 1 \quad \text{by ⟨1⟩5 and ⟨1⟩6} \\
& = & Chaos + M \cdot n - 1
\end{array}
$$

■

Lemma B.34 $\vdash_{\mathcal{A}} Chaos = n - n_R \Rightarrow \left(\forall v_R : v_R \in V_R \setminus \{0\} :: \neg * \mathcal{A}(v_R)\right)$

Proof:

ASSUME: $Chaos = n - n_R$
LET: $v_R \in V_R \setminus \{0\}$ arbitrary
PROVE: $\neg * \mathcal{A}(v_R)$ by case analysis on $v_R = First(f(v_R))$ and $v_R > First(f(v_R))$
$\langle 1 \rangle 1$. $SDistance = n_R - n \wedge SWeight = 0$ by $Chaos = n - n_R$ and lemmas B.23 and B.26
$\langle 1 \rangle 2$. $\Big(\forall v_R : v_R \in V_R :: d[f(v_R)] \geq v_R\Big)$
  LET: $v \in V$ arbitrary
  PROVE: $d[v] = Last(v)$ then the statement follows since $Last(v) = \max Sim(v)$
  $\langle 2 \rangle 1$. $Distance(v) = |Sim(v)| - 1$ by $\langle 1 \rangle 1$ and $Distance(v) \leq |Sim(v)| - 1$ for any $v \in V$ (follows from $Domain$)
  $\langle 2 \rangle 2$. Q.E.D. by $\langle 2 \rangle 1$ and $Domain$
$\langle 1 \rangle 3$. $\Big(\forall v : v \in V :: v \neq f(0) \Rightarrow t[v] = t[W(v)]\Big)$
  LET: $v \in V$ arbitrary
  $\langle 2 \rangle 1$. $\neg W(v) T v \vee t[v] = t[W(v)]$ by $\langle 1 \rangle 1$ and $Weight(v) > 0$ for any $v \in V$
  $\langle 2 \rangle 2$. $\neg W(v) T v \Leftrightarrow v = f(0)$ by def. of uTv
  $\langle 2 \rangle 3$. Q.E.D. by $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$
CASE: $v_R = First(f(v_R))$
  $\langle 2 \rangle 1$. $f(v_R) \neq f(0)$
    for suppose $f(v_R) = f(0)$ then $v_R = First(f(v_R)) = First(f(0)) = 0$ contradictory with $v_R \neq 0$
  $\langle 2 \rangle 2$.
    $t[f(v_R)]$
    $= t[W(f(v_R))]$ by $\langle 2 \rangle 1$ and $\langle 1 \rangle 3$
    $= t[f(First(f(v_R)) - 1 \bmod n_R)]$ by def. of W
    $= t[f(v_R - 1 \bmod n_R)]$ by $v_R = First(f(v_R))$
    $= t[f(v_R - 1)]$ by $v_R \in V_R \setminus \{0\}$
  $\langle 2 \rangle 3$. Q.E.D. by $\langle 2 \rangle 2$ and $v_R = First(f(v_R))$
CASE: $v_R > First(f(v_R))$
  $\langle 2 \rangle 1$. $d[f(v_R)] \geq v_R$ by $\langle 1 \rangle 2$
  $\langle 2 \rangle 2$. Q.E.D. by $\langle 2 \rangle 1$ and $v_R > First(f(v_R))$

■

Lemma B.35 $\vdash_{\mathcal{A}} Legal(n_R - 1) \Rightarrow Chaos = n - n_R$

Proof:

ASSUME: $Legal(n_R - 1)$ so $\Big(\forall v_R : v_R \in V_R :: Green(v_R)\Big)$
PROVE: $Chaos = n - n_R$
$\langle 1 \rangle 1$. $\Big(\forall v : v \in V :: t[v] = t[f(0)]\Big)$
  LET: $v \in V$ arbitrary
  $\langle 2 \rangle 1$.
    $t[v]$
    $= t[f(First(v))]$ by $v = f(First(v))$
    $= t[f(0)]$ by $Green(First(v))$
$\langle 1 \rangle 2$. $\Big(\forall v_R : v_R \in V_R :: d[f(v_R)] \geq v_R\Big)$ by $\Big(\forall v_R : v_R \in V_R :: Green(v_R)\Big)$
$\langle 1 \rangle 3$. $SWeight = 0$
  Let $v \in V$ arbitrary then by $\langle 1 \rangle 1$ $t[v] = t[f(0)] = t[W(v)]$. Since $v \in V$ arbitrary $SWeight = 0$

$\langle 1\rangle 4$. $SDistance = n_R - n$
LET: $v \in V$ arbitrary
PROVE: $d[v] = Last(v)$ so $Distance(v) = |Sim(v)| - 1$ and since $v \in V$ arbitrary $SDistance = n_R - n$
$\langle 2\rangle 1$. $d[v] = d[f(Last(v))]$ by $v = f(Last(v))$
$\geq Last(v)$ by $\langle 1\rangle 2$
$\langle 2\rangle 2$. $d[v] \leq Last(v)$ by *Domain*
$\langle 2\rangle 3$. Q.E.D. by $\langle 2\rangle 1$ and $\langle 2\rangle 2$
$\langle 1\rangle 5$. Q.E.D. by $\langle 1\rangle 3$, $\langle 1\rangle 4$ and $Chaos = M \cdot SWeight - SDistance$

■

Corollary B.36 $\vdash_{\mathcal{A}}$ $Legal(n_R - 1)$
$\Leftrightarrow$ $Chaos = n - n_R$
$\Leftrightarrow$ $\Big(\forall v_R : v_R \in V_R \setminus \{0\} :: \neg * \mathcal{A}(v_R)\Big)$

Proof: Follows from lemmas B.35, B.34 and B.20. ■

B.5 First stabilization phase
In this subsection we will define the function *Phase1* that will decrease with the execution of any occurrence. If it has reached its lower bound we've entered the second phase.

$$c(s) \triangleq \{v \in V \setminus \{f(0)\} | t[v] = s\}$$
$$Missing(s) \triangleq c(s) = \emptyset \wedge (t[f(0)] = s \Rightarrow d[f(0)] = 0)$$
$$Miss \triangleq \Big(\min s : s \in \{0, \ldots, Q-1\} \wedge Missing(s) :: (s - t[f(0)]) \bmod Q\Big)$$

$$Phase1 \triangleq M \cdot n \cdot Miss + Chaos$$

Lemma B.37 *Miss is well defined if $Q > n$, i.e.*
$\vdash_{\mathcal{A}} Q > n \Rightarrow \{(s - t[f(0)]) \bmod Q | s \in \{0, \ldots, Q-1\} \wedge Missing(s)\} \neq \emptyset$

Proof:

ASSUME: $\{(s - t[f(0)]) \bmod Q | s \in \{0, \ldots, Q-1\} \wedge Missing(s)\} = \emptyset$
PROVE: $Q \leq n$
LET: $cc(s) \triangleq \{v \in V | t[v] = s\}$
$\langle 1\rangle 1$. $\Big(\forall s : s \in \{0, \ldots, Q-1\} :: cc(s) \neq \emptyset\Big)$
LET: $s \in \{0, \ldots, Q-1\}$ arbitrary
PROVE: $cc(s) \neq \emptyset$
$\langle 2\rangle 1$. $c(s) \neq \emptyset \vee t[f(0)] = s$
by the assumption we have $\neg Missing(s)$
$\langle 2\rangle 2$. Q.E.D.
$c(s) \subseteq cc(s)$, so in case $c(s) \neq \emptyset$ also $cc(s) \neq \emptyset$. In case $t[f(0)] = s$ then $f(0) \in cc(s)$, so also $cc(s) \neq \emptyset$. Therefore it follows from $\langle 2\rangle 1$.
$\langle 1\rangle 2$. $\Big(\forall s_1, s_2 : s_1, s_2 \in \{0, \ldots, Q\} :: s_1 \neq s_2 \Rightarrow cc(s_1) \cap cc(s_2) = \emptyset\Big)$
LET: $s_1, s_2 \in \{0, \ldots, Q-1\}$ arbitrary

ASSUME: $cc(s_1) \cap cc(s_2) \neq \emptyset$ so there is a $v \in V$ s.t. $v \in c(s_1)$ and $v \in c(s_2)$
PROVE: $s_1 = s_2$
$\langle 2 \rangle 1$. $t[v] = s_1$ by $v \in cc(s_1)$
$\langle 2 \rangle 2$. $t[v] = s_2$ by $v \in cc(s_2)$
$\langle 2 \rangle 3$. Q.E.D. by $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$

$$\begin{array}{lllll} \langle 1 \rangle 3. & Q & \leq & |\left(\bigcup s : s \in \{0, \ldots, Q-1\} :: cc(s)\right)| & \text{by } \langle 1 \rangle 1 \text{ and } \langle 1 \rangle 2 \\ & & \leq & |V| & \text{by } \left(\bigcup s : s \in \{0, \ldots, Q-1\} :: cc(s)\right) \subseteq V \\ & & = & n & \end{array}$$

■

Lemma B.38 $\vdash_{\mathcal{A}} 0 \leq Miss < Q$

Proof: $Miss = \left(\min\, s : s \in \{0, \ldots, Q-1\} \wedge Missing(s) :: (s - t[f(0)]) \bmod Q\right)$ so since $Miss$ is well defined by lemma B.37 we conclude $0 \leq Miss < Q$. ■

Lemma B.39 $\vdash_{\mathcal{A}} \left(\forall v_R : v_R \in V_R \setminus \{0\} :: \mathcal{A}(v_R) \wedge Miss > 0 \Rightarrow Miss' \leq Miss\right)$

Proof:

LET: $v_R \in V_R \setminus \{0\}$ arbitrary
ASSUME: $\mathcal{A}(v_R) \wedge Miss > 0$
PROVE: $Miss' \leq Miss$
$\langle 1 \rangle 1$. $\left(\forall s : s \in \{0, \ldots, Q-1\} :: Missing(s) \Rightarrow Missing(s)'\right)$
  LET: $s \in \{0, \ldots, Q-1\}$ arbitrary s.t. $Missing(s)$
  PROVE: $Missing(s)'$
  $\langle 2 \rangle 1$. $t[f(v_R)]' \neq s$ by case analysis on $f(v_R - 1) = f(0)$ and $f(v_R - 1) \neq f(0)$
  CASE: $f(v_R - 1) = f(0)$
    $\langle 3 \rangle 1$. $t[f(0)] \neq s$
      if $t[f(0)] = s$ then $Missing(t[f(0)])$ so $Miss = 0$ contradicting $Miss > 0$
    $\langle 3 \rangle 2$. $t[f(v_R)]' = t[f(v_R - 1)]$ by (F4), $\mathcal{A}(v_R)$ and $v_R > 0$
      $= t[f(0)]$ by $f(v_R - 1) = f(0)$
      $\neq s$ by $\langle 3 \rangle 1$
  CASE: $f(v_R - 1) \neq f(0)$
    $\langle 3 \rangle 1$. $t[f(v_R)]' = t[f(v_R - 1)]$ by (F4), $\mathcal{A}(v_R)$ and $v_R > 0$
      $\neq s$ by $Missing(s)$ and $f(v_R - 1) \neq f(0)$
  $\langle 2 \rangle 2$. $\left(\forall u : u \in V \setminus \{f(0)\} \wedge u \neq f(v_R) :: t[u] \neq s\right)$
    LET: $u \in V \setminus \{f(0)\}$ arbitrary s.t. $u \neq f(v_R)$
    $\langle 3 \rangle 1$. $t[u]' = t[u]$ by $\mathcal{A}(v_R)$ and $u \neq f(v_R)$
      $\neq s$ by $Missing(s)$ and $u \neq f(0)$
  $\langle 2 \rangle 3$. $\left(\forall v : v \in V \setminus \{f(0)\} :: t[v]' \neq s\right)$ so $c(s)' = \emptyset$ by $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$
  $\langle 2 \rangle 4$. $t[f(0)]' = s \Rightarrow d[f(0)]' = 0$
    by case analysis on $f(v_R) \neq f(0)$, $f(v_R) = f(0) \wedge f(v_R - 1) = f(0)$ and $f(v_R) = f(0) \wedge f(v_R - 1) \neq f(0)$

CASE: $f(v_R) \neq f(0)$
$\langle 3 \rangle 1$. *Unchanged* $(t[f(0)], d[f(0)]$ by $\mathcal{A}(v_R)$ and $f(v_R) \neq f(0)$
$\langle 3 \rangle 2$. Q.E.D. by *Missing*(s) and $\langle 3 \rangle 1$
CASE: $f(v_R) = f(0) \wedge f(v_R - 1) = f(0)$
ASSUME: $t[f(0)]' = s$
PROVE: $\bot$
$\langle 3 \rangle 1$. $t[f(0)]$
$= t[f(v_R - 1)]$ by $f(0) = f(v_R - 1)$
$= t[f(v_R)]'$ by (F4) and $\mathcal{A}(v_R)$ and $v_R \neq 0$
$= t[f(0)]'$ by $f(0) = f(v_R)$
$= s$ by assumption
$\langle 3 \rangle 2$. *Missing*$(t[f(0)])$ by *Missing*(s) and $\langle 3 \rangle 1$
$\langle 3 \rangle 3$. Q.E.D. *Miss* $= 0$ by $\langle 3 \rangle 2$, contradicting *Miss* > 0
CASE: $f(v_R) = f(0) \wedge f(v_R - 1) \neq f(0)$
ASSUME: $t[f(0)]' = s$
PROVE: $\bot$
$\langle 3 \rangle 1$. $t[f(v_R - 1)]$
$= t[f(v_R)]'$ by (F4), $\mathcal{A}(v_R)$ and $v_R \neq 0$
$= t[f(0)]'$ by $f(v_R) = f(0)$
$= s$ by assumption
$\langle 3 \rangle 2$. $c(s) \neq \emptyset$ by $\langle 3 \rangle 1$ and $f(v_R - 1) \neq f(0)$
$\langle 3 \rangle 3$. Q.E.D. by $\langle 3 \rangle 2$ and *Missing*(s)
$\langle 2 \rangle 5$. Q.E.D. by $\langle 2 \rangle 3$ and $\langle 2 \rangle 4$
$\langle 1 \rangle 2$. $t[f(0)]' = t[f(0)]$ by case analysis on $f(v_R) = f(0)$ and $f(v_R) \neq f(0)$
CASE: $f(v_R) = f(0)$ so $v_R >$ *First*$(f(v_R))$ by $v_R > 0 =$ *First*$(f(0))$
$\langle 2 \rangle 1$. $t[f(0)]' = t[f(v_R)]'$ by $f(0) = f(v_R)$
$= t[f(v_R)]$ by $\mathcal{A}(v_R)$ and $v_R >$ *First*$(f(v_R))$
$= t[f(0)]$ by $f(0) = f(v_R)$
CASE: $f(v_R) \neq f(0)$
$\langle 2 \rangle 1$. $t[f(0)]' = t[f(0)]$ by $\mathcal{A}(v_R)$ and $f(v_R) \neq f(0)$
$\langle 1 \rangle 3$. $\{(s - t[f(0)]) \bmod Q | s \in \{0, \ldots, Q-1\} \wedge \mathit{Missing}(s)\}$
$\subseteq \{(s - t[f(0)]) \bmod Q | s \in \{0, \ldots, Q-1\} \wedge \mathit{Missing}(s)'\}$ by $\langle 1 \rangle 1$
$\subseteq \{(s - t[f(0)]') \bmod Q | s \in \{0, \ldots, Q-1\} \wedge \mathit{Missing}(s)'\}$ by $\langle 1 \rangle 2$
$\langle 1 \rangle 4$. Q.E.D. by $\langle 1 \rangle 3$

■

Lemma B.40 $\vdash_{\mathcal{A}} \mathcal{A}(0) \wedge \mathit{Miss} > 0 \Rightarrow \mathit{Miss}' < \mathit{Miss}$

Proof:

ASSUME: $\mathcal{A}(0) \wedge \mathit{Miss} > 0$
$\langle 1 \rangle 1$. $\Big(\forall s : s \in \{0, \ldots, Q-1\} :: \mathit{Missing}(s) \Rightarrow \mathit{Missing}(s)'\Big)$
LET: $s \in \{0, \ldots, Q-1\}$ arbitrary s.t. *Missing*(s)
PROVE: *Missing*$(s)'$
$\langle 2 \rangle 1$. $c(s)' = c(s)$ by $\mathcal{A}(0)$ so $\Big(\forall u : u \in V \setminus \{f(0)\} :: \mathit{Unchanged}\ (t[u])\Big)$

$\langle 2 \rangle 2.\ d[f(0)]' = 0$ by (F3) and $\mathcal{A}(0)$
$\langle 2 \rangle 3.$ Q.E.D. by $\langle 2 \rangle 1$ and $\langle 2 \rangle 2$

$\langle 1 \rangle 2.$ Miss'
$= \ \Big(\min s : s \in \{0, \ldots, Q-1\} \wedge \mathit{Missing}(s)' :: (s - t[f(0)])' \bmod Q\Big)$
$\leq \ \Big(\min s : s \in \{0, \ldots, Q-1\} \wedge \mathit{Missing}(s) :: (s - t[f(0)])' \bmod Q\Big)$ by $\langle 1 \rangle 1$
$= \ \Big(\min s : s \in \{0, \ldots, Q-1\} \wedge \mathit{Missing}(s) :: (s - t[f(0)] - 1) \bmod Q\Big)$ by $\mathcal{A}(0)$
$= \ \mathit{Miss} - 1$ by $\mathit{Miss} > 0$ and $Q \geq 2$
$< \ \mathit{Miss}$

■

Lemma B.41 $\vdash_{\mathcal{A}} n - n_R \leq \mathit{Phase1} \leq M \cdot n \cdot (Q-1) + M\frac{n(n-1)}{2}$

Proof:

$\langle 1 \rangle 1.$ $\mathit{Phase1}$
$= \ M \cdot n \cdot \mathit{Miss} + \mathit{Chaos}$
$\geq \ \mathit{Chaos}$ by lemma B.38
$\geq \ n - n_R$ by lemma B.31

$\langle 1 \rangle 2.$ $\mathit{Phase1}$
$= \ M \cdot n \cdot \mathit{Miss} + \mathit{Chaos}$
$\leq \ M \cdot n \cdot (Q-1) + \mathit{Chaos}$ by lemma B.38
$\leq \ M \cdot n \cdot (Q-1) + M\frac{n(n-1)}{2}$ by lemma B.31

■

Lemma B.42 $\vdash_{\mathcal{A}} \mathcal{T} \wedge \mathit{Miss} > 0 \Rightarrow \mathit{Phase1}' < \mathit{Phase1}$

Proof:

ASSUME: $\mathcal{T} \wedge \mathit{Miss} > 0$
LET: $v_R \in V_R$ s.t. $\mathcal{A}(v_R)$ such v_R exists by $\mathcal{T}$
PROVE: $\mathit{Phase1}' < \mathit{Phase1}$ by case analysis on $v_R = 0$ and $v_R \neq 0$
CASE: $v_R = 0$ so $\mathcal{A}(0)$
$\langle 2 \rangle 1.$ $\mathit{Phase1}'$
$= \ M \cdot n \cdot \mathit{Miss}' + \mathit{Chaos} + M \cdot n - 1$
$\leq \ M \cdot n \cdot (\mathit{Miss} - 1) + \mathit{Chaos} + M \cdot n - 1$ by lemma B.33 and $\mathcal{A}(0)$
$= \ M \cdot n \cdot \mathit{Miss} + \mathit{Chaos} - 1$ by lemma B.40 and $\mathcal{A}(0) \wedge \mathit{Miss} > 0$
$= \ \mathit{Phase1} - 1$
$< \ \mathit{Phase1}$
CASE: $v_R \neq 0$
$\langle 2 \rangle 1.$ $\mathit{Phase1}'$
$= \ M \cdot n \cdot \mathit{Miss}' + \mathit{Chaos}'$
$\leq \ M \cdot n \cdot \mathit{Miss} + \mathit{Chaos}'$ by lemma B.39, $\mathcal{A}(v_R)$ and $v_R \neq 0$
$< \ M \cdot n \cdot \mathit{Miss} + \mathit{Chaos}$ by lemma B.32, $\mathcal{A}(v_R)$ and $v_R \neq 0$
$= \ \mathit{Phase1}$

■

B.6 Second stabilization phase

In this section we define the function *Stabilized* that will be greater or equal to zero during the second stabilization phase. As long as it is greater or equal to zero and smaller than $n_R - 1$ it will not decrease and the root will also not execute. Execution of non-root nodes will decrease the function *Chaos* as already proven. Since the range of *Chaos* is finite, eventually *Stabilized* must equal $n_R - 1$ and this implies $Legal(n_R - 1)$, i.e. we've reached a legitimate configuration.

$$Stable(v_R) \triangleq \Big(\forall u_R : u_R \in V_R :: u_R \leq v_R \Leftrightarrow Green(u_R)\Big)$$

$$Stabilized \triangleq \begin{cases} -1 & \text{if } \neg\Big(\exists u_R : u_R \in V_R :: Stable(u_R)\Big) \\ \Big(\max\, u_R : u_R \in V_R :: Stable(u_R)\Big) & \text{otherwise} \end{cases}$$

Lemma B.43 $\vdash_{\mathcal{A}} Miss = 0 \Rightarrow Stabilized \geq 0$

Proof:

ASSUME: $Miss = 0$ so $Missing(t[f(0)])$ and $c(t[f(0)]) = \emptyset \wedge d[f(0)] = 0$
LET: $u_R \in V_R \setminus \{0\}$ arbitrary
PROVE: $\neg Green(u_R)$ then $Stable(0)$ follows from $u_R \in V_R \setminus \{0\}$ arbitrary and lemma B.5.
Proven by case analysis on $f(u_R) = f(0)$ and $f(u_R) \neq f(0)$
CASE: $f(u_R) = f(0)$

⟨2⟩1.	$d[f(u_R)]$		
	$=$ $d[f(0)]$	by	$f(u_R) = f(0)$
	$=$ 0	by	$Miss = 0$
	$<$ u_R	by	$u_R \in V_R \setminus \{0\}$

⟨2⟩2. Q.E.D. by ⟨2⟩1

CASE: $f(u_R) \neq f(0)$

⟨2⟩1. $t[f(u_R)] \neq t[f(0)]$ by $c(t[f(0)]) \neq \emptyset$ and $f(u_R) \neq f(0)$
⟨2⟩2. Q.E.D. by ⟨2⟩1

■

Lemma B.44 $\vdash_{\mathcal{A}} Stable(n_R - 1) \Leftrightarrow Legal(n_R - 1)$

Proof:

⟨1⟩1.	$Stable(n_R - 1)$	
	$\Leftrightarrow$ $\Big(\forall u_R : u_R \in V_R :: Green(u_R)\Big)$	
	$\Leftrightarrow$ $Legal(n_R - 1)$	

■

Lemma B.45 $\vdash_{\mathcal{A}} -1 \leq Stabilized < n_R$

Proof: By $0 \leq \Big(\max\, u_R : u_R \in V_R :: Stable(u_R)\Big) < n_R$ if $\Big(\exists u_R : u_R \in V_R :: Stable(u_R)\Big)$. ■

Lemma B.46

$$\vdash_{\mathcal{A}} \left(\begin{array}{l}\forall v_R, u_R : v_R, u_R \in V_R \wedge v_R < n_R - 1 :: \\ \qquad \mathit{Stable}(v_R) \wedge \mathcal{A}(u_R) \Rightarrow \mathit{Stable}(v_R)' \vee \mathit{Stable}(v_R + 1)'\end{array}\right)$$

Proof:

LET: $v_R, u_R \in V_R$ arbitrary s.t. $v_R < n_R - 1$
ASSUME: $\mathit{Stable}(v_R) \wedge \mathcal{A}(u_R)$
PROVE: $\mathit{Stable}(v_R)' \vee \mathit{Stable}(v_R + 1)'$
$\langle 1 \rangle 1$. $u_R > 0$
  ASSUME: $u_R = 0$ for the sake of a contradiction
  PROVE: $\bot$
  $\langle 2 \rangle 1$. $\neg \mathit{Green}(n_R - 1)$ by $\mathit{Stable}(v_R) \wedge v_R < n_R - 1$
  $\langle 2 \rangle 2$. $\neg * \mathcal{A}(0)$ by $\langle 2 \rangle 1$ and lemma B.10
  $\langle 2 \rangle 3$. $\neg \mathcal{A}(0)$ by $\langle 2 \rangle 2$ and $\vdash_{\mathcal{A}} \mathcal{A} \Rightarrow * \mathcal{A}$
  $\langle 2 \rangle 4$. $\mathcal{A}(0)$ by $\mathcal{A}(u_R)$ and $u_R = 0$
  $\langle 2 \rangle 5$. Q.E.D. by $\langle 2 \rangle 3$ and $\langle 2 \rangle 4$
$\langle 1 \rangle 2$. $v_R < u_R$
  ASSUME: $u_R \leq v_R$ for the sake of a contradiction
  PROVE: $\bot$
  $\langle 2 \rangle 1$. $0 < u_R \leq v_R$ by $u_R \leq v_R$ and $\langle 1 \rangle 1$
  $\langle 2 \rangle 2$. $\mathit{Green}(u_R - 1) \wedge \mathit{Green}(u_R)$ by $\langle 2 \rangle 1$ and $\mathit{Stable}(v_R)$
  $\langle 2 \rangle 3$. $\neg * \mathcal{A}(u_R)$ by $\langle 1 \rangle 1$, $\langle 2 \rangle 2$ and lemma B.6$[v_R := u_R]$
  $\langle 2 \rangle 4$. $\neg \mathcal{A}(u_R)$ by $\langle 2 \rangle 3$ and $\vdash_{\mathcal{A}} \mathcal{A} \Rightarrow * \mathcal{A}$
  $\langle 2 \rangle 5$. Q.E.D. by $\langle 2 \rangle 4$ and $\mathcal{A}(u_R)$
$\langle 1 \rangle 3$. $\left(\forall w_R : w_R \in V_R \wedge w_R \leq v_R :: \mathit{Green}(w_R)'\right)$
  LET: $w_R \in V_R$ arbitrary s.t. $w_R \leq v_R$
  PROVE: $\mathit{Green}(w_R)'$
  $\langle 2 \rangle 1$. $\mathit{Green}(w_R)$ by $\mathit{Stable}(v_R)$ and $w_R \leq v_R$
  $\langle 2 \rangle 2$. $w_R \leq v_R < u_R$ by $w_R \leq v_R$ and $\langle 1 \rangle 2$
  $\langle 2 \rangle 3$. Q.E.D. by lemma B.14$[u_R, v_R := u_R, w_R]$, $\langle 2 \rangle 2$, $\langle 2 \rangle 1$ and $\mathcal{A}(u_R)$
$\langle 1 \rangle 4$. $\left(\forall w_R : w_R \in V_R \wedge w_R > v_R + 1 :: \neg \mathit{Green}(w_R)'\right)$
  LET: $w_R \in V_R$ arbitrary s.t. $w_R > v_R + 1$
  PROVE: $\neg \mathit{Green}(w_R)'$ by case analysis on $w_R > u_R$ and $w_R \leq u_R$
  $\langle 2 \rangle 1$. $\neg \mathit{Green}(w_R)$ by $\mathit{Stable}(v_R)$ and $w_R > v_R + 1$
  CASE: $w_R > u_R$
    $\langle 3 \rangle 1$. Q.E.D. by lemma B.16$[u_R, v_R := u_R, w_R]$, $\langle 1 \rangle 1$, $u_R < w_R$, $\mathcal{A}(u_R)$ and $\langle 2 \rangle 1$
  CASE: $w_R \leq u_R$
    $\langle 3 \rangle 1$. $u_R - 1 \geq w_R - 1$ by case
    $> v_R$ by let of w_R
    $\langle 3 \rangle 2$. $\neg \mathit{Green}(u_R - 1)$ by $\mathit{Stable}(v_R)$ and $\langle 3 \rangle 1$
    $\langle 3 \rangle 3$. Q.E.D. by lemma B.16$[u_R, v_R := u_R, w_R]$, $\langle 1 \rangle 1$, $\langle 3 \rangle 2$, $\mathcal{A}(u_R)$ and $\langle 2 \rangle 1$
$\langle 1 \rangle 5$. Q.E.D. by $\langle 1 \rangle 3$ and $\langle 1 \rangle 4$.
  Note that we have either $\mathit{Green}(v_R + 1)'$ or $\neg \mathit{Green}(v_R + 1)'$ establishing respectively $\mathit{Stable}(v_R + 1)'$ and $\mathit{Stable}(v_R)'$.

■

Corollary B.47 $\vdash_{\mathcal{A}} 0 \leq \mathit{Stabilized} < n_R - 1 \wedge \mathcal{T} \Rightarrow \mathit{Stabilized} \leq \mathit{Stabilized}'$

Proof:

ASSUME: $0 \leq \mathit{Stabilized} < n_R - 1 \wedge \mathcal{T}$
PROVE: $\mathit{Stabilized} \leq \mathit{Stabilized}'$
LET: $v_R \in V_R$ s.t. $v_R = \mathit{Stabilized} \wedge v_R < n_R - 1$ so also $\mathit{Stable}(v_R)$. Such v_R exists by $0 \leq \mathit{Stabilized} < n_R - 1$.
LET: $u_R \in V_R$ s.t. $\mathcal{A}(u_R)$. Such u_R exists by $\mathcal{T}$
⟨1⟩1. Q.E.D.
By lemma B.46 we have $\mathit{Stable}(v_R)' \vee \mathit{Stable}(v_R+1)'$ and therefore $\mathit{Stabilized}' \geq v_R = \mathit{Stabilized}$

■

B.7 Temporal Glue

In this subsection we will glue all the previous mathematical reasoning together to get the wanted temporal statements. We call a protocol, specified by the formula Φ selfstabilizing with respect to the *legitimate* configurations specified by L if and only if we can show the following:

$$\vdash \Phi \Leftrightarrow \Box\Phi$$

$$\vdash \Phi \wedge L \Rightarrow \Box L$$

$$\vdash \Phi \Rightarrow \Diamond L$$

If we also want to show some bound (*bound*) on the stabilisation time (expressed in configuration changes) we should prove the following instead of the third condition:

$$\vdash \Phi \wedge \mathit{time} = 0 \wedge \Box[\mathit{time}' = \mathit{time} + 1] \Rightarrow \Diamond(L \wedge \mathit{time} \leq \mathit{bound})$$

where *time* is (of course) a variable not occuring in Φ, it is a so called *history variable*. The first condition states that Φ does not specify an initialisation predicate. The second condition states the safety part of the proof and has already been shown in corollary B.19. For the third condition we mainly use the following rule, that can be derived from the Lattice-rule in TLA:

If $\vdash_{\mathcal{A}} \mathcal{N} \wedge P \Rightarrow Q' \vee (f' < f \wedge P')$
and $\vdash_{\mathcal{A}} lb \leq f < ub$
where the range of f is a subset of $\mathbb{Z}$ and $lb, ub \in \mathbb{Z}$
then $\vdash \Box\langle\mathcal{N} \wedge \mathit{time}' = \mathit{time} + 1\rangle \Rightarrow (P \wedge \mathit{time} \leq ts \rightsquigarrow Q \wedge \mathit{time} \leq ts + ub - lb)$

We refer to this rule as the t-Lattice-rule.

Theorem B.48 $\vdash \Phi \wedge \mathit{time} = 0 \wedge \Box[\mathit{time}' = \mathit{time} + 1] \Rightarrow \Diamond(\mathit{Legitimate} \wedge \mathit{time} \leq \mathit{tPhase1} + \mathit{tPhase2})$

where $\mathit{tPhase1} \triangleq M \cdot n \cdot (Q-1) + M\frac{n(n-1)}{2} + n_R - n + 1$
$\mathit{tPhase2} \triangleq M\frac{n(n-1)}{2} + n_R - n + 1$

Proof:

$\langle 1\rangle 1.\ \vdash \Phi \Rightarrow \Box\langle\mathcal{T}\rangle$

$\langle 2\rangle 1.\ \vdash_{\mathcal{A}} \Big(\exists v_R : v_R \in V_R :: *\mathcal{A}(v_R)\Big)$ by corollary B.21

$\langle 2\rangle 2.\ \vdash_{\mathcal{A}} *\mathcal{T}$ by $\langle 2\rangle 1$ and *Domain*

$\langle 2\rangle 3.\ \vdash \Box * \mathcal{T}$ by $\langle 2\rangle 2$

$\langle 2\rangle 4.\ \vdash [\mathcal{T}] \wedge \mathrm{WF}(\mathcal{T}) \wedge *\mathcal{T} \Rightarrow \Box\langle\mathcal{T}\rangle$

This follows from the same rule as used in the proof of theorem B.12:

$$\text{If} \quad \vdash_{\mathcal{A}} P \Rightarrow \Big(\bigwedge i, j : i, j \in I \wedge i \neq j :: *\mathcal{A}(i) \Rightarrow \neg * \mathcal{A}(j)\Big)$$
$$\text{then} \quad \vdash \Big(\bigwedge i : i \in I :: P \wedge [\mathcal{A}] \wedge \mathrm{WF}(\mathcal{A}) \wedge *\mathcal{A}(i) \Rightarrow \langle \mathcal{A}(i)\rangle\Big)$$
$$\text{where} \quad \mathcal{A} \triangleq \Big(\bigvee i : i \in I :: \mathcal{A}(i)\Big)$$

Take $P = \top$ and $I = \{1\}$, then the condition is trivially satisfied. The conclusion follows by taking $\mathcal{A}(1) = \mathcal{T}$.

$\langle 2\rangle 5.\ \Box[\mathcal{T}] \wedge \mathrm{WF}(\mathcal{T}) \wedge \Box * \mathcal{T} \Rightarrow \Box\langle\mathcal{T}\rangle$ by $\Box$-mono on $\langle 2\rangle 4$

$\langle 2\rangle 6.$ Q.E.D. by $\langle 2\rangle 3$, $\langle 2\rangle 5$ and $\Phi \Rightarrow \Box[\mathcal{T}] \wedge \mathrm{WF}(\mathcal{T})$.

$\langle 1\rangle 2.\quad \Phi \wedge \Box[time' = time + 1]$

$\Rightarrow\quad \Box\langle\mathcal{T}\rangle \wedge \Box[time' = time + 1]$ by $\langle 1\rangle 1$

$\Leftrightarrow\quad \Box(\langle\mathcal{T}\rangle \wedge [time' = time + 1])$ by $\Box\phi \wedge \Box\psi \Leftrightarrow \Box(\phi \wedge \psi)$

$\Leftrightarrow\quad \Box(\langle\mathcal{T} \wedge time' = time + 1\rangle)$ by $\langle\mathcal{A}\rangle \wedge [\mathcal{B}] \Leftrightarrow \langle\mathcal{A} \wedge \mathcal{B}\rangle$

$\langle 1\rangle 3.\ \vdash time = 0 \wedge \Box\langle\mathcal{T} \wedge time' = time + 1\rangle \Rightarrow \Diamond(Stabilized \geq 0 \wedge time \leq tPhase1)$

$\langle 2\rangle 1.\quad \mathcal{T} \wedge Miss > 0$

$\Rightarrow\quad Phase1' < Phase1$ by lemma B.42

$\Rightarrow\quad Miss' = 0 \vee (Phase1' < Phase1 \wedge Miss' > 0)$ by lemma B.38

$\langle 2\rangle 2.\ \Box\langle\mathcal{T} \wedge time' = time + 1\rangle \Rightarrow (Miss > 0 \wedge time = 0 \rightsquigarrow Miss = 0 \wedge time \leq tPhase1)$
by the t-Lattice-rule, $\langle 2\rangle 1$ and lemma B.41

$\langle 2\rangle 3.\ time = 0 \Rightarrow (Miss > 0 \wedge time = 0) \vee \Diamond(Miss = 0 \wedge time \leq tPhase1)$
by $tPhase1 \geq 0$ and $\vdash \phi \Rightarrow \Diamond\phi$

$\langle 2\rangle 4.\ time = 0 \wedge \Box\langle\mathcal{T} \wedge time' = time + 1\rangle \Rightarrow \Diamond(Miss = 0 \wedge time \leq tPhase1)$
by $\langle 2\rangle 2$ and $\langle 2\rangle 3$

$\langle 2\rangle 5.$ Q.E.D. by $\langle 2\rangle 4$ and lemma B.43

$\langle 1\rangle 4.\ \vdash Stabilized \geq 0 \wedge time \leq tPhase1 \wedge \Box\langle\mathcal{T} \wedge time' = time + 1\rangle \Rightarrow$
$\Diamond(Legitimate \wedge time \leq tPhase1 + tPhase2)$

$\langle 2\rangle 1.\ \mathcal{T} \wedge 0 \leq Stabilized < n_R - 1 \Rightarrow 0 \leq Stabilized' \leq n_R - 1$ by corollary B.47 and lemma B.45

$\langle 2\rangle 2.\ 0 \leq Stabilized < n_R - 1 \Rightarrow \neg * \mathcal{A}(0)$
Let $v_R = Stabilized$ and assume $0 \leq v_R < n_R - 1$ so $Stable(v_R)$ and since $v_R < n_R - 1$ we have $\neg Green(n_R - 1)$ and by lemma B.10 we conclude $\neg * \mathcal{A}(0)$.

$\langle 2\rangle 3.\quad \mathcal{T} \wedge 0 \leq Stabilized < n_R - 1$

$\Rightarrow\quad \Big(\exists v_R : v_R \in V_R \setminus \{0\} :: \mathcal{A}(v_R)\Big) \wedge 0 \leq Stabilized < n_R - 1$ by $\langle 2\rangle 2$

$\Rightarrow\quad 0 \leq Stabilized' \leq n_R - 1 \wedge Chaos' < Chaos$
by $\langle 2\rangle 1$ and lemma B.32

$\Rightarrow\quad Stabilized' = n_R - 1 \vee (0 \leq Stabilized' < n_R - 1 \wedge Chaos' < Chaos)$ by logic

$\langle 2\rangle 4.\ \Box\langle\mathcal{T} \wedge time' = time + 1\rangle \Rightarrow (0 \leq Stabilized < n_R - 1 \wedge time \leq tPhase1 \rightsquigarrow$
$Stabilized = n_R - 1 \wedge time \leq tPhase1 + tPhase2)$

by t-Lattice-rule, ⟨2⟩3 and lemma B.31

⟨2⟩5. $\mathit{Stabilized} \geq 0 \wedge \mathit{time} \leq \mathit{tPhase1}$
$\Rightarrow \vee\ 0 \leq \mathit{Stabilized} < n_R - 1 \wedge \mathit{time} \leq \mathit{tPhase1}$
$\vee\ \Diamond(\mathit{Stabilized} = n_R - 1 \wedge \mathit{time} \leq \mathit{tPhase1} + \mathit{tPhase2})$
by lemma B.45 and $\mathit{tPhase2} \geq 0$.

⟨2⟩6. $\mathit{Stabilized} \geq 0 \wedge \mathit{time} \leq \mathit{tPhase1} \wedge \Box\langle \mathcal{T} \wedge \mathit{time}' = \mathit{time} + 1\rangle \Rightarrow$
$\Diamond(\mathit{Stabilized} = n_R - 1 \wedge \mathit{time} \leq \mathit{tPhase1} + \mathit{tPhase2})$
by ⟨2⟩5 and ⟨2⟩4

⟨2⟩7. Q.E.D. by lemma B.44 and ⟨2⟩6.

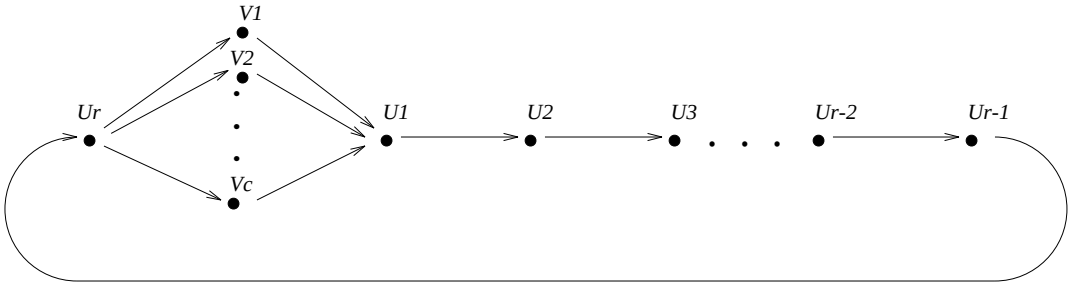
⟨1⟩5. Q.E.D.
by ⟨1⟩1 ... ⟨1⟩4.

■

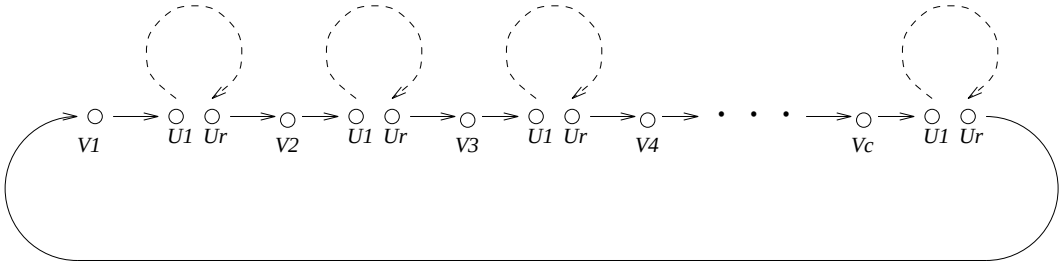
This proves the third and last requirement of a selfstabilizing protocol.

C. Concluding remarks

If the original graph G was already a ring and we take $n_R = n$ and $f(i) = i$, we get $M = 1$, and the stabilization time is $O(n^2)$. This is the same stabilization time Dijkstra gave. Actually our protocol does essentially the same as Dijkstra's protocol. However the bound of $O(n^3)$ can not be smaller in general, due to the following example. Consider the following graph, where both r and c are in $\Theta(n)$.



In a covering ring of this graph, the nodes $U_1, \ldots, U_r$ must be used to connect (the images of) any two nodes V_i and V_j, so $n_R \in \Omega(n^2)$. In the picture below a covering ring is shown. The dashed arcs denote a copy of the chain $U_1, \ldots, U_r$. So each U_i simulates a node in each dashed arc, and the $\mathit{First}(U_i)$ is in the first (leftmost) dashed arc. Lets say that a ticket of w_R is valid if $d[f(w_R)] \geq w_R$. Also call $w_R - 1 \bmod n_R$, the predecessor of w_R. Lets take V_1 as the root node.



The first occurrences (first arc) will always copy a valid ticket from their predecessor in the first arc, if it is unequal to their own ticket. The other occurrences will only increase their

distance in any of the other dashed arcs, if the ticket of their predecessor equals their own and is valid. Now consider the following scenario. First all nodes in the first arc copy the root value (one by one of course). Now instead of travelling through the second arc, this value is present in V_c and valid. So the simulating nodes can increase their distance immediately to the last arc (one by one of course). Now the root can generate a new ticket (since its ticket has returned at $n_R - 1$, and it is again copied by the first arc. Now this ticket has been waiting in V_{c-1}, so the nodes can increase their distance one by one through the last *two* arcs. Again the root generates a new ticket, the first arc copies it, and this ticket was waiting in V_{c-2}. And so on. Finally we end up with a ticket waiting in V_2, only then the protocol has stabilized. We now have had $c - 1 \in \Omega(n)$ separate root tickets. The first (from V_1) travels through the first and last arc. The second (from V_c) travels through the first and last two arcs. In general the i-th travels through the first and last i arcs. So each value travels (on the average) through $\Omega(n)$ arcs. Each arc has $\Omega(n)$ length, so in total we have established a scenario that takes $\Omega(n^3)$ stabilization time.

One might suggest that we should have sticked closer to the colors blue and white, used in Dijkstra's original proof. Dijkstra used a history variable to color its nodes. Initially each node was white. If the root generated a new ticket it became blue and if another node copied a blue ticket it became blue too. First of all one should prove that this indeed is a history variable, which would not be very difficult, but still. Note however that a blue node is also green and that this is actually the property Dijkstra uses of blue when he concludes that when eventually all nodes are blue, the configuration is legitimate. Furthermore our proof more precisely shows what happens to the chaos in the white nodes. One could imagine that there exists a scenario for the white nodes to stay white but still execute once in a while. Due to the decreasing chaos this is not the case. The advantage of blue and white is that there would be no need for the second phase. However the overhead of proof for the second phase is very small, since its proof coincides very much with the proof of staying legitimate once a legitimate configuration has been reached.

In an earlier version of our proof we assumed that the root node had only *one* occurrence. In this case the root only needed to generate a new ticket out of n possible tickets, where as it now needs $n + 1$ possible tickets. In transforming our proof we made a slight mistake in lemma B.43 and had to correct this by changing the definition of $Missing(s)$ (to include the part $t[f(0)] = s \Rightarrow d[f(0)] = 0$).