The Naiad Clock Protocol: Specification, Model Checking, and Correctness Proof

Thomas L. Rodeheffer
Microsoft Research, Silicon Valley

February 12, 2013

Abstract

This report presents a formal specification, written in TLA+, for the Naiad Clock protocol, along with the results of checking the specification using the TLC model checker. Also presented is a formal proof of the Naiad Clock safety properties, which has been mechanically checked using the TLA+ proof system.

This report is based partly on work with Martín Abadi, Frank McSherry, and Derek Murray.

Contents

Chapter 1

The Naiad Clock Protocol

We describe the Naiad Clock protocol informally, followed by a basic specification. We assume that the reader is familiar with TLA+ [5].

1.1 Informal description

The Naiad Clock Protocol oversees the progress of a computation running within Naiad [7, 8], a distributed dataflow system in which records flow through an abstract dataflow computation graph.

In any state of a Naiad computation, the existing records can occupy different stages in the logical progress of the computation. For example, a simple computation may consist of several successive stages in which an input record at each stage is transformed into an input record at the following stage. Each stage represents a point in virtual time. In this case, the points would be arranged in a linear order.

In general, we assume a set of points in virtual time, with a partial order, and associate each record with a point in virtual time, but the set of points need not be finite, and the partial order need not be linear. An operation can consume input records from a set of points and produce output records at another set of points.

We do require that, if an operation produces a record at one point in virtual time, then the operation has consumed at least one record at a strictly lower point according to the partial order. Therefore, as the computation proceeds, the population of records will migrate away from lower points. Should a downward-closed set of points become vacant, this set will always thereafter remain vacant, as any operation that might produce a record associated with a point in the set would need to consume such a record as well. This monotonically increasing set of permanently vacant points represents the progress of the Naiad computation.

It is important that a Naiad processor become aware that a set of points is permanently vacant, because some of the Naiad operations perform an aggreation of all records arriving at a given point. The aggregation (along with any temporary storage it might need) is not complete until all records have been seen.

Since a Naiad computation runs on a distributed collection of processors, each processor is not able to observe, directly, the exact contents of the set of records in order to measure progress. Processors must instead communicate with each other, as they perform operations, exchanging information about the records that those operations consume and produce. With this information, each processor can maintain a possibly delayed but always safe approximation to the set of permanently vacant points in virtual time.

More concretely, in the Naiad Clock Protocol, each processor maintains a local occupancy vector that maps each point to the processor's view of the number of records at that point, depicted in Figure 1.1. At the start of the Naiad computation, this local vector is initialized from the initial set of records in the system. A processor tracks changes in occupancy due to the operations that it performs. When convenient, the processor broadcasts incremental updates to all processors, sending updates about points with net poduction of records before those about points with net consumption of records. When a processor receives one of these updates, it adjusts its local occupancy vector accordingly. The protocol assumes

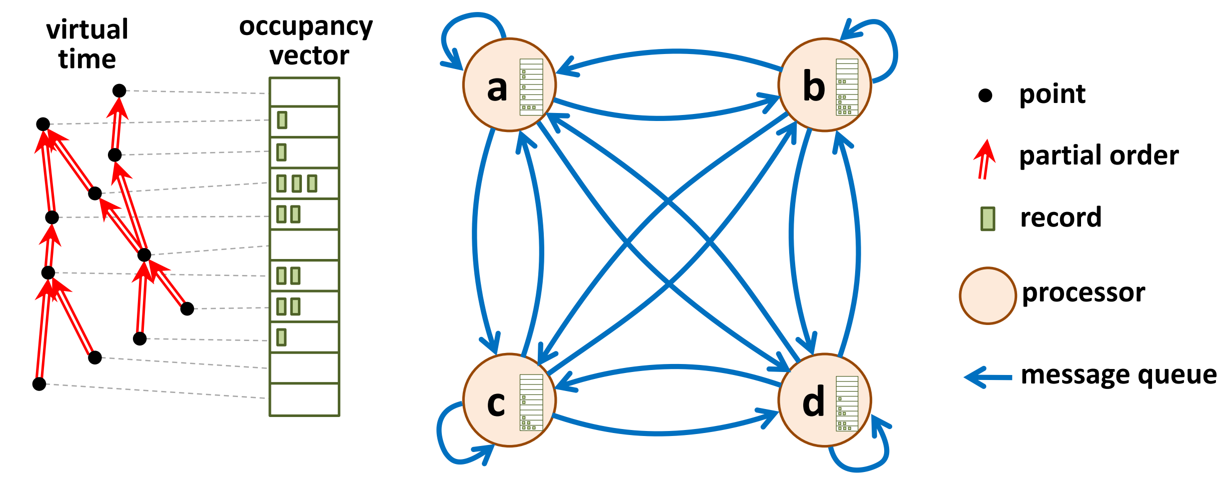

Figure 1.1: Overall structure: each processor locally accumulates a delayed view of the occupancy vector.

that communication channels between processors are reliable and completely ordered, so that updates are neither dropped nor delivered out of order.

The intent of this approach is that, once a downward-closed set of points becomes vacant in the local occupancy vector of some processor, that same set of points is in fact vacant thereafter in the global set of records. Although the local occupancy vector can be a delayed view of the true occupancy vector, it is a safe approximation, so it allows each processor to report correct results from completed parts of the computation to external observers; it is also a useful input to each processor's memory management and scheduling decisions.

1.2 Basic specification

To progress to a more formal description of the clock protocol, we introduce the definitions shown in Figure 1.2. $Point$ is the set of points, $Proc$ is the set of processors, and $\preceq$ is a partial order on $Point$.

A count vector maps each point to a natural number, the count of the number of records at that point. A delta vector maps each point to an integer, representing a change in the record count at that point.

We use Z to designate the delta vector that is everywhere zero and $\oplus$ and $\ominus$ to indicate component-wise addition and subtraction.

Given a delta vector a, we say that point t is positive iff $a[t] > 0$ and negative iff $a[t] < 0$. Since talking about the locations of positive and negative points in delta vectors turns out to be important in the clock protocol and in its proof of correctness, we define several predicates for this purpose. A delta vector a is vacant up to point t iff $a[s] = 0$ for all $s \preceq t$ and that it is non-positive up to point t iff $a[s] \leq 0$ for all $s \preceq t$. A delta vector s is supported at point t iff there exists $s \prec t$ such that $a[s] < 0$ and a is non-positive up to s. We call s the support for t. A delta vector is upright iff all of its positive points are supported.

This definition of upright delta vectors arises because we use delta vectors to describe the changes in record counts that operations cause. As indicated in Section 1.1, we require that for any point t at which an operation causes a net production of records there must be a lower point s at which the operation causes a net consumption of records; this property explains why, in an upright delta vector, for each positive point t there must exist a negative point $s \preceq t$. For s to support t, we further require that all points $u \preceq s$ be non-positive; this property prevents cases of infinite descent. It yields, in particular, that the sum of two upright delta vectors is upright. (In cases where $\preceq$ is well-founded, infinite descent is impossible, so the further requirement becomes superfluous.)

CONSTANT $Point$ — set of points
CONSTANT $Proc$ — set of processors
CONSTANT $\_ \preceq \_$ — partial order on $Point$

$CountVec \triangleq [Point \rightarrow Nat]$ — count vectors
$DeltaVec \triangleq [Point \rightarrow Int]$ — delta vectors

$Z \triangleq [t \in Point \mapsto 0]$ — everywhere zero
$a \oplus b \triangleq [t \in Point \mapsto a[t] + b[t]]$ — component-wise addition
$a \ominus b \triangleq [t \in Point \mapsto a[t] - b[t]]$ — component-wise subtraction
$s \prec t \triangleq s \preceq t \land s \neq t$ — strictly lower

$IsVacantUpto(a, t) \triangleq \forall s \in Point : s \preceq t \Rightarrow a[s] = 0$
$IsNonposUpto(a, t) \triangleq \forall s \in Point : s \preceq t \Rightarrow a[s] \leq 0$
$IsSupported(a, t) \triangleq \exists s \in Point : s \prec t \land a[s] < 0 \land IsNonposUpto(a, s)$
$IsUpright(a) \triangleq \forall t \in Point : a[t] > 0 \Rightarrow IsSupported(a, t)$

VARIABLE $nrec \in CountVec$
VARIABLE $temp \in [Proc \rightarrow DeltaVec]$
VARIABLE $msg \in [Proc \rightarrow [Proc \rightarrow Seq(DeltaVec)]]$
VARIABLE $glob \in [Proc \rightarrow DeltaVec]$

$Init \triangleq$
$\land\ nrec \in CountVec$ — any initial population of records
$\land\ temp = [p \in Proc \mapsto Z]$ — no unsent changes
$\land\ msg = [p \in Proc \mapsto [q \in Proc \mapsto \langle\rangle]]$ — no unreceived updates
$\land\ glob = [q \in Proc \mapsto nrec]$ — each processor knows the initial $nrec$

$NextPerformOperation \triangleq \exists p \in Proc, c \in CountVec, r \in CountVec :$
LET $delta \triangleq r \ominus c$ IN — the net change in record population
$\land\ \forall t \in Point : c[t] \leq nrec[t]$ — only consume what exists
$\land\ IsUpright(delta)$ — net change must be upright
$\land\ nrec' = nrec \oplus delta$
$\land\ temp' = [temp$ EXCEPT $![p] = temp[p] \oplus delta]$
$\land$ UNCHANGED msg
$\land$ UNCHANGED $glob$

$NextSendUpdate \triangleq \exists p \in Proc, tt \in$ SUBSET $Point :$
LET $gamma \triangleq [t \in Point \mapsto$ IF $t \in tt$ THEN $temp[p][t]$ ELSE $0]$ IN
$\land\ gamma \neq Z$ — update must say something
$\land\ IsUpright(temp[p] \ominus gamma)$ — what is left must be upright
$\land$ UNCHANGED $nrec$
$\land\ temp' = [temp$ EXCEPT $![p] = temp[p] \ominus gamma]$
$\land\ msg' = [msg$ EXCEPT $![p] = [q \in Proc \mapsto Append(msg[p][q], gamma)]]$
$\land$ UNCHANGED $glob$

$NextReceiveUpdate \triangleq \exists p \in Proc, q \in Proc :$
LET $kappa \triangleq msg[p][q][1]$ IN — oldest unreceived update from p to q
$\land\ msg[p][q] \neq \langle\rangle$ — message queue must be non-empty
$\land$ UNCHANGED $nrec$
$\land$ UNCHANGED $temp$
$\land\ msg' = [msg$ EXCEPT $![p][q] = Tail(msg[p][q])]$
$\land\ glob' = [glob$ EXCEPT $![q] = glob[q] \oplus kappa]$

$Next \triangleq NextPerformOperation \lor NextSendUpdate \lor NextReceiveUpdate$

$Spec \triangleq Init \land \Box Next$

For any point t and processor q, if $glob[q]$ is vacant up to t, then, at this and all future times, $nrec$ is vacant up to t.

$Safe \triangleq \forall t \in Point, q \in Proc : (IsVacantUpto(glob[q], t) \Rightarrow \Box IsVacantUpto(nrec, t))$

Safe always holds in any execution that obeys *Spec*.

THEOREM $Spec \Rightarrow \Box Safe$

Figure 1.2: Basic specification of the clock protocol.

Finally, the clock protocol uses four state variables: $nrec$, $temp$, msg, and $glob$.

- $nrec$ is the occupancy vector, which represents the number of records that currently exist at each point.
- $temp[p]$ is the local (temporary) change in the occupancy vector due to the performance of operations at processor p. Note that the change at a given point can be negative (net records consumed), positive (net records produced), or zero. We call it temporary because eventually the processor takes the information from $temp[p]$ and broadcasts it as an incremental update.
- $msg[p][q]$ is the queue of updates from processor p to processor q. Each update is a delta vector that is zero everywhere except at those points that contain information about net changes. Implementations may of course limit the number of non-zero points and represent updates in a compact form.
- $glob[q]$ is the delayed view at processor q of the occupancy vector. It is a delta vector, rather than a count vector, because $glob[q][t]$ can be negative for some point t. Such negative values can appear, for example, when one processor p_1 produces a record at point t, a second processor p_2 consumes it and, because of different queuing delays, processor q receives the update from p_2 before that from p_1.

The basic specification of the clock protocol defines an initial $Init$, a next-state relation $Next$, and then a complete specification *Spec* which states that *Init* must hold and then forever each step must satisfy the $Next$ relation.

$Init$ states that $nrec$ can be any mapping from $Point$ to Nat; this mapping represents an arbitrary initial population of records. Initially, there are no unsent changes, no unreceived updates, and each processor knows the initial population.

Each step from a current state to a next state is an action specified as a relation between the values of the state variables in the current state (unprimed) and in the next state (primed). The algorithm has three actions: *NextPerformOperation*, *NextSendUpdate*, and *NextReceiveUpdate*.

- In the *NextPerformOperation* action, processor p performs an operation that consumes and produces some number of records at each point. The records to be consumed must exist and the net change in records $delta$ must be an upright delta vector. The action adds $delta$ to $nrec$ and to $temp[p]$.
- In the *NextSendUpdate* action, processor p selects a set of points tt and broadcasts an update about its changes at those points. The update is represented by $gamma$. The processor must choose tt in such a way that $temp[p] \ominus gamma$ is upright. This requirement holds, in particular, when tt consists of positive points in $temp[p]$ if any exist, because $temp[p]$ is always upright. The action subtracts $gamma$ from $temp[p]$ and appends $gamma$ to $msg[p][q]$ for all q.
- In the *NextReceiveUpdate* action, processor q selects a processor p and receives the oldest update $kappa$ on the message queue from p to q. For this action to take place, the current message queue $msg[p][q]$ must be non-empty. The action adds $kappa$ to $glob[q]$ and removes it from $msg[p][q]$.

The next-state relation $Next$ is simply the disjunction of the relations for these three actions.

The main safety property of the clock protocol is *Safe*, which states that if any processor q has a $glob[q]$ that is vacant up to some point t, then the actual set of records, $nrec$ is vacant up to point t. Our goal is to establish that this safety property always holds in every execution that obeys the specification.

Chapter 2

Discussion of the specification

Appendix A gives a full TLA+ specification of the Naiad Clock protocol.

The full TLA+ specification follows the outline of the basic specification presented in Section 1.2 in most regards. However, there are some differences.

The basic specification uses $\preceq$ for the partial order. In order to facilitate model checking, the full specification uses the variable $lleq$ to hold the partial order. This variable is initialized to any partial order and never changed afterwards. This permits the model checker to explore the state space separately for each possible partial order.[1]

For clarity, the basic specification uses short names for operators and definitions. For example, it uses Z for the everywhere zero delta vector and $\oplus$ and $\ominus$ for addition and subtraction of delta vectors. The full specification uses long names for everything. Using long names is perhaps a bit more cumbersome, but it prevents name collisions. TLA+ absolutely forbids name collisions. It is possible to perform named instantiation of TLA+ modules in order to use multiple modules that would otherwise have name collisions, but that solution is even more cumbersome than using long names.

The full specification admits stutter steps (steps in which nothing changes), in addition to steps performed by the defined actions. TLA+ encourages writing specifications that admit stutter steps in order to make it possible to prove a refinement mapping by using a one-to-one correlation of states. The basic specification does not envision stutter steps.

In addition to the main safety property, the full specification includes a couple of additional safety properties and several invariants. The model checker can easily check invariants. The model checker can also check general safety properties, but to do so it has to keep information about the entire state graph, which causes it to run much slower.

However, the full specification uses a trick to enable the model checker to check two of the safety properties as simple state predicates. These two safety properties are "sticky" in the sense that once some state predicate is true of some state, it remains true for all following states. By adding a state variable to remember the value of the predicate from the previous state, the "stickiness" can be checked as a simple state predicate. The full specification introduces the two state variables $nrecvut$ and $globvut$ for this purpose.

[1] An alternative solution, perhaps more in the style of TLA+, would be to declare $lleq$ as a constant and then construct a mapping from each possible partial order to an instantiation of the clock protocol specification. Unfortunately, this solution would greatly explode the number of states the model checker would have to explore, because each state would correspond to a mapping from the partial orders to a state within an execution for that partial order.

Chapter 3

Discussion of model checking

Appendix B gives a TLA+ extension of the Naiad Clock protocol that defines default constants and introduces a constraint so that model checking has only a finite number of states to explore. The configuration parameters are as follows:

- $MaxProc$, the number of processors.
- $MaxPoint$, the number of points.
- $MaxRecPerPoint$, the maximum number of records per point that exist (in $nrec$) in any state.
- $MaxRec$, the maximum total number of records that exist (in $nrec$) in any state.
- $MaxMsgPerQueue$, the maximum number of messages that can be on any single queue in any state.

We used the TLA+ toolbox [4] to construct and manage models for model checking the specification.

Using a 2.67 GHz Intel i7 with 4 GB of memory running TLC2 version 2.05, we model checked the specification in various configurations. For each configuration, TLC determined the maximum depth of the state space graph as well as the total number of distinct states. Table 3.1 shows the statistics.

As expected, the number of distinct states and consequently the model checking run time blow up enormously as the configuration parameters are increased. This limits the feasibility of model checking of this specification to small configurations only.

Using the model checker, we checked the following invariants:

$MaxProc$	$MaxPoint$	$MaxRecPerPoint$	$MaxRec$	$MaxMsgPerQueue$	depth	distinct states	run time (sec)
2	2	1	2	1	12	2690	5
2	2	1	2	2	14	5286	5
2	2	1	2	3	14	6110	5
2	2	2	2	1	17	47192	27
2	2	2	2	2	21	271870	121
2	2	2	2	3	22	538738	201
2	2	2	4	1	18	278138	184
2	2	2	4	2	23	3418972	2053
2	2	2	4	3	28	13293954	5785
2	3	1	2	1	21	1461100	1502
2	3	1	2	2	25	16744480	19339

Table 3.1: Model checking statistics. Complete state space exploration.

- $InvType$, which states that all state variables contain values of their expected types.
- $InvTempUpright$, which states that $temp[p]$ is upright.
- $InvGlobalRecordCount$, which states that $glob[q]$ plus all infomation heading toward q equals $nrec$.
- $InvStickyNrecVacantUpto$, which states that if $nrec$ is vacant up to point t, then it will be so

in the next state. This invariant is checked using the fiducial variable $nrecvut$, which remembers $IsVacantUpto(nrec, t)$ from the previous state.

- $InvStickyGlobVacantUpto$, which states that if $glob[q]$ is vacant up to point t, then it will be so in the next state. This invariant is checked using the fiducial variable $globvut$, which remembers $IsVacantUpto(glob[q], t)$ from the previous state.

- $InvGlobVacantUptoImpliesNrec$, which states that if $glob[q]$ is vacant up to a point t, then so is $nrec$.

In every state explored by the model checker, all of these invariants were found to hold.

Based on these model checking results, we were fairly confident that the specification was correct. However, because of state space explosion, we could only check some small configurations using 2 processors and 3 points in virtual time. In the next chapter we discuss our formal proof.

Chapter 4

Discussion of the proof

Appendix C gives a TLA+ proof of the Naiad Clock protocol invariants. The proof has been mechanically checked using the TLA+ Proof System [2, 3] except for a few minor details. Unfortunately, the current TLA+ proof system cannot handle temporal reasoning, so any temporal deductions have to be checked manually. Fortunately, the vast majority of the proof deals with state predicates and next state relations, all of which is checked mechanically. Only the final steps in proving that the specification implies some temporal property require temporal deductions and thus have to be checked manually.

The proof is quite long, so we divided it into modules for ease of understanding and management. In Section 4.1, we walk through the proof and explain what each module accomplishes. In Section 4.2, we give a brief description of the TLA+ proof system. In Section 4.3, we discuss the performance of the proof system in checking our proof. In Section 4.4, we discuss what we learned about writing and checking such a large proof.

4.1 A walk through the proof

The proof is divided into modules for ease of understanding and management. Each module contains a collection of theorems and definitions relating to a certain concept. As we discuss in Section 4.4.1, the proof is composed of modules that build on one another in a linear sequence.

4.1.1 Basic definitions

NaiadClockProofBase (C.1) provides some additional definitions that are needed in the proof but do not appear in the specification. For example, the proof needs the concept of a beta-upright delta vector, which is a generalization of the concept of an upright delta vector. It also turns out to be useful in the proof to have symbolic definitions for various formulas that appear written out in the specification. This lets proof steps use these formulas symbolically, which helps keep the back-end provers from getting lost when trying to check proof obligations.

4.1.2 Basic library theorems

Next follow a number of modules that contain various theorems about naturals (C.2), sequences (C.3), the $RemoveAt$ sequence operator (C.4), finite sets (C.5), exact sequences (C.6), and partial orders (C.7). We consider these modules as library modules, because their theorems are of general usefulness.

4.1.3 Properties of delta vectors

Next come several modules that prove various properties of delta vectors.

NaiadClockProofDeltaVecs (C.8) proves that the addition of delta vectors is commutative, associative, closed, and has an identity. In other words, that it is a commutative monoid.

NaiadClockProofDeltaVecSeqs (C.9) contains theorems about the sum of a sequence of delta vectors. These theorems have to dig inside the recursive definition of the sum of a sequence of delta vectors and they are extremely tedious. We consider this module as a library module because it could be recast in general terms to apply to any commutative monoid.

NaiadClockProofDeltaVecFuns (C.10) contains theorems about the sum of the delta vectors in the range of a function. These theorems are also extremely tedious. We consider this module as a library module because it could be recast in general terms to apply to any commutative monoid.

NaiadClockProofDeltaVecUpright (C.11) contains theorems about upright delta vectors, especially the theorem that the sum of two upright delta vectors is upright and the corollaries for the sum of a sequence of delta vectors and for the sum of the delta vectors in the range of a function.

NaiadClockProofDeltaVecBetaUpright (C.12) contains theorems about beta-upright delta vectors.

NaiadClockProofDeltaVecVacantUpto (C.13) contains theorems about delta vectors that are vacant up to a given point.

4.1.4 Additional invariants

NaiadClockProofInvariants (C.14) introduces the definitions of some additional invariants that are needed in the proof.

4.1.5 Deduction of some invariants

NaiadClockProofDeduceInv (C.15) contains theorems that deduce certain invariants from others. These theorems state the deductions in both the current state and in the next state, as we describe in Section 4.4.4.

4.1.6 The effects of actions

The next several modules contain theorems about the effects of the actions. As we discuss in Section 4.4.2, we discovered that many of the same deductions about various effects of actions kept reappearing in proofs of the various invariants. The entire proof was made much simpler by refactoring these deductions into their own theorems, which we call action effect theorems. As we discuss in Section 4.4.3, the conclusions of the action effect theorems tend to be quite complicated, with multiple conjuncts and internal case analysis, and the back-end provers tended to have difficulty in applying them. We solved this latter problem by defining symbolic predicates for the conclusions.

NaiadClockProofAffectState (C.16) contains theorems on how the actions affect the state variables.

NaiadClockProofAffectInfoAt (C.17) contains theorems on how the actions affect the state operator $InfoAt$.

NaiadClockProofAffectIncomingInfo (C.18) contains theorems on how the actions affect the state operator $IncomingInfo$.

NaiadClockProofAffectGlobalIncomingInfo (C.19) contains theorems on how the actions affect the state operator $GlobalIncomingInfo$.

4.1.7 Proving invariants

The next several modules contain theorems that prove invariants. Each module deals with one invariant and contains three main theorems: first a theorem that the invariant holds in the initial state, then a theorem that the invariant is maintained through the next state relation, and then a finally a theorem that the specification implies that the invariant always holds. The proof of this last theorem requires a temporal deduction and therefore can not entirely be checked mechanically by the current TLA+ proof system.

NaiadClockProofInvType (C.20) proves that all state variablaes always have their expected types.

NaiadClockProofInvTempUpright (C.21) proves that $temp[p]$ is always upright.

NaiadClockProofInvIncomingInfoUpright (C.22) proves that any suffix of incoming infomation is always upright.

NaiadClockProofInvInfoAtBetaUpright (C.23) proves that any update is always beta-upright with the incoming information behind it.

NaiadClockProofInvGlobalRecordCount (C.24) proves that the sum of $glob[q]$ plus all infomation heading toward q is always $nrec$.

NaiadClockProofInvStickyNrecVacantUpto (C.25) proves that whenever $nrec$ is vacant up to a point t, it stays that way for all future times.

NaiadClockProofInvStickyGlobVacantUpto (C.26) proves that whenever $glob[q]$ is vacant up to a point t, it stays that way for all future times.

4.1.8 Proving the main safety properties

Finally, *NaiadClockProof* (C.27) contains theorems that prove the main safety properties. The proofs of these theorems basically consist of appealing to prior theorems about invariants and then making some temporal deductions. Unfortunately, the temporal deductions cannot be checked mechanically by the current TLA+ proof system.

4.2 Proof system overview

The proof system consists of a proof manager *tlapm* that parses the TLA+ modules, expands definitions, constructs proof obligations, and employs back-end provers to discharge obligations.

To discharge an obligation, tlapm itself first checks to see if the obligation is "trivially identical" with some known fact.[1] If this fails, tlapm then hands the obligation off to the back-end provers.

By default, tlapm first invokes Zenon [1], a tableau prover for classical first-order logic with equality. Zenon is generally quick to solve simple problems, but tends to fail on anything complicated. If Zenon fails, tlapm then invokes Isabelle [6] using a specialized TLA+ object logic that includes propositional and first-order logic, elementary set theory, functions, and the construction of natural numbers.

Pragmas can be used to direct tlapm to appeal to other back-end provers. An entire catagory of provers based on Satisfiability Modulo Theory (SMT) is especially good with some hard problems involving arithmetic, uninterpreted functions, and quantifiers. The back-end prover *smt3* is one such SMT prover.

4.3 Proof statistics

The entire proof contains 27 modules, 146 theorems, and 10743 lines. Using an Intel® Core™ i7 CPU M 620 laptop with 4 GB of memory running at 2.67 GHz, the entire proof is verified in less than two hours.

Table 4.1 shows a number of statistics for each module in the proof. The line counts are based on the number of lines in the ASCII TLA+ source files, which may differ slightly from the number of lines in the typeset TLA+ format.

It turns out that dividing the proof into modules is also necessary to enable the proof manager to handle the proof. We tried combining everything into one long module and then asking the proof manager to prove the entire thing. It failed due to running out of Java heap space before collecting even one-third of the total obligations.

As can be seen in Table 4.1, about two-thirds of the total proof obligations are discharged by tlapm itself, meaning that they are "trivially identical" to some known fact. This might seem surprising, but it results from the way the proof manager treats the adduced facts mentioned in the BY clause of each leaf proof step. Each of these adduced facts is considered as a separate proof obligation that must be discharged. In the general case, one could write an arbitrary formula as an adduced fact. However, we never do that: instead, we always just reference some earlier proof step or theorem. Nonetheless, the way the proof manager is currently implemented, it examines the adduced fact, compares it against all known facts while expanding all useable definitions, and eventually arrives at the conclusion that, yes, indeed, the adduced fact is "trivially identical" to some known fact.

The remaining obligations actually require work by a back-end prover. Almost all of these are proved by Zenon, which shows the utility of this prover. The smt3 prover is needed for several hundred obligations that depend on arithmetic properties. The remaining few obligations are proved by Isabelle.

4.4 What we learned

4.4.1 Linear module structure

The module structure in the proof is completely linear. That is to say, each module in the proof extends the previous module, in a strictly linear order.

The linear module structure is not the most logical organization of the modules in the proof. For example, the module *NaiadClockProofDeltaVecSeqs* (C.9) contains theorems about properties of summing up sequences of delta vectors. These properties depend on the fact that addition of delta vectors is commutative and associative.

[1] In the current implementation, "trivially identical" means identical up to renaming of bound variables, after expanding all usable definitions.

	module name	theorems	lines	run time (sec)	obligations proved by isabelle	smt3	tlapm	zenon
C.1	NaiadClockProofBase	0	114	50	0	0	0	0
C.2	NaiadClockProofNaturals (lib)	5	120	98	1	15	37	32
C.3	NaiadClockProofSequences (lib)	18	500	139	7	12	119	73
C.4	NaiadClockProofRemoveAt (lib)	1	393	152	4	21	180	76
C.5	NaiadClockProofFiniteSets (lib)	5	225	127	2	9	114	67
C.6	NaiadClockProofExactSeqs (lib)	6	511	308	0	39	368	218
C.7	NaiadClockProofPartialOrders (lib)	4	145	62	0	0	16	11
C.8	NaiadClockProofDeltaVecs	7	153	66	4	3	0	10
C.9	NaiadClockProofDeltaVecSeqs (lib)	19	1805	674	23	98	978	602
C.10	NaiadClockProofDeltaVecFuns (lib)	17	1149	385	1	2	452	312
C.11	NaiadClockProofDeltaVecUpright	6	308	114	1	5	95	54
C.12	NaiadClockProofDeltaVecBetaUpright	5	364	128	1	5	110	68
C.13	NaiadClockProofDeltaVecVacantUpto	2	226	91	0	9	83	37
C.14	NaiadClockProofInvariants	0	153	51	0	0	0	0
C.15	NaiadClockProofDeduceInv	7	594	225	3	12	230	160
C.16	NaiadClockProofAffectState	4	677	256	1	11	328	176
C.17	NaiadClockProofAffectInfoAt	5	329	163	1	22	163	83
C.18	NaiadClockProofAffectIncomingInfo	4	383	156	2	2	140	86
C.19	NaiadClockProofAffectGlobalIncomingInfo	5	429	198	2	3	167	115
C.20	NaiadClockProofInvType	3	169	100	1	1	48	38
C.21	NaiadClockProofInvTempUpright	3	199	97	0	0	59	41
C.22	NaiadClockProofInvIncomingInfoUpright	3	224	109	0	1	69	54
C.23	NaiadClockProofInvInfoAtBetaUpright	3	382	172	0	9	176	105
C.24	NaiadClockProofInvGlobalRecordCount	3	299	131	0	0	104	71
C.25	NaiadClockProofInvStickyNrecVacantUpto	4	304	120	0	3	97	63
C.26	NaiadClockProofInvStickyGlobVacantUpto	4	417	160	4	0	145	87
C.27	NaiadClockProof	3	171	63	0	0	27	10
	library (8 modules) total:	**75**	**4848**	**1945**	**38**	**196**	**2264**	**1391**
	special (19 modules) total:	**71**	**5895**	**2450**	**20**	**86**	**2041**	**1258**
	(27 modules) Total:	**146**	**10743**	**4395**	**58**	**282**	**4305**	**2649**

Table 4.1: Statistics by proof module. Modules indicated by (lib) are of general interest, or could be so rewritten, and we consider them as library modules.

It would be a more logical organization to have written a library module that proved such properties for any commutative and associative binary operation and then instantiated this module for the particular operation of delta vector addition. And it would be more logical to make each module extend only those modules upon which it directly depended.

We originally tried writing the proof with this more logical organization. Unfortunately, this organization had the result of causing the current TLA+ proof manager to bog down and become so slow that it was unusable. Our speculation is that the "logical organization" resulted in an exponentially growing number of extension paths reaching to the more fundamental theorems and that the current TLA+ proof manager wastes itself in searching through these paths in trying to find "trivial" matches for each proof obligation.

Writing the proof with a linear module structure avoids this TLA+ proof manager performance problem.

We created an example set of modules that exhibited this TLA+ proof manager performance problem and supplied it to the implementation team at MSR-INRIA. Damien Doligez investigated the problem and fixed the proof manager to avoid it. However, by this time we had already completed the linear module structure of our proof and we did not want to spend the time to recast it back into a more logical structure of library modules.

4.4.2 Refactoring action effects

The proof of the main safety properties relies on a number of invariants. When developing the subproofs of how each of the actions maintain these invariants, we discovered that we were often repeating many of the same deductions from one invariant to another. This was tedious.

So we refactored the proof by breaking out separate theorems about how each action affected the state variables and each of the state operators. This refactoring greatly simplified many of the invariant proofs. Furthermore, it made it much easier later with slight revisions to the specification, because generally much of the required proof changes occurred in the action effect theorems without impacting the rest of the proof.

4.4.3 Symbolic conclusions

We found that the back-end provers had little success in applying theorems whose statements are complicated. This was a particular problem with the action effect theorems, whose conclusions often include a case analysis. For example, *NextSendUpdate* appends a delta vector on all queues from processor p: the effect of this action on the message queue from processor fp to processor fq depends on whether $p = fp$ or not.

The solution to this problem was to define a symbolic predicate that captured the conclusion of such a theorem. When the provers were faced with verifying such a symbolic predicate, they usually had no difficulty applying the theorem. Then the proof could continue, deducing any desired conclusion from the predicate by expanding its definition.

A few of the theorems had complicated assumptions that seemed to suffer from the same problem. So in these cases we created a symbolic predicate for the theorem's hypothesis. In order to use the theorem, we first establish the hypothesis, using its definition. Then the back-end provers can see how to justify the theorem's conclusion by applying the symbolic hypothesis.

4.4.4 Parallel deduction

Several of the invariants in the proof can be deduced from other invariants. For example, given that all state variables contain values of their expected types, we can deduce that each of the state operators also compute values of their expected types.

Since these deductions involve state predicates without involving any temporal operators, the exact same deduction works in both the current state (unprimed) and in the next state (primed). Essentially, what we have is a proof macro that can be expanded to a proof in the current state and to a proof in the next state. Unfortunately, there is no provision for proof macros in the TLA+ proof system.

However, we found a way to express the proof macro, as follows. What we did for each proof was define a local operator

$$DoPr(primeit, x) \triangleq \text{IF } primeit \text{ THEN } x' \text{ ELSE } x$$

and then wrap all state variables and state operators in the steps in the proof inside instances of this operator.

Then by setting the proof steps in a context in which $primeit \in$ BOOLEAN, we manage to prove a conclusion in both the current state and in the next state simultaneously.

The only problem with this approach is that all of the proof steps were more difficult for the back-end provers to verify, since the formulas were littered with $DoPr$ all over the place. Usually, the provers managed to verify the proof steps anyway. When this turned out to be too difficult, our solution was to use PICK to define new constants for each of the state variables and state operators based on their $DoPr$ wrappings. The necessary properties of the new constants could be proved from their definitions, and then further deductions using the new constants could be checked without difficulty.

4.4.5 Checking the entire proof

Although the TLA+ proof system provides a nice interface for checking the proofs of theorems in a single module, it currently lacks any ability to run a complete check on a multi-module proof. So we wrote a Perl script that, given a top-level TLA+ module, determines the closure of all referenced modules, invokes the proof manager on each module, and then collects and summarizes the results.

Whenever we tweaked some definition or theorem, after we were fairly sure it was correct, we would run the Perl script to verify that the entire proof was still correct. We also used this Perl script to prepare the statistics listed in Table 4.1.

We found it particularly important to collect information about proof obligations that failed to be verified by the back-end provers. From time to time we would discover that a back-end prover would take more run time than usual when trying to check a proof obligation, and it would exhaust its time allocation from the proof manager, resulting in a failed proof obligation. Also, whenever the proof system implementers released an update, usually the back-end provers became more capable but there were sometimes cases in which the new release failed on some obligation that the prior release had checked successfully.

When we found such a failed obligation, we took one of two approaches to fix it. First, often the problem was that the back-end prover just occasionally needed more run time. In this case, we annotated the proof step with a pragma that instructed the proof manager how much time to give. After several iterations of this issue with various proof steps that needed to be checked by SMT, we just changed all of these steps from the 5 second SMT default to 10 seconds.

Second, sometimes the problem was that the back-end prover had retrogressed and was no longer capable of checking the proof step in the context in which it appeared.[2] Of course, we reported these cases to the proof system implementers, but then we still had to fix the proof. We found two approaches that worked. Usually we decomposed the failing proof step into simpler steps that were easier to check. However, if the proof step was already blindingly simple, it would work to create a new theorem that specifically applied to the deduction we wanted to prove. The new theorem could often be easier for the back-end provers to check because it isolated the deduction from whatever context existed at the proof step where we wanted to use it.

Generally, we found that the back-end provers could often be distracted by an excessive context containing too many usable facts. The TLA+ proof system provides ways of managing the set of usable facts, and such management is often an important contributor to a back-end prover's success. Unfortunately, the way the proof manager currently works, some facts, such as the types of introduced constants, cannot be excluded from the proof obligation, even if they are irrelevant. We found a few cases were such irrelevant facts would cause the back-end provers to fail.

[2] Usually, what had happened was that the proof manager's translation of a proof obligation into the language of the back-end prover had retrogressed, or perhaps the description of the theory used by the back-end prover had retrogressed. Such retrogression all looks the same to the user of the proof system.

Acknowledgements

We would like to acknowledge various people who helped with the proof.

Thanks to Martín Abadi for a hand-proof of an initial formulation of the safety properties, and for many discussions on the exact definitions which should be used in the specification and safety properties.

Thanks to Leslie Lamport for inspiring the development of TLA+ and its tools and for many discussions on how to use TLA+ and the proof system. Leslie also demonstrated that an SMT solver could directly prove an earlier version of the theorem that the sum of upright delta vectors is upright.

Thanks to Stephan Merz and Damien Doligez for assistance in using the TLA+ proof system, especially for improving it and fixing the occasional bug that we found.

Thanks to Frank McSherry and the other members of the Naiad project for help in understanding how the Naiad Clock Protocol fits into the greater scheme of Naiad.

Bibliography

[1] R. Bonichon, D. Delahaye, and D. Doligez. Zenon: An extensible automated theorem prover producing checkable proofs. In *Proc. 14th LPAR*, pages 151–165, 2007.

[2] K. Chaudhuri, D. Doligez, L. Lamport, and S. Merz. A TLA+ proof system. In *Proc. Combined KEAPPA - IWIL Workshops*, pages 17–37, 2008. `http://ceur-ws.org/Vol-418/paper2.pdf`.

[3] K. Chaudhuri, D. Doligez, L. Lamport, and S. Merz. Verifying safety properties with the TLA+ proof system. In *IJCAR*, pages 142–148, 2010.

[4] L. Lamport. The TLA toolbox. `http://research.microsoft.com/en-us/um/people/lamport/tla/toolbox.html`.

[5] L. Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.

[6] L. C. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer, Berlin, Germany, 1994.

[7] The Naiad project. Forthcoming systems paper on Naiad.

[8] The Naiad project. Naiad (project web page). `http://research.microsoft.com/en-us/projects/naiad/`.

Appendix A

Specification

MODULE $NaiadClock$

EXTENDS $Naturals$, $Integers$, $FiniteSets$, $Sequences$

CONSTANT $Point$ set of points
CONSTANT $Proc$ set of processors

Set of possible record-point configurations that can be initialized, consumed, or produced. In general, this is [$Point \rightarrow Nat$], but for model checking we have to be able to supply some finite set of possibilities.

CONSTANT $PointToNat$
ASSUME $AssumePointToNat \triangleq PointToNat \subseteq [Point \rightarrow Nat]$

A relation between points.

$PointRelationType \triangleq [Point \rightarrow [Point \rightarrow \text{BOOLEAN}]]$

Definition of a partial order.

$IsPartialOrder(leq) \triangleq$

$\quad \land \forall s, t, u \in Point : leq[s][t] \land leq[t][u] \Rightarrow leq[s][u]$ transitive
$\quad \land \forall s, t \in Point : leq[s][t] \land leq[t][s] \Rightarrow (s = t)$ antisymmetric
$\quad \land \forall s \in Point : leq[s][s]$ reflexive

Design comment:

Preferably, I would create separate modules for exact sequences, the remove at operator, and delta vectors, and parameterized modules for summing sequences and summing the range of functions. This would be a cleaner design than the presentation here in which everything appears in one module.

Unfortunately, since each of these modules would depend on many lower-layer modules, there would be many layers of duplicative module extension and instantiation, which sadly creates an impossible performance drag on the current proof manager, as it tries to perform its “trivial” identity check of each proof obligation against an apparent exponential explosion of known facts. The presentation here avoids this performance drag.

Exact sequences.

Each $s \in S$ appears on Q.

$$ExactSeq\_Each(Q, S) \triangleq \forall s \in S : \exists i \in 1 \,..\, Len(Q) : Q[i] = s$$

Anything on Q appears at most once.

$$ExactSeq\_Once(Q) \triangleq \forall i, j \in 1 \,..\, Len(Q) : Q[i] = Q[j] \Rightarrow i = j$$

Q is an exact sequence for the set S.

$$\begin{aligned} &IsExactSeqFor(Q, S) \triangleq \\ &\quad \land Q \in Seq(S) \\ &\quad \land ExactSeq\_Each(Q, S) \\ &\quad \land ExactSeq\_Once(Q) \end{aligned}$$

For any finite set S, choose a sequence in which each element of S appears exactly once.

$$ExactSeqFor(S) \triangleq \text{CHOOSE } Q : IsExactSeqFor(Q, S)$$

Delta vectors.

A delta vector maps each point to an integer.

$$DeltaVecType \triangleq [Point \rightarrow Int]$$

The zero delta vector is everywhere zero.

$DeltaVecZero \triangleq [t \in Point \mapsto 0]$

Pointwise addition of delta vectors.

$DeltaVecAdd(a, b) \triangleq [t \in Point \mapsto a[t] + b[t]]$

Pointwise negation of a delta vector.

$DeltaVecNeg(a) \triangleq [t \in Point \mapsto 0 - a[t]]$

A delta vector v is vacant up to point t iff for all points $s \preceq t$ we have $v[s] = 0$.

$IsDeltaVecVacantUpto(leq, v, t) \triangleq$
LET
$a \preceq b \triangleq leq[a][b]$
$a \prec b \triangleq a \preceq b \land a \neq b$
IN
$\forall s \in Point : s \preceq t \Rightarrow v[s] = 0$

A delta vector v is nonpos up to point t iff for all points $s \preceq t$ we have $\neg(v[s] > 0)$.

$IsDeltaVecNonposUpto(leq, v, t) \triangleq$
LET
$a \preceq b \triangleq leq[a][b]$
$a \prec b \triangleq a \preceq b \land a \neq b$
IN
$\forall s \in Point : s \preceq t \Rightarrow \neg(v[s] > 0)$

A delta vector v is supported at point t iff there exists a point $s \prec t$ such that $v[s] < 0$ and v is non-positive up to s.

$IsDeltaVecSupported(leq, v, t) \triangleq$
LET
$a \preceq b \triangleq leq[a][b]$
$a \prec b \triangleq a \preceq b \land a \neq b$
IN
$\exists s \in Point :$
$\land s \prec t$
$\land v[s] < 0$
$\land IsDeltaVecNonposUpto(leq, v, s)$

A delta vector v is upright iff it is supported at every positive point.

$IsDeltaVecUpright(leq,\ v) \triangleq$
LET
$a \preceq b \triangleq leq[a][b]$
$a \prec b \triangleq a \preceq b \land a \neq b$
IN
$\forall\, t \in Point : v[t] > 0 \Rightarrow IsDeltaVecSupported(leq,\ v,\ t)$

Summing up a sequence of delta vectors.

The sum of a sequence of delta vectors, skipping the first k.

We define the sum in terms of a recursive function over the naturals. Such a recursive function is the only formulation for which the current $TLAPS$ libraries provide theorems to help prove things. Based on the complexity of the proofs of those library theorems, I don't want to start trying to roll my own.

The recursive function $Sumv[i]$ starts at element i and recursively sums up each element going backwards towards element 1. The actual computation is

$$((\ldots(((0 + Q[1]) + Q[2]) + Q[3]) + \ \ldots \ + Q[i-1]) + Q[i])$$

Since the recursive function has to be defined for all naturals, and not just those that are in the domain of the sequence, we make it look through an infinte extension of the sequence created by the operator $Elem$. $Elem(i)$ just returns $Zero$ whenever i is greater than the length the sequence.

$Elem(i)$ also handles the job of skipping the first k elements of the sequence. It does this through the simple expedient of returning $Zero$ whenever i is not greater than k. Note that if you feed in $k = 0$ you get the sum of the entire sequence.

$DeltaVecSeqSkipSum(k,\ Q) \triangleq$
LET
$Zero \triangleq DeltaVecZero$
$Add(a,\ b) \triangleq DeltaVecAdd(a,\ b)$
$n \triangleq Len(Q)$
$Elem(i) \triangleq$ IF $k < i \land i \leq n$ THEN $Q[i]$ ELSE $Zero$
$Sumv[i \in Nat] \triangleq$ IF $i = 0$ THEN $Zero$ ELSE $Add(Sumv[i-1],\ Elem(i))$
IN
$Sumv[n]$

The sum of a sequence of delta vectors. This is just the special case $k = 0$.

$DeltaVecSeqSum(Q) \triangleq DeltaVecSeqSkipSum(0,\ Q)$

Construct a sequence of delta vectors just like Q, but add d to element $Q[n]$.

$DeltaVecSeqAddAt(Q,\ n,\ d) \triangleq$
LET

$Zero \triangleq DeltaVecZero$
$Add(a, b) \triangleq DeltaVecAdd(a, b)$
IN
$[Q \text{ EXCEPT } ![n] = Add(Q[n], d)]$

Summing up delta vectors in the range of a function.

Given a function F with range $DeltaVecType$ and a sequence $I \in Seq(\text{DOMAIN } F)$, compute the sum of $F[I[i]]$ over all $i \in 1 \,..\, Len(I)$.

$DeltaVecFunIndexSum(F, I) \triangleq$
LET
$Zero \triangleq DeltaVecZero$
$Add(a, b) \triangleq DeltaVecAdd(a, b)$
$SeqSum(Q) \triangleq DeltaVecSeqSum(Q)$
IN
$SeqSum([i \in 1 \,..\, Len(I) \mapsto F[I[i]]])$

Given a function F with range $DeltaVecType$ and a finite set $S \subseteq \text{DOMAIN } F$, compute the sum of $F[s]$ for all $s \in S$.

$DeltaVecFunSubsetSum(F, S) \triangleq$
LET
$Zero \triangleq DeltaVecZero$
$Add(a, b) \triangleq DeltaVecAdd(a, b)$
$SeqSum(Q) \triangleq DeltaVecSeqSum(Q)$
IN
$DeltaVecFunIndexSum(F, ExactSeqFor(S))$

Given a function F with range $DeltaVecType$ determine if the set of all $s \in \text{DOMAIN } F$ such that $F[s] \neq Zero$ is finite.

$DeltaVecFunHasFiniteNonZeroRange(F) \triangleq$
LET
$Zero \triangleq DeltaVecZero$
$Add(a, b) \triangleq DeltaVecAdd(a, b)$
$SeqSum(Q) \triangleq DeltaVecSeqSum(Q)$
IN
$IsFiniteSet(\{d \in \text{DOMAIN } F : F[d] \neq Zero\})$

Given a function F with range $DeltaVecType$ such that the set S of all $s \in \text{DOMAIN } F$ such that $F[s] \neq Zero$ is finite, compute the sum of $F[s]$ over all $s \in S$.

$DeltaVecFunSum(F) \triangleq$

LET
$Zero \triangleq DeltaVecZero$
$Add(a, b) \triangleq DeltaVecAdd(a, b)$
$SeqSum(Q) \triangleq DeltaVecSeqSum(Q)$
IN
$DeltaVecFunSubsetSum(F, \{d \in \text{DOMAIN } F : F[d] \neq Zero\})$

Given a function F with range $DeltaVecType$, construct a function just like it but with v added to component x.

$DeltaVecFunAddAt(F, x, v) \triangleq$
LET
$Zero \triangleq DeltaVecZero$
$Add(a, b) \triangleq DeltaVecAdd(a, b)$
$SeqSum(Q) \triangleq DeltaVecSeqSum(Q)$
IN
$[F \text{ EXCEPT } ![x] = Add(F[x], v)]$

Other data types.

A count vector gives a count of records for each point.

$CountVecType \triangleq [Point \rightarrow Nat]$

State variables.

"$lleq$" is a state variable so that $Init$ can initialize it to any partial order, for the purpose of model checking. Afterwards, it never changes.

VARIABLE $lleq$ — the precedence between points

VARIABLE $nrec$ — how many records at each point

VARIABLE $glob$ global count for each processor and point
VARIABLE $temp$ temporary count for each processor and point
VARIABLE msg clock message queues between processors

fiducial variables

VARIABLE $nrecvut$ in prev state $nrec$ was vacant at all points up thru t
VARIABLE $globvut$ in prev state $glob$ was vacant at all points up thru t

$vars \triangleq \langle lleq,\ nrec,\ glob,\ temp,\ msg,\ nrecvut,\ globvut \rangle$

State operators.

All points s up thru t have no records.

$NrecVacantUpto(t) \triangleq IsDeltaVecVacantUpto(lleq,\ nrec,\ t)$

All points s up thru t have zero in $glob[q]$.

$GlobVacantUpto(q,\ t) \triangleq IsDeltaVecVacantUpto(lleq,\ glob[q],\ t)$

After skipping the first k, the sum of the delta vectors on the message queue from p to q, plus $temp[p]$.

$IncomingInfo(k,\ p,\ q) \triangleq$
  LET
    $sum \triangleq DeltaVecSeqSkipSum(k,\ msg[p][q])$
  IN
  $DeltaVecAdd(sum,\ temp[p])$

The sum of all incoming information heading toward processor q, except for skipping the first k delta vectors coming from p.

Observe that $GlobalIncomingInfo(0,\ q,\ q)$ is a way to refer to the sum of all incoming information heading toward processor q.

$GlobalIncomingInfo(k,\ p,\ q) \triangleq$
  LET
    $F \triangleq [xp \in Proc \mapsto$
        LET $xk \triangleq$ IF $xp = p$ THEN k ELSE 0 IN

$IncomingInfo(xk, xp, q)$
]
IN
$DeltaVecFunSum(F)$

Next state relation.

Common part of each next action.

$NextCommon \triangleq$

The partial order cannot change.

$\land$ UNCHANGED $lleq$

Compute fiducial variables based on old state.

$\land\ nrecvut' = [ft \in Point \mapsto NrecVacantUpto(ft)]$
$\land\ globvut' = [fq \in Proc \mapsto [ft \in Point \mapsto GlobVacantUpto(fq, ft)]]$

Perform an operation.

$NextPerformOperation \triangleq$
$\exists\, p \in Proc :$ any processor
$\exists\, c \in PointToNat :$ consumed records per point
$\exists\, r \in PointToNat :$ result records per point
LET
$delta \triangleq [t \in Point \mapsto r[t] - c[t]]$
IN

Can consume only such records as exist.

$\land\ \forall\, t \in Point : c[t] \leq nrec[t]$

delta must be an upright delta vector.

$\land\ IsDeltaVecUpright(lleq, delta)$
$\land\ nrec' = DeltaVecAdd(nrec, delta)$
$\land\ temp' = [temp$ EXCEPT $![p] = DeltaVecAdd(temp[p], delta)]$
$\land$ UNCHANGED $glob$
$\land$ UNCHANGED msg

$\wedge\ NextCommon$

Send an update. The update is broadcast to all processors. The processor is required to choose a set of points in its $temp$ array to send that will leave its $temp$ array as an upright delta vector.

One simple way to do this is to always send positive points in preference to negative points.

$NextSendUpdate \triangleq$

$\quad \exists\, p \in Proc :$

$\quad \exists\, tt \in \text{SUBSET } Point :$

$\quad \text{LET}$

$\quad\quad tempp \triangleq temp[p]$

$\quad\quad gamma \triangleq [t \in Point \mapsto \text{IF } t \in tt \text{ THEN } tempp[t] \text{ ELSE } 0]$

$\quad\quad newtempp \triangleq [t \in Point \mapsto \text{IF } t \in tt \text{ THEN } 0 \text{ ELSE } tempp[t]]$

$\quad \text{IN}$

$\quad \wedge\ gamma \neq DeltaVecZero$

$\quad \wedge\ IsDeltaVecUpright(lleq,\ newtempp)$

$\quad \wedge\ temp' = [temp \text{ EXCEPT } ![p] = newtempp]$

$\quad \wedge\ msg' = [msg \text{ EXCEPT } ![p] = [q \in Proc \mapsto Append(msg[p][q],\ gamma)]]$

$\quad \wedge\ \text{UNCHANGED } nrec$

$\quad \wedge\ \text{UNCHANGED } glob$

$\quad \wedge\ NextCommon$

Receive an update.

$NextReceiveUpdate \triangleq$

$\quad \exists\, p \in Proc :$

$\quad \exists\, q \in Proc :$

$\quad \text{LET}$

$\quad\quad kappa \triangleq Head(msg[p][q])$

$\quad \text{IN}$

$\quad \wedge\ msg[p][q] \neq \langle\rangle$

$\quad \wedge\ glob' = [glob \text{ EXCEPT } ![q] = DeltaVecAdd(glob[q],\ kappa)]$

$\quad \wedge\ msg' = [msg \text{ EXCEPT } ![p][q] = Tail(msg[p][q])]$

$\quad \wedge\ \text{UNCHANGED } nrec$

$\quad \wedge\ \text{UNCHANGED } temp$

$\quad \wedge\ NextCommon$

Specification.

$Init \triangleq$

Any point relation that is a partial order.

$\wedge\ lleq \in PointRelationType$
$\wedge\ IsPartialOrder(lleq)$

Any initial record-point arrangement.

$\wedge\ nrec \in PointToNat$

Initial values.

$\wedge\ glob \ = [p \in Proc \mapsto nrec]$
$\wedge\ temp = [p \in Proc \mapsto DeltaVecZero]$
$\wedge\ msg = [p \in Proc \mapsto [q \in Proc \mapsto \langle\rangle]]$

Initial fiducial variables based on initial state.

$\wedge\ nrecvut = [ft \in Point \mapsto NrecVacantUpto(ft)]$
$\wedge\ globvut = [fp \in Proc \mapsto [ft \in Point \mapsto GlobVacantUpto(fp, ft)]]$

$Next \triangleq$

Any action.

$\vee\ NextPerformOperation$
$\vee\ NextSendUpdate$
$\vee\ NextReceiveUpdate$

$Spec \triangleq Init \wedge \Box[Next]_{vars}$

Invariants.

Only a finite number of processors have information in $temp$.

$IsFiniteTempProcs \triangleq$
$\quad IsFiniteSet(\{p \in Proc : temp[p] \neq DeltaVecZero\})$

Only a finite number of processors sending messages to any q.

$IsFiniteMsgSenders \triangleq$
$\quad \forall q \in Proc :$
$\quad IsFiniteSet(\{p \in Proc : msg[p][q] \neq \langle\rangle\})$

Invariant: State variables have the correct type.

$InvType \triangleq$
$\quad \land\ lleq \in PointRelationType$
$\quad \land\ nrec \in CountVecType$
$\quad \land\ glob \in [Proc \rightarrow DeltaVecType]$
$\quad \land\ temp \in [Proc \rightarrow DeltaVecType]$
$\quad \land\ msg \in [Proc \rightarrow [Proc \rightarrow Seq(DeltaVecType)]]$
$\quad \land\ nrecvut \in [Point \rightarrow \text{BOOLEAN}]$
$\quad \land\ globvut \in [Proc \rightarrow [Point \rightarrow \text{BOOLEAN}]]$
$\quad \land\ IsPartialOrder(lleq)$
$\quad \land\ IsFiniteTempProcs$
$\quad \land\ IsFiniteMsgSenders$

Invariant: For all processors p, $temp[p]$ is an upright delta vector.

$InvTempUpright \triangleq$
$\quad \forall p \in Proc :$
$\quad IsDeltaVecUpright(lleq, temp[p])$

Invariant: For all processors p and q, the sum of all information about updates performed by p which is incoming at q, after skipping the first k updates on the message queue, is an upright delta vector.

Note that $IncomingInfo(k, p, q)$ sums up the delta vectors on the message queue from p to q skipping the first k, then adds $temp[p]$. Taking $k = 0$ includes all delta vectors on the message queue in the sum.

$InvIncomingInfoUpright \triangleq$
$\quad \forall k \in Nat :$
$\quad \forall p \in Proc :$
$\quad \forall q \in Proc :$
$\quad IsDeltaVecUpright(lleq, IncomingInfo(k, p, q))$

Invariant: For all processors q the sum of all information incoming at q, except for skipping the first k updates on the message queue from processor p, is an upright delta vector.

Note that $GlobalIncomingInfo(0, q, q)$ sums up all of the information incoming at q.

$InvGlobalIncomingInfoUpright \triangleq$
$\quad \forall k \in Nat :$
$\quad \forall p \in Proc :$
$\quad \forall q \in Proc :$
$\quad IsDeltaVecUpright(lleq,\ GlobalIncomingInfo(k,\ p,\ q))$

Invariant: For all processors q, the sum of all information incoming at q, plus $glob[q]$, equals $nrec$.

$InvGlobalRecordCount \triangleq$
$\quad \forall q \in Proc :$
$\quad nrec = DeltaVecAdd(GlobalIncomingInfo(0,\ q,\ q),\ glob[q])$

Invariant: For all processors q and points t, whenever all points s up thru t have zero in $glob[q]$, then all points s up thru t have no records.

$InvGlobVacantUptoImpliesNrec \triangleq$
$\quad \forall q \in Proc :$
$\quad \forall t \in Point :$
$\quad GlobVacantUpto(q,\ t) \Rightarrow NrecVacantUpto(t)$

Safety property: For all points t, if $NrecVacantUpto(t)$ is TRUE, then it will stay TRUE.

$SafeStickyNrecVacantUpto \triangleq$
$\quad \forall t \in Point :$
$\quad NrecVacantUpto(t) \Rightarrow \Box NrecVacantUpto(t)$

Unfortunately, the TLC model checker runs much more slowly when trying to check that a general temporal property always holds, since it has to keep additional information about the full state graph. TLC works much better if formulas can be written as an invariant (a simple state predicate) that holds in every reachable state. This is the purpose of the fiducial variable "$nrecvut$".

$Init$ sets $nrecvut[t] = NrecVacantUpto(t)$. The $Next$ actions set $nrecvut[t]' = NrecVacantUpto(t)$. (Note the absence of a trailing prime.) This makes $nrecvut[t]$ the value of $NrecVacantUpto(t)$ from the previous state, permitting us to check that $NrecVacantUpto(t) \Rightarrow NrecVacantUpto(t)'$. Hence we have the following invariant.

Invariant: For all points t, $nrecvut[t]$ is sticky.

$InvStickyNrecVacantUpto \triangleq$
$\quad \forall t \in Point :$
$\quad nrecvut[t] \Rightarrow NrecVacantUpto(t)$

Safety property: For all processors q and points t, if $GlobVacantUpto(q,\ t)$ is TRUE, then it will stay TRUE.

$SafeStickyGlobVacantUpto \triangleq$
$\quad \forall q \in Proc :$
$\quad \forall t \in Point :$
$\quad GlobVacantUpto(q,\ t) \Rightarrow \Box GlobVacantUpto(q,\ t)$

Unfortunately, the TLC model checker runs much more slowly when trying to check that a general temporal property always holds, since it has to keep additional information about the full state graph. TLC works much better if formulas can be written as an invariant (a simple state predicate) that holds in every reachable state. This is the purpose of the fiducial variable "$globvut$".

$Init$ sets $globvut[q][t] = GlobVacantUpto(q, t)$. The $Next$ actions set $globvut[q][t]' = GlobVacantUpto(q, t)$. (Note the absence of a trailing prime.) This makes $globvut[q][t]$ the value of $GlobVacantUpto(q, t)$ from the previous state, permitting us to check that $GlobVacantUpto(q, t)$ is sticky. Hence we have the following invariant.

Invariant: For all processors q and points t, $globvut[q][t]$ is sticky.

$$
\begin{aligned}
&InvStickyGlobVacantUpto \triangleq \\
&\quad \forall\, q \in Proc : \\
&\quad \forall\, t \in Point : \\
&\quad globvut[q][t] \Rightarrow GlobVacantUpto(q, t)
\end{aligned}
$$

Safety property: For all processors q and points t, whenever all points s up thru t have zero in $glob[q]$, then all points s up thru t have no records and never will in any following state.

$$
\begin{aligned}
&SafeGlobVacantUptoImpliesStickyNrec \triangleq \\
&\quad \forall\, q \in Proc : \\
&\quad \forall\, t \in Point : \\
&\quad GlobVacantUpto(q, t) \Rightarrow \Box NrecVacantUpto(t)
\end{aligned}
$$

Appendix B

Model

MODULE *NaiadClockModel*

EXTENDS *Naturals*, *Sequences*

VARIABLE *lleq* — the precedence order between points
VARIABLE *nrec* — how many records at each point
VARIABLE *glob* — global count for each processor and point
VARIABLE *temp* — temporary count for each processor and point
VARIABLE *msg* — message queues between processors
VARIABLE *nrecvut* — in prev state *nrec* was vacant at all points *lleq* t
VARIABLE *globvut* — in prev state *glob* was vacant at all points *lleq* t

Default configuration parameters.

$MaxProc \triangleq 2$ — number of processors
$MaxPoint \triangleq 3$ — number of points
$MaxRecPerPoint \triangleq 1$ — max records/point in *nrec*
$MaxRec \triangleq 2$ — max total records in *nrec*
$MaxMsgPerQueue \triangleq 1$ — max length of any message queue

$Proc \triangleq 1 \,..\, MaxProc$
$Point \triangleq 1 \,..\, MaxPoint$

Sum up records for all points in m.

$Sum(m) \triangleq$
  LET RECURSIVE $S(\_)$ $S(T) \triangleq$
    IF $T = \{\}$ THEN 0 ELSE LET $t \triangleq$ CHOOSE $t \in T$: TRUE IN $m[t] + S(T \setminus \{t\})$
  IN $S(Point)$

$PointToNat \triangleq [Point \rightarrow 0 \,..\, MaxRecPerPoint]$

INSTANCE *NaiadClock*

$Constraint \triangleq$

$\land Sum(nrec) \leq MaxRec$
$\land \forall t \in Point : nrec[t] \leq MaxRecPerPoint$
$\land \forall p, q \in Proc : Len(msg[p][q]) \leq MaxMsgPerQueue$

$ModelExactSeqFor(S) \triangleq$
LET RECURSIVE $Q(\_)Q(SS) \triangleq$
IF $SS = \{\}$ THEN $\langle\rangle$ ELSE
LET $s \triangleq$ CHOOSE $s \in SS$: TRUEIN
$Append(Q(SS \setminus \{s\}), s)$
IN
$Q(S)$

Appendix C

Proof of Correctness

C.1 Basic additional definitions

MODULE $NaiadClockProofBase$

EXTENDS $NaiadClock$, $NaturalsInduction$, $TLAPS$

Basic additional definitions.

Make a new sequence by removing the element at index n from sequence q.

$$RemoveAt(q, n) \triangleq$$
$$[i \in 1 \,..\, Len(q) - 1 \mapsto \text{IF } i < n \text{ THEN } q[i] \text{ ELSE } q[i+1]]$$

For the proof we need to know what delta vector information is at position k on the message queue from processor p to processor q. For convenience, we define the information as $DeltaVecZero$ when position k falls outside the domain of the message queue.

$$InfoAt(k, p, q) \triangleq$$
$$\text{LET}$$
$$M \triangleq msg[p][q]$$
$$LenM \triangleq Len(M)$$
$$\text{IN}$$
$$\text{IF } 0 < k \land k \leq LenM \text{ THEN } M[k] \text{ ELSE } DeltaVecZero$$

A delta vector va is vb-upright iff for every positive point t in va there is a strictly lower point s that is negative in va or in vb and no point at s or yet lower is positive in va.

$$IsDeltaVecBetaUpright(leq, va, vb) \triangleq$$
$$\text{LET}$$
$$a \preceq b \triangleq leq[a][b]$$
$$a \prec b \triangleq a \preceq b \land a \neq b$$
$$\text{IN}$$

$\forall t \in Point :$ — for any point

$va[t] > 0$ — that is positive in va

$\Rightarrow$

$\exists s \in Point :$ — there is a point

$\land\ s \prec t$ — strictly lower

$\land\ (va[s] < 0 \lor vb[s] < 0)$ — that is negative in va or vb

$\land\ IsDeltaVecNonposUpto(leq,\ va,\ s)$ and va is nonpos up thru s

We say that delta vector $vecsrc$ positive implies delta vector $vecdst$ iff for every point t such that $vecsrc[t]$ is positive, $vecdst[t]$ is positive.

$IsDeltaVecPositiveImplies(vecsrc,\ vecdst)\ \triangleq$
$\quad\land\ vecsrc \in DeltaVecType$
$\quad\land\ vecdst \in DeltaVecType$
$\quad\land\ \forall\, t \in Point : vecsrc[t] > 0 \Rightarrow vecdst[t] > 0$

Definitions for the part of each action after its existential variables have been bound. Using these definitions permits the complicated expressions defining the actions to be hidden from the prover until needed.

$NextPerformOperation\_WithPCR(p,\ c,\ r)\ \triangleq\ NextPerformOperation\,!(p)!(c)!(r)$
$NextSendUpdate\_WithPTT(p,\ tt)\ \triangleq\ NextSendUpdate\,!(p)!(tt)$
$NextReceiveUpdate\_WithPQ(p,\ q)\ \triangleq\ NextReceiveUpdate\,!(p)!(q)$

Definitions for an important value within each action. Using these definitions permits the complicated expressions defining these values to be hidden from the prover until needed.

$NextPerformOperation\_Delta(p,\ c,\ r)\ \triangleq\ NextPerformOperation\,!(p)!(c)!(r)!delta$
$NextSendUpdate\_Gamma(p,\ tt)\ \triangleq\ NextSendUpdate\,!(p)!(tt)!gamma$
$NextReceiveUpdate\_Kappa(p,\ q)\ \triangleq\ NextReceiveUpdate\,!(p)!(q)!kappa$

Definitions for the LET locals inside the definition of $GlobalIncomingInfo$. Using these definitions permits the complicated expressions defining these values to be hidden from the prover until needed.

$GlobalIncomingInfo\_F(k,\ p,\ q)\ \triangleq\ GlobalIncomingInfo(k,\ p,\ q)!:!F$

C.2 Facts about naturals

MODULE $NaiadClockProofNaturals$

EXTENDS $NaiadClockProofBase$

Facts about naturals.

This really ought to be a library of theorems.

Dot dot facts.

THEOREM $DotDotDef \triangleq \forall i, m, n \in Nat : (m \leq i \land i \leq n) \equiv i \in m \,..\, n$ BY $SMTT(10)$
THEOREM $DotDotType \triangleq \forall m, n \in Nat : m \,..\, n \subseteq Nat$ BY $SMTT(10)$
THEOREM $DotDotType2 \triangleq \forall m, n \in Nat : \forall i \in m \,..\, n : i \in Nat$ BY $SMTT(10)$

$1 \,..\, n$ is equivalent to n itself for $n \in Nat$.

THEOREM $DotDotOneThruN \triangleq$
  $\forall m, n \in Nat : 1 \,..\, m = 1 \,..\, n \equiv m = n$
PROOF
  $\langle 1 \rangle 1.$ SUFFICES ASSUME NEW $m \in Nat$, NEW $n \in Nat$, $m \neq n$ PROVE $1 \,..\, m \neq 1 \,..\, n$ OBVIOUS

  Without loss of generality, assume ma is smaller than na.

  $\langle 1 \rangle$ DEFINE $ma \triangleq$ IF $m < n$ THEN m ELSE n
  $\langle 1 \rangle$ DEFINE $na \triangleq$ IF $m < n$ THEN n ELSE m
  $\langle 1 \rangle 2.$ $ma \in Nat$ OBVIOUS
  $\langle 1 \rangle 3.$ $na \in Nat$ OBVIOUS
  $\langle 1 \rangle 4.$ $ma < na$
    $\langle 2 \rangle 1.$ CASE $m < n$ BY $\langle 2 \rangle 1$, $\langle 1 \rangle 1$, $SMTT(10)$
    $\langle 2 \rangle 2.$ CASE $\neg(m < n)$ BY $\langle 2 \rangle 2$, $\langle 1 \rangle 1$, $SMTT(10)$
    $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$
  $\langle 1 \rangle$ SUFFICES $1 \,..\, ma \neq 1 \,..\, na$ BY $SMTT(10)$
  $\langle 1 \rangle$ HIDE DEF ma, na

  na shows that the ranges differ.

  $\langle 1 \rangle 5.$ $0 < na$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $SMTT(10)$
  $\langle 1 \rangle 6.$ $1 \leq na$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 5$, $SMTT(10)$
  $\langle 1 \rangle 7.$ $na \in 1 \,..\, na$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 6$, $SMTT(10)$

$\langle 1 \rangle 8.$ $na \notin 1 \,..\, ma$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $SMTT(10)$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 7$, $\langle 1 \rangle 8$

Any non-empty subset of Nat has a minimum element. You would think this would be a library theorem, but I could not find it. We use the classic inductive proof by contradiction for this theorem.

THEOREM $NatWellFounded \triangleq$
$\quad \forall N \in$ SUBSET $Nat : N \neq \{\} \Rightarrow \exists n \in N : \forall m \in N : n \leq m$
PROOF
$\langle 1 \rangle 1.$ SUFFICES ASSUME NEW $N \in$ SUBSET Nat, $N \neq \{\}$
PROVE $\exists n \in N : \forall m \in N : n \leq m$
OBVIOUS

Assuming that no minimum element of N exists, we prove that N must be empty, which is a contradiction.

$\langle 1 \rangle 2.$ SUFFICES ASSUME $\neg \exists n \in N : \forall m \in N : n \leq m$ PROVE $N = \{\}$ BY $\langle 1 \rangle 1$

$P(i)$ says that no naturals less than i are in N. We prove this for all i in Nat using induction.

$\langle 1 \rangle$ DEFINE $P(i) \triangleq \forall k \in Nat : k < i \Rightarrow k \notin N$
$\langle 1 \rangle 3.$ $\forall i \in Nat : P(i)$
$\langle 2 \rangle 1.$ $P(0)$ BY $SMTT(10)$
$\langle 2 \rangle 2.$ $\forall i \in Nat : P(i) \Rightarrow P(i+1)$
$\langle 3 \rangle 1.$ SUFFICES ASSUME NEW $i \in Nat$, $P(i)$ PROVE $P(i+1)$ OBVIOUS
$\langle 3 \rangle 2.$ SUFFICES ASSUME NEW $k \in Nat$, $k < i+1$ PROVE $k \notin N$ OBVIOUS
$\langle 3 \rangle 3.$ CASE $k < i$ BY $\langle 3 \rangle 1$, $\langle 3 \rangle 3$
$\langle 3 \rangle 4.$ CASE $k = i$
$\langle 4 \rangle 1.$ SUFFICES ASSUME $k \in N$ PROVE FALSE OBVIOUS
$\langle 4 \rangle 2.$ $\forall j \in N : k \leq j$ BY $\langle 3 \rangle 1$, $\langle 3 \rangle 4$, $SMTT(10)$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 1 \rangle 2$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$, $\langle 3 \rangle 3$, $\langle 3 \rangle 4$, $SMTT(10)$
$\langle 2 \rangle$ HIDE DEF P
$\langle 2 \rangle$ QED BY ONLY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $NatInduction$, Isa

Since $P(i)$ is true for all i in Nat, N must be empty.

$\langle 1 \rangle 4.$ $\forall i \in Nat : i \notin N$
$\langle 2 \rangle$ SUFFICES ASSUME NEW $i \in Nat$ PROVE $i \notin N$ OBVIOUS
$\langle 2 \rangle 1.$ $i + 1 \in Nat$ BY $SMTT(10)$
$\langle 2 \rangle 2.$ $P(i+1)$ BY $\langle 2 \rangle 1$, $\langle 1 \rangle 3$
$\langle 2 \rangle 3.$ $i < i+1$ BY $SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 4$

C.3 Facts about sequences

MODULE $NaiadClockProofSequences$

EXTENDS $NaiadClockProofNaturals$

Facts about sequences.

This really ought to be a library of theorems.

The following definitions are essentially copied from the standard Sequences module. We could prove them if the definitions of Seq, Len, Head, $Tail$, and $Append$ could be expanded, but unfortunately the current proof system does not permit this.

THEOREM $SeqDef \triangleq \forall S : Seq(S) = \text{UNION } \{[1 \,..\, n \rightarrow S] : n \in Nat\}$
THEOREM $LenDef \triangleq \forall S : \forall seq \in Seq(S) : \text{DOMAIN } seq = 1 \,..\, Len(seq)$
THEOREM $HeadDef \triangleq \forall seq : Head(seq) = seq[1]$
THEOREM $TailDef \triangleq \forall seq : Tail(seq) = [i \in 1 \,..\, (Len(seq) - 1) \mapsto seq[i + 1]]$

THEOREM $AppendDef \triangleq$
$\forall seq, elt :$
$Append(seq, elt) = [i \in 1 \,..\, (Len(seq) + 1) \mapsto$
$\text{IF } i \leq Len(seq) \text{ THEN } seq[i] \text{ ELSE } elt]$

Prove that $q \in Seq(S)$.

For some reason, the provers find it difficult to deduce this from the given predicates using just $SeqDef$, so it helps to prove it once here.

THEOREM $IsASeq \triangleq$
ASSUME
NEW S,
NEW $n \in Nat$,
NEW $q \in [1 \,..\, n \rightarrow S]$
PROVE
$q \in Seq(S)$
PROOF
$\langle 1 \rangle$ QED BY $SeqDef$, $IsaT(120)$

Axiom about Len.

THEOREM $LenAxiom \triangleq$
  $\forall S :$
  $\forall seq \in Seq(S) :$
  $Len(seq) \in Nat \land seq \in [1 \,..\, Len(seq) \to S]$
PROOF
  $\langle 1 \rangle 1.$ SUFFICES ASSUME
      NEW S,
      NEW $q \in Seq(S)$
    PROVE
    $\land\ Len(q) \in Nat$
    $\land\ q \in [1 \,..\, Len(q) \to S]$
    OBVIOUS

  $\langle 1 \rangle 2.$ $Len(q) \in Nat$ BY Isa isabelle knows this axiomatically

  $\langle 1 \rangle 3.$ $q \in [1 \,..\, Len(q) \to S]$
    $\langle 2 \rangle 1.$ DOMAIN $q = 1 \,..\, Len(q)$ BY $LenDef$
    $\langle 2 \rangle 2.$ $\exists\, n \in Nat : q \in [1 \,..\, n \to S]$ BY $SeqDef$, Isa
    $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$

  $\langle 1 \rangle$ QED BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$

The length of a sequence is a natural number.

COROLLARY $LenInNat \triangleq$
  $\forall S : \forall seq \in Seq(S) : Len(seq) \in Nat$
PROOF
  BY $LenAxiom$

When the domain of a sequence is $1 \,..\, n$, then n is the length of the sequence.

THEOREM $LenDomain \triangleq$
  $\forall S :$

$\forall\, seq \in Seq(S) :$
$\forall\, n \in Nat :$
DOMAIN $seq = 1 \,..\, n \Rightarrow n = Len(seq)$
PROOF
$\langle 1 \rangle 1.$ SUFFICES ASSUME
NEW S,
NEW $q \in Seq(S)$,
NEW $n \in Nat$,
DOMAIN $q = 1 \,..\, n$
PROVE $n = Len(q)$
OBVIOUS

$\langle 1 \rangle 2.$ $Len(q) \in Nat$ BY $LenAxiom$
$\langle 1 \rangle 3.$ DOMAIN $q = 1 \,..\, Len(q)$ BY $LenAxiom$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $DotDotOneThruN$

The element of a $Seq(S)$ is in S.

THEOREM $ElementOfSeq \triangleq$
$\forall\, S :$
$\forall\, seq \in Seq(S) :$
$\forall\, n \in 1 \,..\, Len(seq) :$
$seq[n] \in S$
PROOF
$\langle 1 \rangle 1.$ SUFFICES ASSUME
NEW S,
NEW $q \in Seq(S)$,
NEW $n \in 1 \,..\, Len(q)$
PROVE $q[n] \in S$
OBVIOUS

$\langle 1 \rangle 2.$ $q \in [1 \,..\, Len(q) \rightarrow S]$ BY $LenAxiom$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2$

Properties of the empty sequence.

THEOREM $EmptySeq \triangleq$

$\forall S :$
$\land \langle\rangle \in Seq(S)$
$\land \forall seq \in Seq(S) : (seq = \langle\rangle) \equiv (Len(seq) = 0)$

PROOF

$\langle 1 \rangle 1.$ ASSUME NEW S
PROVE $\langle\rangle \in Seq(S)$
BY Isa isabelle knows this axiomatically

$\langle 1 \rangle 2.$ ASSUME NEW S, NEW $q \in Seq(S)$
PROVE $(q = \langle\rangle) \equiv (Len(q) = 0)$
BY Isa isabelle knows this axiomatically

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$

The empty sequence is a sequence.

COROLLARY $EmptySeqIsASeq \triangleq$
$\forall S : \langle\rangle \in Seq(S)$

PROOF
BY $EmptySeq$

An empty sequence has length zero.

THEOREM $LenEmptyIsZero \triangleq$
$Len(\langle\rangle) = 0$

PROOF
OBVIOUS

The head of a non-empty $Seq(S)$ is an S.

THEOREM $HeadType \triangleq$
$\forall S : \forall q \in Seq(S) : q \neq \langle\rangle \Rightarrow Head(q) \in S$

PROOF
$\langle 1 \rangle 1.$ SUFFICES ASSUME
NEW S,
NEW $q \in Seq(S)$,
$q \neq \langle\rangle$
PROVE $Head(q) \in S$
OBVIOUS

$\langle 1 \rangle$ DEFINE $n \triangleq Len(q)$
$\langle 1 \rangle$ HIDE DEF n

$\langle 1 \rangle$ SUFFICES $1 \in 1 \mathrel{..} n$ BY $\langle 1 \rangle 1$, $HeadDef$, $ElementOfSeq$ DEF n
$\langle 1 \rangle 2.$ $n \neq 0$ BY $\langle 1 \rangle 1$, $EmptySeq$ DEF n
$\langle 1 \rangle 3.$ $n \in Nat$ BY $LenAxiom$ DEF n
$\langle 1 \rangle 4.$ $n > 0$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $SMTT(10)$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $SMTT(10)$

Properties of $Tail$.

THEOREM $TailProp \triangleq$
$\forall S :$
$\forall seq \in Seq(S) :$
$seq \neq \langle\rangle$
$\Rightarrow$
$\wedge\ Tail(seq) \in Seq(S)$
$\wedge\ Len(Tail(seq)) = Len(seq) - 1$
$\wedge\ \forall i \in 1 \mathrel{..} Len(Tail(seq)) :$
$\quad \wedge\ i + 1 \in 1 \mathrel{..} Len(seq)$
$\quad \wedge\ Tail(seq)[i] = seq[i + 1]$
PROOF
$\langle 1 \rangle 1.$ SUFFICES ASSUME
NEW S,
NEW $q \in Seq(S)$,
$q \neq \langle\rangle$
PROVE
$\wedge\ Tail(q) \in Seq(S)$
$\wedge\ Len(Tail(q)) = Len(q) - 1$
$\wedge\ \forall i \in 1 \mathrel{..} Len(Tail(q)) :$
$\quad \wedge\ i + 1 \in 1 \mathrel{..} Len(q)$
$\quad \wedge\ Tail(q)[i] = q[i + 1]$
OBVIOUS

$\langle 1 \rangle$ DEFINE $n \triangleq Len(q)$
$\langle 1 \rangle$ DEFINE $m \triangleq n - 1$
$\langle 1 \rangle$ HIDE DEF $n,\ m$

$\langle 1 \rangle 2.$ $n \in Nat$ BY $LenInNat$ DEF n
$\langle 1 \rangle 3.$ $n \neq 0$ BY $\langle 1 \rangle 1,\ EmptySeq$ DEF n
$\langle 1 \rangle 4.$ $m \in Nat$ BY $\langle 1 \rangle 2,\ \langle 1 \rangle 3,\ SMTT(10)$ DEF m

$\langle 1 \rangle 5.$ $q \in [1 \,..\, n \rightarrow S]$ BY $LenAxiom$ DEF n
$\langle 1 \rangle 6.$ $Tail(q) = [i \in 1 \,..\, m \mapsto q[i+1]]$ BY $TailDef$ DEF $n,\ m$

$\langle 1 \rangle 7.$ $Tail(q) \in Seq(S)$
  $\langle 2 \rangle 1.$ $[i \in 1 \,..\, m \mapsto q[i+1]] \in Seq(S)$
    $\langle 3 \rangle 1.$ ASSUME NEW $i \in 1 \,..\, m$ PROVE $q[i+1] \in S$
      $\langle 4 \rangle 1.$ $i + 1 \in 1 \,..\, n$ BY $\langle 3 \rangle 1,\ \langle 1 \rangle 2,\ \langle 1 \rangle 3,\ SMTT(10)$ DEF m
      $\langle 4 \rangle$ QED BY $\langle 4 \rangle 1,\ \langle 1 \rangle 5$
    $\langle 3 \rangle 2.$ $[i \in 1 \,..\, m \mapsto q[i+1]] \in [1 \,..\, m \rightarrow S]$ BY $\langle 3 \rangle 1$
    $\langle 3 \rangle$ QED BY $\langle 1 \rangle 4,\ \langle 3 \rangle 2,\ IsASeq$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 1 \rangle 6$

$\langle 1 \rangle 8.$ $Len(Tail(q)) = m$
  $\langle 2 \rangle 1.$ $Len(Tail(q)) \in Nat$ BY $\langle 1 \rangle 7,\ LenInNat$
  $\langle 2 \rangle 2.$ DOMAIN $Tail(q) = 1 \,..\, Len(Tail(q))$ BY $\langle 1 \rangle 7,\ LenDef$
  $\langle 2 \rangle 3.$ $1 \,..\, Len(Tail(q)) = 1 \,..\, m$ BY $\langle 2 \rangle 2,\ \langle 1 \rangle 6$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 3,\ \langle 1 \rangle 4,\ DotDotOneThruN$

$\langle 1 \rangle 9.$ ASSUME NEW $i \in 1 \,..\, m$ PROVE $i + 1 \in 1 \,..\, n$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 2,\ \langle 1 \rangle 3,\ \langle 1 \rangle 9,\ SMTT(10)$ DEF m

$\langle 1 \rangle 10.$ ASSUME NEW $i \in 1 \,..\, m$ PROVE $Tail(q)[i] = q[i+1]$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 6,\ \langle 1 \rangle 9$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 7,\ \langle 1 \rangle 8,\ \langle 1 \rangle 9,\ \langle 1 \rangle 10$ DEF $n,\ m$

The tail of a non-empty $Seq(S)$ is a $Seq(S)$.

COROLLARY $TailType \triangleq$
  $\forall S : \forall q \in Seq(S) : q \neq \langle\rangle \Rightarrow Tail(q) \in Seq(S)$
PROOF
  BY $TailProp$

Properties of $Append$.

THEOREM $AppendProperties \triangleq$
$\forall S :$
$\forall seq \in Seq(S),\ elt \in S :$
$\wedge\ Append(seq,\ elt) \in Seq(S)$
$\wedge\ Len(Append(seq,\ elt)) = Len(seq) + 1$
$\wedge\ \forall\, i \in 1\,..\,Len(seq) : Append(seq,\ elt)[i] = seq[i]$
$\wedge\ Append(seq,\ elt)[Len(seq) + 1] = elt$
PROOF
$\langle 1 \rangle 1.$ SUFFICES ASSUME
NEW S,
NEW $q \in Seq(S)$,
NEW $e \in S$
PROVE
$\wedge\ Append(q,\ e) \in Seq(S)$
$\wedge\ Len(Append(q,\ e)) = Len(q) + 1$
$\wedge\ \forall\, i \in 1\,..\,Len(q) : Append(q,\ e)[i] = q[i]$
$\wedge\ Append(q,\ e)[Len(q) + 1] = e$
OBVIOUS

$\langle 1 \rangle$ DEFINE $n \triangleq Len(q)$
$\langle 1 \rangle$ DEFINE $m \triangleq n + 1$
$\langle 1 \rangle$ HIDE DEF $n,\ m$

$\langle 1 \rangle 2.$ $n \in Nat$ BY $LenInNat$ DEF n
$\langle 1 \rangle 3.$ $m \neq 0$ BY $\langle 1 \rangle 2,\ SMTT(10)$ DEF m
$\langle 1 \rangle 4.$ $m \in Nat$ BY $\langle 1 \rangle 2,\ SMTT(10)$ DEF m

$\langle 1 \rangle 5.$ $q \in [1\,..\,n \rightarrow S]$ BY $LenAxiom$ DEF n
$\langle 1 \rangle 6.$ $Append(q,\ e) = [i \in 1\,..\,m \mapsto$ IF $i \leq n$ THEN $q[i]$ ELSE $e]$ BY $AppendDef$ DEF $n,\ m$

$\langle 1 \rangle 7.$ $Append(q,\ e) \in Seq(S)$
$\langle 2 \rangle 1.$ ASSUME NEW $i \in 1\,..\,m$ PROVE $Append(q,\ e)[i] \in S$
$\langle 3 \rangle 1.$ CASE $i \leq n$
$\langle 4 \rangle 1.$ $i \in 1\,..\,n$ BY $\langle 3 \rangle 1,\ \langle 1 \rangle 2,\ \langle 1 \rangle 4,\ SMTT(10)$
$\langle 4 \rangle 2.$ $Append(q,\ e)[i] = q[i]$ BY $\langle 3 \rangle 1,\ \langle 1 \rangle 6$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1,\ \langle 4 \rangle 2,\ \langle 1 \rangle 5$
$\langle 3 \rangle 2.$ CASE $\neg(i \leq n)$
$\langle 4 \rangle 1.$ $Append(q,\ e)[i] = e$ BY $\langle 3 \rangle 2,\ \langle 1 \rangle 6$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 3 \rangle 2$
$\langle 2 \rangle 2.$ $Append(q,\ e) \in [1\,..\,m \rightarrow S]$ BY $\langle 2 \rangle 1,\ \langle 1 \rangle 6$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2,\ \langle 1 \rangle 4,\ IsASeq$

$\langle 1 \rangle 8.$ $Len(Append(q,\ e)) = m$
  $\langle 2 \rangle 1.$ $Len(Append(q,\ e)) \in Nat$ BY $\langle 1 \rangle 7,\ LenInNat$
  $\langle 2 \rangle 2.$ DOMAIN $Append(q,\ e) = 1 \,..\, Len(Append(q,\ e))$ BY $\langle 1 \rangle 7,\ LenDef$
  $\langle 2 \rangle 3.$ $1 \,..\, Len(Append(q,\ e)) = 1 \,..\, m$ BY $\langle 2 \rangle 2,\ \langle 1 \rangle 6$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 3,\ \langle 1 \rangle 4,\ DotDotOneThruN$

$\langle 1 \rangle 9.$ ASSUME NEW $i \in 1 \,..\, n$ PROVE $Append(q,\ e)[i] = q[i]$
  $\langle 2 \rangle 1.$ $i \leq n$ BY $\langle 1 \rangle 2,\ \langle 1 \rangle 9,\ SMTT(10)$
  $\langle 2 \rangle 2.$ $i \in 1 \,..\, m$ BY $\langle 1 \rangle 2,\ \langle 1 \rangle 9,\ SMTT(10)$ DEF m
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 2,\ \langle 1 \rangle 6$

$\langle 1 \rangle 10.$ $Append(q,\ e)[m] = e$
  $\langle 2 \rangle 1.$ $m \in 1 \,..\, m$ BY $\langle 1 \rangle 3,\ \langle 1 \rangle 4,\ SMTT(10)$
  $\langle 2 \rangle 2.$ $\neg(m \leq n)$ BY $\langle 1 \rangle 2,\ SMTT(10)$ DEF m
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 2,\ \langle 1 \rangle 6$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 7,\ \langle 1 \rangle 8,\ \langle 1 \rangle 9,\ \langle 1 \rangle 10$ DEF $n,\ m$

The elements at positions $1 \,..\, Len(Q)$ are unchanged by appending a new element to Q.

This is a trivial corollary of $AppendProperties$, but it is difficult for the provers to conclude it in some contexts. So we make it explicit.

COROLLARY $AppendPropertiesOldElems \triangleq$
  ASSUME
    NEW S,
    NEW $Q \in Seq(S)$,
    NEW $s \in S$,
    NEW $i \in 1 \,..\, Len(Q)$
  PROVE $Append(Q,\ s)[i] = Q[i]$
PROOF
  BY $Isa,\ AppendProperties$

The element at position $Len(Q) + 1$ in $Append(Q,\ s)$ is s.

This is a trivial corollary of $AppendProperties$, but it is difficult for the provers to conclude it in some contexts. So we make it explicit.

COROLLARY $AppendPropertiesNewElem \triangleq$
  ASSUME
    NEW S,

NEW $Q \in Seq(S)$,
NEW $s \in S$
PROVE $Append(Q, s)[Len(Q) + 1] = s$
PROOF
BY $AppendProperties$

$Q \in Seq(S)$ implies $Q \in Seq(T)$ for T any superset of S.

THEOREM $SeqSupset \triangleq$
ASSUME
NEW S,
NEW $Q \in Seq(S)$,
NEW T, $S \subseteq T$
PROVE
$Q \in Seq(T)$
PROOF
BY Isa

C.4 Properties of RemoveAt

MODULE $NaiadClockProofRemoveAt$

EXTENDS $NaiadClockProofSequences$

Properties of RemoveAt.

This really ought to be a library of theorems.

To make life easier for the theorem prover, we define operators for each of the complicated properties. These operators assume that we have $Q \in Seq(S)$ and $n \in 1 \,..\, Len(Q)$.

Mapping index qi forward from Q to $RemoveAt(Q, n)$.

$RemoveAt\_ForwardIndex(Q, n, qi) \triangleq$ IF $qi < n$ THEN qi ELSE $qi - 1$

Mapping index ri backward from $RemoveAt(Q, n)$ to Q.

$RemoveAt\_BackwardIndex(Q, n, ri) \triangleq$ IF $ri < n$ THEN ri ELSE $ri + 1$

How each index maps forward.

$RemoveAt\_MapForward(Q, n) \triangleq$
LET
$R \triangleq RemoveAt(Q, n)$
IN
$\forall qi \in 1 \,..\, Len(Q) :$
$qi \neq n$
$\Rightarrow$
LET
$ri \triangleq RemoveAt\_ForwardIndex(Q, n, qi)$
IN
$ri \in 1 \,..\, Len(R) \land Q[qi] = R[ri]$

How each index maps backward.

$RemoveAt\_MapBackward(Q, n) \triangleq$
LET
$R \triangleq RemoveAt(Q, n)$
IN
$\forall ri \in 1 \,..\, Len(R) :$

LET
$qi \triangleq RemoveAt\_BackwardIndex(Q, n, ri)$
IN
$qi \in 1 \mathrel{..} Len(Q) \land qi \neq n \land Q[qi] = R[ri]$

Each index maps forward.

$RemoveAt\_EachForward(Q, n) \triangleq$
LET
$R \triangleq RemoveAt(Q, n)$
IN
$\forall qi \in 1 \mathrel{..} Len(Q) :$
$qi \neq n$
$\Rightarrow$
$\exists ri \in 1 \mathrel{..} Len(R) :$
$Q[qi] = R[ri]$

Each index maps backward.

$RemoveAt\_EachBackward(Q, n) \triangleq$
LET
$R \triangleq RemoveAt(Q, n)$
IN
$\forall ri \in 1 \mathrel{..} Len(R) :$
$\exists qi \in 1 \mathrel{..} Len(Q) :$
$qi \neq n \land Q[qi] = R[ri]$

Indexes map forward preserving order.

$RemoveAt\_OrderedForward(Q, n) \triangleq$
LET
$R \triangleq RemoveAt(Q, n)$
IN
$\forall qi1, qi2 \in 1 \mathrel{..} Len(Q) :$
$qi1 < qi2 \land qi1 \neq n \land qi2 \neq n$
$\Rightarrow$
$\exists ri1, ri2 \in 1 \mathrel{..} Len(R) :$
$ri1 < ri2 \land Q[qi1] = R[ri1] \land Q[qi2] = R[ri2]$

Indexes map backward preserving order.

$RemoveAt\_OrderedBackward(Q, n) \triangleq$
LET
$R \triangleq RemoveAt(Q, n)$
IN
$\forall ri1, ri2 \in 1 \mathrel{..} Len(R) :$
$ri1 < ri2$

$\Rightarrow$
$\exists\, qi1,\; qi2 \in 1 \,..\, Len(Q) :$
$qi1 < qi2 \land qi1 \neq n \land qi2 \neq n \land Q[qi1] = R[ri1] \land Q[qi2] = R[ri2]$

Indexes map forward preserving distinctness.

$RemoveAt\_DistinctForward(Q,\; n) \;\triangleq$
LET
$R \;\triangleq\; RemoveAt(Q,\; n)$
IN
$\forall\, qi1,\; qi2 \in 1 \,..\, Len(Q) :$
$qi1 \neq qi2 \land qi1 \neq n \land qi2 \neq n$
$\Rightarrow$
$\exists\, ri1,\; ri2 \in 1 \,..\, Len(R) :$
$ri1 \neq ri2 \land Q[qi1] = R[ri1] \land Q[qi2] = R[ri2]$

Indexes map backward preserving distinctness.

$RemoveAt\_DistinctBackward(Q,\; n) \;\triangleq$
LET
$R \;\triangleq\; RemoveAt(Q,\; n)$
IN
$\forall\, ri1,\; ri2 \in 1 \,..\, Len(R) :$
$ri1 \neq ri2$
$\Rightarrow$
$\exists\, qi1,\; qi2 \in 1 \,..\, Len(Q) :$
$qi1 \neq qi2 \land qi1 \neq n \land qi2 \neq n \land Q[qi1] = R[ri1] \land Q[qi2] = R[ri2]$

The theorem.

THEOREM $RemoveAtProperties \;\triangleq$
ASSUME
NEW S,
NEW $Q \in Seq(S)$,
NEW $n \;\in 1 \,..\, Len(Q)$
PROVE
LET
$R \;\triangleq\; RemoveAt(Q,\; n)$
IN
$\land\, R \in Seq(S)$
$\land\, Len(R) = Len(Q) - 1$
$\land\, RemoveAt\_MapForward(Q,\; n)$
$\land\, RemoveAt\_MapBackward(Q,\; n)$
$\land\, RemoveAt\_EachForward(Q,\; n)$

$\wedge$ $RemoveAt\_EachBackward(Q, n)$
$\wedge$ $RemoveAt\_OrderedForward(Q, n)$
$\wedge$ $RemoveAt\_OrderedBackward(Q, n)$
$\wedge$ $RemoveAt\_DistinctForward(Q, n)$
$\wedge$ $RemoveAt\_DistinctBackward(Q, n)$

PROOF
$\langle 1 \rangle$ USE DEF $RemoveAt\_ForwardIndex$
$\langle 1 \rangle$ USE DEF $RemoveAt\_BackwardIndex$

$\langle 1 \rangle$ DEFINE $R \triangleq RemoveAt(Q, n)$
$\langle 1 \rangle$ DEFINE $LenQ \triangleq Len(Q)$
$\langle 1 \rangle$ DEFINE $LenR \triangleq Len(R)$
$\langle 1 \rangle$ HIDE DEF $R, LenQ, LenR$

Prove that $R \in Seq(S)$.

$\langle 1 \rangle 1.$ $Q \in [1 \,..\, LenQ \rightarrow S]$ BY $LenAxiom$ DEF $LenQ$
$\langle 1 \rangle 2.$ $n \in 1 \,..\, LenQ$ BY DEF $LenQ$
$\langle 1 \rangle 3.$ $LenQ \in Nat$ BY $LenInNat$ DEF $LenQ$
$\langle 1 \rangle 4.$ $LenQ > 0$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $SMTT(10)$
$\langle 1 \rangle 5.$ $LenQ - 1 \in Nat$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $SMTT(10)$
$\langle 1 \rangle 6.$ $R = [ri \in 1 \,..\, LenQ - 1 \mapsto Q[RemoveAt\_BackwardIndex(Q, n, ri)]]$
BY DEF $R, LenQ, RemoveAt$
$\langle 1 \rangle 7.$ $\forall i \in 1 \,..\, LenQ - 1 : R[i] \in S$
$\langle 2 \rangle 1.$ SUFFICES ASSUME NEW i, $i \in 1 \,..\, LenQ - 1$ PROVE $R[i] \in S$ OBVIOUS
$\langle 2 \rangle 2.$ CASE $i < n$
$\langle 3 \rangle 1.$ $i \in 1 \,..\, LenQ$ BY $\langle 2 \rangle 1$, $\langle 1 \rangle 3$, $SMTT(10)$
$\langle 3 \rangle 2.$ $R[i] = Q[i]$ BY $\langle 1 \rangle 6$, $\langle 2 \rangle 1$, $\langle 2 \rangle 2$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 1 \rangle 1$
$\langle 2 \rangle 3.$ CASE $\neg(i < n)$
$\langle 3 \rangle 1.$ $i + 1 \in 1 \,..\, LenQ$ BY $\langle 2 \rangle 1$, $\langle 1 \rangle 3$, $SMTT(10)$
$\langle 3 \rangle 2.$ $R[i] = Q[i + 1]$ BY $\langle 1 \rangle 6$, $\langle 2 \rangle 1$, $\langle 2 \rangle 3$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 1 \rangle 1$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$
$\langle 1 \rangle 8.$ $R \in [1 \,..\, LenQ - 1 \rightarrow S]$ BY $\langle 1 \rangle 6$, $\langle 1 \rangle 7$
$\langle 1 \rangle 9.$ $R \in Seq(S)$ BY $\langle 1 \rangle 5$, $\langle 1 \rangle 8$, $IsASeq$

Prove that $Len(R) = Len(Q) - 1$.

$\langle 1 \rangle 10.$ DOMAIN $R = 1 \,..\, LenQ - 1$ BY $\langle 1 \rangle 8$
$\langle 1 \rangle 11.$ DOMAIN $R = 1 \,..\, LenR$ BY $\langle 1 \rangle 9$, $LenDef$ DEF $LenR$
$\langle 1 \rangle 12.$ $LenR \in Nat$ BY $\langle 1 \rangle 9$, $LenAxiom$ DEF $LenR$
$\langle 1 \rangle 13.$ $LenR = LenQ - 1$ BY $\langle 1 \rangle 5$, $\langle 1 \rangle 10$, $\langle 1 \rangle 11$, $\langle 1 \rangle 12$, $DotDotOneThruN$
$\langle 1 \rangle 14.$ $Len(R) = Len(Q) - 1$BY $\langle 1 \rangle 13$ DEF $LenQ, LenR$

The mapping of indexes forward.

$\langle 1 \rangle 15.$ $RemoveAt\_MapForward(Q, n)$
$\langle 2 \rangle 1.$ SUFFICES ASSUME

NEW qi, $qi \in 1 \,..\, LenQ$, $qi \neq n$,
NEW ri, $ri = RemoveAt\_ForwardIndex(Q, n, qi)$
PROVE $ri \in 1 \,..\, LenQ - 1 \land Q[qi] = R[ri]$
BY $\langle 1 \rangle 13$ DEF R, $LenQ$, $LenR$, $RemoveAt\_MapForward$
$\langle 2 \rangle 2$. $qi \in 1 \,..\, LenQ \land qi \neq n$ BY $\langle 2 \rangle 1$
$\langle 2 \rangle 3$. CASE $qi < n$
$\langle 3 \rangle 1$. $ri = qi$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 3$
$\langle 3 \rangle 2$. $n \leq LenQ$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $SMTT(10)$
$\langle 3 \rangle 3$. $ri \in 1 \,..\, LenQ - 1$ BY $\langle 1 \rangle 3$, $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $SMTT(10)$
$\langle 3 \rangle 4$. $R[ri] = Q[qi]$ BY $\langle 1 \rangle 6$, $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 3 \rangle 1$, $\langle 3 \rangle 3$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 3$, $\langle 3 \rangle 4$
$\langle 2 \rangle 4$. CASE $qi > n$
$\langle 3 \rangle 1$. $\neg(qi < n)$ BY $\langle 1 \rangle 3$, $\langle 2 \rangle 2$, $\langle 2 \rangle 4$, $SMTT(10)$
$\langle 3 \rangle 2$. $ri = qi - 1$ BY $\langle 2 \rangle 1$, $\langle 3 \rangle 1$
$\langle 3 \rangle 3$. $qi > 1$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\langle 2 \rangle 2$, $\langle 2 \rangle 4$, $SMTT(10)$
$\langle 3 \rangle 4$. $ri \in 1 \,..\, LenQ - 1$ BY $\langle 1 \rangle 3$, $\langle 2 \rangle 2$, $\langle 3 \rangle 2$, $\langle 3 \rangle 3$, $SMTT(10)$
$\langle 3 \rangle 5$. $ri + 1 = qi$ BY $\langle 1 \rangle 3$, $\langle 2 \rangle 2$, $\langle 3 \rangle 2$, $SMTT(10)$
$\langle 3 \rangle 6$. $\neg(ri < n)$ BY $\langle 1 \rangle 3$, $\langle 2 \rangle 2$, $\langle 2 \rangle 4$, $\langle 3 \rangle 2$, $SMTT(10)$
$\langle 3 \rangle 7$. $R[ri] = Q[qi]$ BY $\langle 1 \rangle 6$, $\langle 3 \rangle 4$, $\langle 3 \rangle 5$, $\langle 3 \rangle 6$, $SMTT(10)$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 4$, $\langle 3 \rangle 7$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 2 \rangle 4$, $SMTT(10)$

The mapping of indexes backward.

$\langle 1 \rangle 16$. $RemoveAt\_MapBackward(Q, n)$
$\langle 2 \rangle 1$. SUFFICES ASSUME
NEW ri, $ri \in 1 \,..\, LenR$,
NEW qi, $qi = RemoveAt\_BackwardIndex(Q, n, ri)$
PROVE $qi \in 1 \,..\, LenQ \land qi \neq n \land Q[qi] = R[ri]$
BY DEF R, $LenR$, $LenQ$, $RemoveAt\_MapBackward$
$\langle 2 \rangle 2$. $ri \in 1 \,..\, LenR$ BY $\langle 2 \rangle 1$
$\langle 2 \rangle 3$. CASE $ri < n$
$\langle 3 \rangle 1$. $qi = ri$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 3$
$\langle 3 \rangle 2$. $qi \in 1 \,..\, LenQ$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 13$, $\langle 2 \rangle 2$, $\langle 3 \rangle 1$, $SMTT(10)$
$\langle 3 \rangle 3$. $qi \neq n$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 3$, $SMTT(10)$
$\langle 3 \rangle 4$. $R[ri] = Q[qi]$ BY $\langle 1 \rangle 6$, $\langle 1 \rangle 13$, $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 3 \rangle 1$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$, $\langle 3 \rangle 3$, $\langle 3 \rangle 4$
$\langle 2 \rangle 4$. CASE $\neg(ri < n)$
$\langle 3 \rangle 1$. $qi = ri + 1$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 4$
$\langle 3 \rangle 2$. $qi \in 1 \,..\, LenQ$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 13$, $\langle 2 \rangle 2$, $\langle 3 \rangle 1$, $SMTT(10)$
$\langle 3 \rangle 3$. $qi \neq n$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 4$, $\langle 3 \rangle 1$, $SMTT(10)$
$\langle 3 \rangle 4$. $R[ri] = Q[qi]$ BY $\langle 1 \rangle 6$, $\langle 1 \rangle 13$, $\langle 2 \rangle 2$, $\langle 2 \rangle 4$, $\langle 3 \rangle 1$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$, $\langle 3 \rangle 3$, $\langle 3 \rangle 4$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 3$, $\langle 2 \rangle 4$

Each index maps forward.

$\langle 1 \rangle 17.$ $RemoveAt\_EachForward(Q, n)$
$\langle 2 \rangle 1.$ SUFFICES ASSUME NEW qi, $qi \in 1 \ldots Len(Q)$, $qi \neq n$
PROVE $\exists\, ri \in 1 \ldots Len(R) : Q[qi] = R[ri]$
BY DEF $RemoveAt\_EachForward$, R
$\langle 2 \rangle$ DEFINE $ri \triangleq RemoveAt\_ForwardIndex(Q, n, qi)$
$\langle 2 \rangle 2.$ $ri \in 1 \ldots Len(R) \land Q[qi] = R[ri]$
BY $\langle 1 \rangle 15$, $\langle 2 \rangle 1$ DEF $RemoveAt\_MapForward$, R
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$

Each index maps backward.

$\langle 1 \rangle 18.$ $RemoveAt\_EachBackward(Q, n)$
$\langle 2 \rangle 1.$ SUFFICES ASSUME NEW ri, $ri \in 1 \ldots Len(R)$
PROVE $\exists\, qi \in 1 \ldots Len(Q) : qi \neq n \land Q[qi] = R[ri]$
BY DEF $RemoveAt\_EachBackward$, R
$\langle 2 \rangle$ DEFINE $qi \triangleq RemoveAt\_BackwardIndex(Q, n, ri)$
$\langle 2 \rangle 2.$ $qi \in 1 \ldots Len(Q) \land qi \neq n \land Q[qi] = R[ri]$
BY $\langle 1 \rangle 16$, $\langle 2 \rangle 1$ DEF $RemoveAt\_MapBackward$, R
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$

Indexes map forward preserving order.

$\langle 1 \rangle 19.$ $RemoveAt\_OrderedForward(Q, n)$
$\langle 2 \rangle 1.$ SUFFICES ASSUME
NEW $qi1$, $qi1 \in 1 \ldots LenQ$, $qi1 \neq n$,
NEW $qi2$, $qi2 \in 1 \ldots LenQ$, $qi2 \neq n$,
$qi1 < qi2$
PROVE
$\exists\, ri1, ri2 \in 1 \ldots Len(R) :$
$ri1 < ri2 \land Q[qi1] = R[ri1] \land Q[qi2] = R[ri2]$
BY Isa DEF $RemoveAt\_OrderedForward$, R, $LenQ$
$\langle 2 \rangle$ DEFINE $ri1 \triangleq RemoveAt\_ForwardIndex(Q, n, qi1)$
$\langle 2 \rangle$ DEFINE $ri2 \triangleq RemoveAt\_ForwardIndex(Q, n, qi2)$
$\langle 2 \rangle 2.$ $ri1 \in 1 \ldots Len(R) \land Q[qi1] = R[ri1]$
BY $\langle 1 \rangle 15$, $\langle 2 \rangle 1$ DEF $RemoveAt\_MapForward$, R, $LenQ$
$\langle 2 \rangle 3.$ $ri2 \in 1 \ldots Len(R) \land Q[qi2] = R[ri2]$
BY $\langle 1 \rangle 15$, $\langle 2 \rangle 1$ DEF $RemoveAt\_MapForward$, R, $LenQ$
$\langle 2 \rangle 4.$ $ri1 < ri2$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\langle 2 \rangle 1$, $SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 2 \rangle 4$

Indexes map backward preserving order.

$\langle 1 \rangle 20.$ $RemoveAt\_OrderedBackward(Q, n)$
$\langle 2 \rangle 1.$ SUFFICES ASSUME
NEW $ri1$, $ri1 \in 1 \ldots LenR$,
NEW $ri2$, $ri2 \in 1 \ldots LenR$,
$ri1 < ri2$
PROVE
$\exists\, qi1, qi2 \in 1 \ldots Len(Q) :$

$qi1 < qi2 \land qi1 \neq n \land qi2 \neq n \land Q[qi1] = R[ri1] \land Q[qi2] = R[ri2]$
BY Isa DEF $RemoveAt\_OrderedBackward,\ R,\ LenR$
$\langle 2\rangle$ DEFINE $qi1 \triangleq RemoveAt\_BackwardIndex(Q,\ n,\ ri1)$
$\langle 2\rangle$ DEFINE $qi2 \triangleq RemoveAt\_BackwardIndex(Q,\ n,\ ri2)$
$\langle 2\rangle 2.$ $qi1 \in 1\,..\,Len(Q) \land qi1 \neq n \land Q[qi1] = R[ri1]$
BY $\langle 1\rangle 16,\ \langle 2\rangle 1$ DEF $RemoveAt\_MapBackward,\ R,\ LenR$
$\langle 2\rangle 3.$ $qi2 \in 1\,..\,Len(Q) \land qi2 \neq n \land Q[qi2] = R[ri2]$
BY $\langle 1\rangle 16,\ \langle 2\rangle 1$ DEF $RemoveAt\_MapBackward,\ R,\ LenR$
$\langle 2\rangle 4.$ $qi1 < qi2$ BY $\langle 1\rangle 2,\ \langle 1\rangle 12,\ \langle 2\rangle 1,\ SMTT(10)$
$\langle 2\rangle$ QED BY $\langle 2\rangle 2,\ \langle 2\rangle 3,\ \langle 2\rangle 4$

Indexes map forward preserving distinctness. Essentially an identical proof to $RemoveAt\_OrderedForward(Q,\ n)$.

$\langle 1\rangle 21.$ $RemoveAt\_DistinctForward(Q,\ n)$
$\langle 2\rangle 1.$ SUFFICES ASSUME
NEW $qi1,\ qi1 \in 1\,..\,LenQ,\ qi1 \neq n,$
NEW $qi2,\ qi2 \in 1\,..\,LenQ,\ qi2 \neq n,$
$qi1 \neq qi2$
PROVE
$\exists\, ri1,\ ri2 \in 1\,..\,Len(R):$
$ri1 \neq ri2 \land Q[qi1] = R[ri1] \land Q[qi2] = R[ri2]$
BY Isa DEF $RemoveAt\_DistinctForward,\ R,\ LenQ$
$\langle 2\rangle$ DEFINE $ri1 \triangleq RemoveAt\_ForwardIndex(Q,\ n,\ qi1)$
$\langle 2\rangle$ DEFINE $ri2 \triangleq RemoveAt\_ForwardIndex(Q,\ n,\ qi2)$
$\langle 2\rangle 2.$ $ri1 \in 1\,..\,Len(R) \land Q[qi1] = R[ri1]$
BY $\langle 1\rangle 15,\ \langle 2\rangle 1$ DEF $RemoveAt\_MapForward,\ R,\ LenQ$
$\langle 2\rangle 3.$ $ri2 \in 1\,..\,Len(R) \land Q[qi2] = R[ri2]$
BY $\langle 1\rangle 15,\ \langle 2\rangle 1$ DEF $RemoveAt\_MapForward,\ R,\ LenQ$
$\langle 2\rangle 4.$ $ri1 \neq ri2$ BY $\langle 1\rangle 2,\ \langle 1\rangle 3,\ \langle 2\rangle 1,\ SMTT(10)$
$\langle 2\rangle$ QED BY $\langle 2\rangle 2,\ \langle 2\rangle 3,\ \langle 2\rangle 4$

Indexes map backward preserving distinctness. Essentially an identical proof to $RemoveAt\_OrderedBackward(Q,\ n)$.

$\langle 1\rangle 22.$ $RemoveAt\_DistinctBackward(Q,\ n)$
$\langle 2\rangle 1.$ SUFFICES ASSUME
NEW $ri1,\ ri1 \in 1\,..\,LenR,$
NEW $ri2,\ ri2 \in 1\,..\,LenR,$
$ri1 \neq ri2$
PROVE
$\exists\, qi1,\ qi2 \in 1\,..\,Len(Q):$
$qi1 \neq qi2 \land qi1 \neq n \land qi2 \neq n \land Q[qi1] = R[ri1] \land Q[qi2] = R[ri2]$
BY Isa DEF $RemoveAt\_DistinctBackward,\ R,\ LenR$
$\langle 2\rangle$ DEFINE $qi1 \triangleq RemoveAt\_BackwardIndex(Q,\ n,\ ri1)$
$\langle 2\rangle$ DEFINE $qi2 \triangleq RemoveAt\_BackwardIndex(Q,\ n,\ ri2)$
$\langle 2\rangle 2.$ $qi1 \in 1\,..\,Len(Q) \land qi1 \neq n \land Q[qi1] = R[ri1]$
BY $\langle 1\rangle 16,\ \langle 2\rangle 1$ DEF $RemoveAt\_MapBackward,\ R,\ LenR$
$\langle 2\rangle 3.$ $qi2 \in 1\,..\,Len(Q) \land qi2 \neq n \land Q[qi2] = R[ri2]$
BY $\langle 1\rangle 16,\ \langle 2\rangle 1$ DEF $RemoveAt\_MapBackward,\ R,\ LenR$

$\langle 2\rangle 4.\ qi1 \neq qi2$ BY $\langle 1\rangle 2$, $\langle 1\rangle 12$, $\langle 2\rangle 1$, $SMTT(10)$
$\langle 2\rangle$ QED BY $\langle 2\rangle 2$, $\langle 2\rangle 3$, $\langle 2\rangle 4$

$\langle 1\rangle$ QED BY $\langle 1\rangle 9$, $\langle 1\rangle 14$, $\langle 1\rangle 15$, $\langle 1\rangle 16$, $\langle 1\rangle 17$, $\langle 1\rangle 18$, $\langle 1\rangle 19$, $\langle 1\rangle 20$, $\langle 1\rangle 21$, $\langle 1\rangle 22$ DEF R

C.5 Facts about finite sets

MODULE $NaiadClockProofFiniteSets$

EXTENDS $NaiadClockProofRemoveAt$

Facts about finite sets.

This really ought to be a library of theorems.

The built-in definition of $IsFiniteSet$ in $TLAPS$ version 1.0.25464 has a typo. This is the correct definition. 15 $June$ 2012

Still wrong in $TLAPM$ version 1.1.1 (commit 29945). 7 $December$ 2012

THEOREM $CorrectIsFiniteSet \triangleq$
  $\forall S : IsFiniteSet(S) \equiv \exists seq \in Seq(S) : \forall s \in S : \exists n \in 1 \,..\, Len(seq) : seq[n] = s$

The empty set is finite.

THEOREM $FiniteSetEmpty \triangleq$
  $IsFiniteSet(\{\})$
PROOF
  $\langle 1 \rangle$ DEFINE $Q \triangleq \langle\rangle$
  $\langle 1 \rangle 1.\ Q \in Seq(\{\})$ BY $IsASeq,\ IsaT(120)$
  $\langle 1 \rangle 2.\ \forall s \in \{\} : \exists i \in 1 \,..\, Len(Q) : Q[i] = s$ OBVIOUS
  $\langle 1 \rangle$ QED BY $\langle 1 \rangle 1,\ \langle 1 \rangle 2,\ CorrectIsFiniteSet$

A singleton set is finite.

THEOREM $FiniteSetSingleton \triangleq$
  ASSUME NEW $s0$
  PROVE $IsFiniteSet(\{s0\})$
PROOF
  $\langle 1 \rangle$ DEFINE $Q \triangleq \langle s0 \rangle$
  $\langle 1 \rangle$ DEFINE $LenQ \triangleq Len(Q)$

$\langle 1 \rangle 1.\ LenQ = 1$ OBVIOUS
$\langle 1 \rangle 2.\ Q \in Seq(\{s0\})$ BY $IsASeq,\ IsaT(120)$
$\langle 1 \rangle 3.\ \forall\, s \in \{s0\} : \exists\, i \in 1 \mathrel{..} LenQ : Q[i] = s$
  $\langle 2 \rangle$ SUFFICES $\exists\, i \in 1 \mathrel{..} LenQ : Q[i] = s0$ OBVIOUS
  $\langle 2 \rangle 1.\ 1 \in 1 \mathrel{..} LenQ$
    $\langle 3 \rangle$ HIDE DEF $LenQ$
    $\langle 3 \rangle$ QED BY $\langle 1 \rangle 1,\ SMTT(10)$
  $\langle 2 \rangle 2.\ Q[1] = s0$ OBVIOUS
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 2$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2,\ \langle 1 \rangle 3,\ CorrectIsFiniteSet$

Any subset of a finite set is finite.

THEOREM $FiniteSetSubset \triangleq$
  ASSUME
    NEW $S,\ IsFiniteSet(S)$,
    NEW $D \in$ SUBSET S
  PROVE $IsFiniteSet(D)$
PROOF
  $\langle 1 \rangle 1.$ SUFFICES ASSUME $D \neq \{\}$ PROVE $IsFiniteSet(D)$ BY $FiniteSetEmpty$

  Now we only have to consider non-empty D. Hence we can pick an element of D.

  $\langle 1 \rangle$ PICK $d0 \in D$: TRUE BY $\langle 1 \rangle 1$

  Since S is finite, we can pick a $Q \in Seq(S)$ on which each element of S appears.

  $\langle 1 \rangle 2.$ PICK $Q \in Seq(S) : \forall\, s \in S : \exists\, i \in 1 \mathrel{..} Len(Q) : Q[i] = s$ BY $CorrectIsFiniteSet$
  $\langle 1 \rangle$ DEFINE $LenQ \triangleq Len(Q)$
  $\langle 1 \rangle$ HIDE DEF $LenQ$
  $\langle 1 \rangle 3.\ LenQ \in Nat$ BY $LenInNat$ DEF $LenQ$

  Using Q construct $P \in Seq(D)$ on which each element of D appears.

  $\langle 1 \rangle$ DEFINE $P \triangleq [i \in 1 \mathrel{..} LenQ \mapsto$ IF $Q[i] \in D$ THEN $Q[i]$ ELSE $d0]$
  $\langle 1 \rangle$ HIDE DEF P
  $\langle 1 \rangle 4.\ P \in Seq(D)$
    $\langle 2 \rangle 1.\ \forall\, i \in 1 \mathrel{..} LenQ : P[i] \in D$ BY DEF P
    $\langle 2 \rangle 2.\ P \in [1 \mathrel{..} LenQ \rightarrow D]$ BY $\langle 2 \rangle 1$ DEF P
    $\langle 2 \rangle$ QED BY $\langle 1 \rangle 3,\ \langle 2 \rangle 2,\ IsASeq$
  $\langle 1 \rangle$ DEFINE $LenP \triangleq Len(P)$
  $\langle 1 \rangle$ HIDE DEF $LenP$
  $\langle 1 \rangle 5.\ LenP = LenQ$
    $\langle 2 \rangle 1.$ DOMAIN $P = 1 \mathrel{..} LenQ$ BY DEF P
    $\langle 2 \rangle$ QED BY $\langle 1 \rangle 3,\ \langle 1 \rangle 4,\ \langle 2 \rangle 1,\ LenDomain$ DEF $LenP$

  Prove that each element of D appears on P.

$\langle 1 \rangle 6.\ \forall\, d \in D : \exists\, i \in 1 \mathrel{..} \mathit{LenP} : P[i] = d$
$\quad\langle 2 \rangle$ SUFFICES ASSUME NEW $d \in D$ PROVE $\exists\, i \in 1 \mathrel{..} \mathit{LenP} : P[i] = d$ BY DEF LenP
$\quad\langle 2 \rangle 1.\ d \in S$ OBVIOUS
$\quad\langle 2 \rangle 2.\ \exists\, i \in 1 \mathrel{..} \mathit{LenQ} : Q[i] = d$ BY $\langle 1 \rangle 2$, $\langle 2 \rangle 1$ DEF LenQ
$\quad\langle 2 \rangle 3.\ \exists\, i \in 1 \mathrel{..} \mathit{LenP} : P[i] = d$ BY $\langle 1 \rangle 5$, $\langle 2 \rangle 2$ DEF P
$\quad\langle 2 \rangle$ QED BY $\langle 2 \rangle 3$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 4$, $\langle 1 \rangle 6$, $\mathit{CorrectIsFiniteSet}$ DEF LenP

The union of two finite sets is finite.

THEOREM $\mathit{FiniteSetUnion} \triangleq$
$\quad$ASSUME
$\quad\quad$NEW $S1$, $\mathit{IsFiniteSet}(S1)$,
$\quad\quad$NEW $S2$, $\mathit{IsFiniteSet}(S2)$
$\quad$PROVE $\mathit{IsFiniteSet}(S1 \cup S2)$
PROOF

Since $S1$ is finite, we can pick a $Q1 \in \mathit{Seq}(S1)$ on which each element of $S1$ appears.

$\langle 1 \rangle 1.$ PICK $Q1 \in \mathit{Seq}(S1) : \forall\, s \in S1 : \exists\, i \in 1 \mathrel{..} \mathit{Len}(Q1) : Q1[i] = s$ BY $\mathit{CorrectIsFiniteSet}$
$\langle 1 \rangle$ DEFINE $\mathit{LenQ1} \triangleq \mathit{Len}(Q1)$
$\langle 1 \rangle$ HIDE DEF $\mathit{LenQ1}$
$\langle 1 \rangle 2.\ \mathit{LenQ1} \in \mathit{Nat}$ BY $\mathit{LenInNat}$ DEF $\mathit{LenQ1}$
$\langle 1 \rangle a.\ Q1 \in [1 \mathrel{..} \mathit{LenQ1} \to S1]$ BY $\mathit{LenAxiom}$ DEF $\mathit{LenQ1}$

Since $S2$ is finite, we can pick a $Q2 \in \mathit{Seq}(S2)$ on which each element of $S2$ appears.

$\langle 1 \rangle 3.$ PICK $Q2 \in \mathit{Seq}(S2) : \forall\, s \in S2 : \exists\, i \in 1 \mathrel{..} \mathit{Len}(Q2) : Q2[i] = s$ BY $\mathit{CorrectIsFiniteSet}$
$\langle 1 \rangle$ DEFINE $\mathit{LenQ2} \triangleq \mathit{Len}(Q2)$
$\langle 1 \rangle$ HIDE DEF $\mathit{LenQ2}$
$\langle 1 \rangle 4.\ \mathit{LenQ2} \in \mathit{Nat}$ BY $\mathit{LenInNat}$ DEF $\mathit{LenQ2}$
$\langle 1 \rangle b.\ Q2 \in [1 \mathrel{..} \mathit{LenQ2} \to S2]$ BY $\mathit{LenAxiom}$ DEF $\mathit{LenQ2}$

From $Q1$ and $Q2$ construct a sequence $Q \in \mathit{Seq}(S1 \cup S2)$ on which each element of $S1 \cup S2$ appears.

$\langle 1 \rangle$ DEFINE $S \triangleq S1 \cup S2$
$\langle 1 \rangle$ DEFINE $n \triangleq \mathit{LenQ1} + \mathit{LenQ2}$
$\langle 1 \rangle 5.\ n \in \mathit{Nat}$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 4$, $\mathit{SMTT}(10)$
$\langle 1 \rangle$ DEFINE $Q \triangleq [i \in 1 \mathrel{..} n \mapsto$ IF $i \leq \mathit{LenQ1}$ THEN $Q1[i]$ ELSE $Q2[i - \mathit{LenQ1}]]$
$\langle 1 \rangle$ HIDE DEF Q
$\langle 1 \rangle 6.\ Q \in \mathit{Seq}(S)$
$\quad\langle 2 \rangle 1.\ \forall\, i \in 1 \mathrel{..} n : Q[i] \in S$
$\quad\quad\langle 3 \rangle$ SUFFICES ASSUME NEW $i \in 1 \mathrel{..} n$ PROVE $Q[i] \in S$ OBVIOUS
$\quad\quad\langle 3 \rangle 1.$ CASE $i \leq \mathit{LenQ1}$
$\quad\quad\quad\langle 4 \rangle 1.\ Q[i] = Q1[i]$ BY $\langle 3 \rangle 1$ DEF Q
$\quad\quad\quad\langle 4 \rangle 2.\ i \in 1 \mathrel{..} \mathit{LenQ1}$ BY $\langle 3 \rangle 1$, $\langle 1 \rangle 2$, $\langle 1 \rangle 4$, $\mathit{DotDotDef}$, $\mathit{SMTT}(10)$

$\langle 4\rangle 3.\ Q1[i] \in S1$ BY $\langle 4\rangle 2,\ \langle 1\rangle$a
$\langle 4\rangle$ QED BY $\langle 4\rangle 1,\ \langle 4\rangle 3$
$\langle 3\rangle 2.$ CASE $\neg(i \leq LenQ1)$
$\langle 4\rangle 1.\ Q[i] = Q2[i - LenQ1]$ BY $\langle 3\rangle 2$ DEF Q
$\langle 4\rangle 2.\ i - LenQ1 \in 1 \,..\, LenQ2$ BY $\langle 3\rangle 2,\ \langle 1\rangle 2,\ \langle 1\rangle 4,\ DotDotDef,\ SMTT(10)$
$\langle 4\rangle 3.\ Q2[i - LenQ1] \in S2$ BY $\langle 4\rangle 2,\ \langle 1\rangle$b
$\langle 4\rangle$ QED BY $\langle 4\rangle 1,\ \langle 4\rangle 3$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ \langle 3\rangle 2$
$\langle 2\rangle 2.\ Q \in [1 \,..\, n \rightarrow S]$ BY $\langle 2\rangle 1$ DEF Q
$\langle 2\rangle$ QED BY $\langle 1\rangle 5,\ \langle 2\rangle 2,\ IsASeq$
$\langle 1\rangle$ DEFINE $LenQ \triangleq Len(Q)$
$\langle 1\rangle$ HIDE DEF $LenQ$
$\langle 1\rangle 7.\ LenQ = n$
$\langle 2\rangle 1.$ DOMAIN $Q = 1 \,..\, n$ BY DEF Q
$\langle 2\rangle$ QED BY $\langle 1\rangle 5,\ \langle 1\rangle 6,\ \langle 2\rangle 1,\ LenDomain$ DEF $LenQ$

Prove that each element of S appears on Q.

$\langle 1\rangle 8.\ \forall s \in S : \exists i \in 1 \,..\, LenQ : Q[i] = s$
$\langle 2\rangle$ SUFFICES ASSUME NEW $s \in S$ PROVE $\exists i \in 1 \,..\, n : Q[i] = s$ BY $\langle 1\rangle 7$
$\langle 2\rangle 1.$ CASE $s \in S1$
$\langle 3\rangle 1.$ PICK $i1 \in 1 \,..\, LenQ1 : Q1[i1] = s$ BY $\langle 2\rangle 1,\ \langle 1\rangle 1$ DEF $LenQ1$
$\langle 3\rangle 2.\ i1 \leq LenQ1$ BY $\langle 1\rangle 2,\ DotDotDef,\ SMTT(10)$
$\langle 3\rangle 3.\ i1 \in 1 \,..\, n$ BY $\langle 1\rangle 2,\ \langle 1\rangle 4,\ DotDotDef,\ SMTT(10)$
$\langle 3\rangle 4.\ Q[i1] = s$ BY $\langle 3\rangle 1,\ \langle 3\rangle 2,\ \langle 3\rangle 3$ DEF Q
$\langle 3\rangle$ QED BY $\langle 3\rangle 3,\ \langle 3\rangle 4$
$\langle 2\rangle 2.$ CASE $s \in S2$
$\langle 3\rangle 1.$ PICK $i2 \in 1 \,..\, LenQ2 : Q2[i2] = s$ BY $\langle 2\rangle 2,\ \langle 1\rangle 3$ DEF $LenQ2$
$\langle 3\rangle 2.\ \neg(i2 + LenQ1 \leq LenQ1)$ BY $\langle 1\rangle 2,\ \langle 1\rangle 4,\ DotDotDef,\ SMTT(10)$
$\langle 3\rangle 3.\ i2 + LenQ1 \in 1 \,..\, n$ BY $\langle 1\rangle 2,\ \langle 1\rangle 4,\ DotDotDef,\ SMTT(10)$
$\langle 3\rangle 4.\ Q[i2 + LenQ1] = s$
$\langle 4\rangle 1.\ (i2 + LenQ1) - LenQ1 = i2$ BY $\langle 1\rangle 2,\ \langle 1\rangle 4,\ SMTT(10)$
$\langle 4\rangle 2.\ Q[i2 + LenQ1] = Q2[i2]$ BY $\langle 4\rangle 1,\ \langle 3\rangle 2,\ \langle 3\rangle 3$ DEF Q
$\langle 4\rangle$ QED BY $\langle 4\rangle 2,\ \langle 3\rangle 1$
$\langle 3\rangle$ QED BY $\langle 3\rangle 3,\ \langle 3\rangle 4$
$\langle 2\rangle$ QED BY $\langle 2\rangle 1,\ \langle 2\rangle 2$
$\langle 1\rangle$ QED BY $\langle 1\rangle 6,\ \langle 1\rangle 8,\ CorrectIsFiniteSet$ DEF $LenQ$

C.6 Facts about exact sequences

MODULE $NaiadClockProofExactSeqs$

EXTENDS $NaiadClockProofFiniteSets$

Facts about exact sequences.

This really ought to be a library of theorems.

An exact sequence exists for any finite set.

THEOREM $ExactSeqExists \triangleq$
  ASSUME
    NEW S, $IsFiniteSet(S)$
  PROVE
  $\exists\, Q : IsExactSeqFor(Q, S)$
PROOF
  $\langle 1 \rangle$ USE DEF $ExactSeq\_Each$
  $\langle 1 \rangle$ USE DEF $ExactSeq\_Once$
  $\langle 1 \rangle$ DEFINE $each(Q) \triangleq ExactSeq\_Each(Q, S)$
  $\langle 1 \rangle$ DEFINE $once(Q) \triangleq ExactSeq\_Once(Q)$

  $\langle 1 \rangle$ SUFFICES $\exists\, n \in Nat : \exists\, Q \in [1 \,..\, n \rightarrow S] : each(Q) \land once(Q)$
    $\langle 2 \rangle 1.$ PICK $n \in Nat : \exists\, Q \in [1 \,..\, n \rightarrow S] : each(Q) \land once(Q)$ OBVIOUS
    $\langle 2 \rangle 2.$ PICK $Q \in [1 \,..\, n \rightarrow S] : each(Q) \land once(Q)$ OBVIOUS
    $\langle 2 \rangle 3.$ $Q \in Seq(S)$ BY $IsASeq$
    $\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$ DEF $IsExactSeqFor$

Define N as the set of all natural numbers n such that there exists a sequence of length n that contains each element of S. Because S is finite, such a sequence exists and hence N is non-empty.

  $\langle 1 \rangle$ DEFINE $N \triangleq \{n \in Nat : \exists\, Q \in [1 \,..\, n \rightarrow S] : each(Q)\}$
  $\langle 1 \rangle 1.$ $N \neq \{\}$
    $\langle 2 \rangle 1.$ PICK $Q \in Seq(S) : each(Q)$ BY $CorrectIsFiniteSet$
    $\langle 2 \rangle 2.$ $Len(Q) \in Nat$ BY $LenInNat$
    $\langle 2 \rangle 3.$ $Q \in [1 \,..\, Len(Q) \rightarrow S]$ BY $\langle 2 \rangle 1$, $LenAxiom$
    $\langle 2 \rangle 4.$ $Len(Q) \in N$BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $\langle 2 \rangle 3$
    $\langle 2 \rangle$ QED BY $\langle 2 \rangle 4$
  $\langle 1 \rangle$ HIDE DEF N

Pick the smallest $n \in N$.

$\langle 1 \rangle 2.$ PICK $n \in Nat :$
$\quad \wedge \ n \in N$
$\quad \wedge \ \forall m \in N : n \leq m$
$\langle 2 \rangle 1.$ PICK $n \in N : \forall m \in N : n \leq m$
$\langle 3 \rangle 1.$ $N \in$ SUBSET Nat BY DEF N
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 1 \rangle 1$, $NatWellFounded$
$\langle 2 \rangle 2.$ $n \in Nat$ BY DEF N
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$

Pick Q a sequence of length n that contains each element of S. Since this is the smallest such length, Q can contain no duplicates.

$\langle 1 \rangle 3.$ PICK $Q \in [1 \,..\, n \rightarrow S] : each(Q)$
$\langle 2 \rangle 1.$ $n \in N$ BY $\langle 1 \rangle 2$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$ DEF N
$\langle 1 \rangle 4.$ SUFFICES $once(Q)$
$\langle 2 \rangle$ HIDE DEF $each$, $once$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\langle 1 \rangle 4$ DEF N

To show that every element on Q appears at most once, we assume that Q contains duplicates and derive a contradiction.

$\langle 1 \rangle 5.$ SUFFICES ASSUME $\neg once(Q)$ PROVE FALSE OBVIOUS

It turns out to be important to know that $Len(Q) = n$ and is a natural.

$\langle 1 \rangle$ DEFINE $LenQ \triangleq Len(Q)$
$\langle 1 \rangle$ HIDE DEF $LenQ$
$\langle 1 \rangle 6.$ $LenQ = n \wedge LenQ \in Nat$
$\langle 2 \rangle 1.$ $Q \in Seq(S)$ BY $IsASeq$
$\langle 2 \rangle 2.$ $LenQ \in Nat$ BY $\langle 2 \rangle 1$, $LenInNat$ DEF $LenQ$
$\langle 2 \rangle 3.$ DOMAIN $Q = 1 \,..\, LenQ$ BY $\langle 2 \rangle 1$, $LenAxiom$ DEF $LenQ$
$\langle 2 \rangle 4.$ $1 \,..\, LenQ = 1 \,..\, n$BY $\langle 2 \rangle 3$
$\langle 2 \rangle 5.$ $LenQ = n$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 4$, $DotDotOneThruN$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 5$

Under the assumption that Q has duplicate elements, we can pick two distinct indexes j and k containing the same element. Without loss of generality, let j be the smaller index.

$\langle 1 \rangle 7.$ PICK $j, k \in Nat : Q[j] = Q[k] \wedge 1 \leq j \wedge j < k \wedge k \leq LenQ$
$\langle 2 \rangle$ $LenQ \in Nat$ BY $\langle 1 \rangle 6$
$\langle 2 \rangle 1.$ PICK $ja, ka \in 1 \,..\, LenQ : Q[ja] = Q[ka] \wedge ja \neq ka$ BY $\langle 1 \rangle 5$ DEF $LenQ$
$\langle 2 \rangle$ $ja \in Nat$ BY $\langle 2 \rangle 1$, $SMTT(10)$
$\langle 2 \rangle$ $ka \in Nat$ BY $\langle 2 \rangle 1$, $SMTT(10)$
$\langle 2 \rangle 2.$ $1 \leq ja$ BY $\langle 2 \rangle 1$, $SMTT(10)$
$\langle 2 \rangle 3.$ $1 \leq ka$ BY $\langle 2 \rangle 1$, $SMTT(10)$
$\langle 2 \rangle 4.$ $ja \leq LenQ$ BY $\langle 2 \rangle 1$, $SMTT(10)$
$\langle 2 \rangle 5.$ $ka \leq LenQ$ BY $\langle 2 \rangle 1$, $SMTT(10)$
$\langle 2 \rangle 6.$ CASE $ja < ka$ BY $\langle 2 \rangle 6$, $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $\langle 2 \rangle 5$
$\langle 2 \rangle 7.$ CASE $ka < ja$ BY $\langle 2 \rangle 7$, $\langle 2 \rangle 1$, $\langle 2 \rangle 3$, $\langle 2 \rangle 4$
$\langle 2 \rangle 8.$ $ja < ka \vee ka < ja$
$\langle 3 \rangle 1.$ $LenQ \in Nat$ BY $\langle 1 \rangle 6$
$\langle 3 \rangle 2.$ $ja \in Nat$ BY $\langle 3 \rangle 1$, $DotDotType$
$\langle 3 \rangle 3.$ $ka \in Nat$ BY $\langle 3 \rangle 1$, $DotDotType$

$\langle 3 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 3 \rangle 2$, $\langle 3 \rangle 3$, $SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 6$, $\langle 2 \rangle 7$, $\langle 2 \rangle 8$, $\langle 1 \rangle 6$, $SMTT(10)$

Define $m \triangleq n - 1$ and prove some properties of j, k, m, n. Later we construct a sequence P of length m.

$\langle 1 \rangle 8.$ $1 \leq j$ BY $\langle 1 \rangle 7$, $SMTT(10)$
$\langle 1 \rangle 9.$ $j < k$ BY $\langle 1 \rangle 7$
$\langle 1 \rangle 10.$ $k \leq n$ BY $\langle 1 \rangle 6$, $\langle 1 \rangle 7$, $SMTT(10)$
$\langle 1 \rangle 11.$ $1 < k$ BY $\langle 1 \rangle 8$, $\langle 1 \rangle 9$, $SMTT(10)$
$\langle 1 \rangle 12.$ $1 \leq k$ BY $\langle 1 \rangle 11$, $SMTT(10)$
$\langle 1 \rangle 13.$ $2 \leq n$ BY $\langle 1 \rangle 10$, $\langle 1 \rangle 11$, $SMTT(10)$
$\langle 1 \rangle 14.$ $n \neq 0$ BY $\langle 1 \rangle 13$, $SMTT(10)$
$\langle 1 \rangle 15.$ $n - 1 \in Nat$ BY $\langle 1 \rangle 14$, $SMTT(10)$
$\langle 1 \rangle 16.$ PICK $m \in Nat : m = n - 1$ BY $\langle 1 \rangle 15$
$\langle 1 \rangle 17.$ $m < n$ BY $\langle 1 \rangle 16$, $SMTT(10)$
$\langle 1 \rangle 18.$ $\neg(n \leq m)$ BY $\langle 1 \rangle 17$, $SMTT(10)$
$\langle 1 \rangle 19.$ $j < n$BY $\langle 1 \rangle 9$, $\langle 1 \rangle 10$, $SMTT(10)$
$\langle 1 \rangle 20.$ $j \leq m$ BY $\langle 1 \rangle 16$, $\langle 1 \rangle 19$, $SMTT(10)$
$\langle 1 \rangle 21.$ $j \in 1 \,..\, m$ BY $\langle 1 \rangle 8$, $\langle 1 \rangle 20$, $SMTT(10)$
$\langle 1 \rangle 22.$ $n \in 1 \,..\, n$ BY $\langle 1 \rangle 14$, $SMTT(10)$

$\langle 1 \rangle 23.$ ASSUME $k \neq n$ PROVE $k \in 1 \,..\, m$
$\langle 2 \rangle 1.$ $k \leq m$ BY $\langle 1 \rangle 10$, $\langle 1 \rangle 16$, $\langle 1 \rangle 23$, $SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 12$, $\langle 2 \rangle 1$, $SMTT(10)$

$\langle 1 \rangle 24.$ ASSUME NEW $i \in 1 \,..\, n$, $i \neq n$ PROVE $i \in 1 \,..\, m$
$\langle 2 \rangle 1.$ $1 \leq i$ BY $\langle 1 \rangle 24$, $SMTT(10)$
$\langle 2 \rangle 2.$ $i \leq n$ BY $\langle 1 \rangle 24$, $SMTT(10)$
$\langle 2 \rangle 3.$ $i < n$ BY $\langle 1 \rangle 24$, $\langle 2 \rangle 2$, $SMTT(10)$
$\langle 2 \rangle 4.$ $i \leq m$ BY $\langle 2 \rangle 3$, $\langle 1 \rangle 16$, $SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 4$, $SMTT(10)$

Construct P from Q as a shorter sequence in which each element of S appears. However, since Q is the shortest such sequence, this is a contradiction.

$\langle 1 \rangle$ DEFINE $P \triangleq [i \in 1 \,..\, m \mapsto$ IF $i = k$ THEN $Q[n]$ ELSE $Q[i]]$
$\langle 1 \rangle 25.$ $P \in [1 \,..\, m \rightarrow S]$
$\langle 2 \rangle$ SUFFICES ASSUME NEW $i \in 1 \,..\, m$ PROVE $P[i] \in S$ BY $SMTT(10)$
$\langle 2 \rangle 1.$ $i \in 1 \,..\, n$ BY $\langle 1 \rangle 16$, $SMTT(10)$
$\langle 2 \rangle 2.$ $n \in 1 \,..\, n$ BY $\langle 1 \rangle 22$
$\langle 2 \rangle 3.$ $Q[i] \in S$ BY $\langle 2 \rangle 1$
$\langle 2 \rangle 4.$ $Q[n] \in S$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 3$, $\langle 2 \rangle 4$
$\langle 1 \rangle$ HIDE DEF P
$\langle 1 \rangle 26.$ SUFFICES $each(P)$
$\langle 2 \rangle 2.$ $m \in N$ BY $\langle 1 \rangle 25$, $\langle 1 \rangle 26$ DEF N
$\langle 2 \rangle$ HIDE DEF $each$
$\langle 2 \rangle 3.$ $\neg(n \leq m)$BY $\langle 1 \rangle 18$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 1 \rangle 2$, $SMTT(10)$

To show that each element of S appears in P, we assume that P has missing elements and derive a contradiction.

$\langle 1 \rangle 27.$ SUFFICES ASSUME $\neg each(P)$ PROVE FALSE OBVIOUS

It turns out to be important to know that $Len(P) = m$ and is a natural.

$\langle 1 \rangle$ DEFINE $LenP \triangleq Len(P)$
$\langle 1 \rangle$ HIDE DEF $LenP$
$\langle 1 \rangle 28.$ $LenP = m \land LenP \in Nat$
  $\langle 2 \rangle$ HIDE DEF P
  $\langle 2 \rangle 2.$ $P \in Seq(S)$ BY $\langle 1 \rangle 25$, $IsASeq$
  $\langle 2 \rangle 3.$ $LenP \in Nat$ BY $\langle 2 \rangle 2$, $LenInNat$ DEF $LenP$
  $\langle 2 \rangle 4.$ DOMAIN $P = 1 \,..\, LenP$ BY $\langle 2 \rangle 2$, $LenAxiom$ DEF $LenP$
  $\langle 2 \rangle 5.$ $1 \,..\, LenP = 1 \,..\, m$BY $\langle 2 \rangle 4$, $\langle 1 \rangle 25$
  $\langle 2 \rangle 6.$ $LenP = m$ BY $\langle 2 \rangle 3$, $\langle 2 \rangle 5$, $DotDotOneThruN$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 6$

Since we assume that P has missing elements, we can pick an element that fails to appear. But this element appears on Q, from which we can find it on P, thus establishing a contradiction.

$\langle 1 \rangle 29.$ PICK $s \in S : \neg \exists\, i \in 1 \,..\, LenP : P[i] = s$ BY $\langle 1 \rangle 27$ DEF $LenP$
$\langle 1 \rangle 30.$ PICK $i \in 1 \,..\, n : Q[i] = s$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 6$ DEF $LenQ$
$\langle 1 \rangle 31.$ CASE $i = k$

  A duplicate of $Q[k]$ appears in $Q[j]$. Since $j \neq k$ and $j \in 1 \,..\, m$, we copied $Q[j]$ to $P[j]$.

  $\langle 2 \rangle 1.$ $j \neq k$ BY $\langle 1 \rangle 9$, $SMTT(10)$
  $\langle 2 \rangle 2.$ $P[j] = Q[j]$ BY $\langle 2 \rangle 1$, $\langle 1 \rangle 21$ DEF P
  $\langle 2 \rangle 3.$ $Q[j] = Q[k]$ BY $\langle 1 \rangle 7$
  $\langle 2 \rangle 4.$ $P[j] = s$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 1 \rangle 31$, $\langle 1 \rangle 30$
  $\langle 2 \rangle 5.$ $j \in 1 \,..\, LenP$ BY $\langle 1 \rangle 21$, $\langle 1 \rangle 28$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 4$, $\langle 2 \rangle 5$, $\langle 1 \rangle 29$
$\langle 1 \rangle 32.$ CASE $i \neq k \land i = n$

  Since $k \in 1 \,..\, m$, we copied $Q[n]$ to $P[k]$.

  $\langle 2 \rangle 1.$ $k \in 1 \,..\, m$ BY $\langle 1 \rangle 32$, $\langle 1 \rangle 23$
  $\langle 2 \rangle 2.$ $P[k] = Q[n]$ BY $\langle 2 \rangle 1$ DEF P
  $\langle 2 \rangle 3.$ $P[k] = s$ BY $\langle 2 \rangle 2$, $\langle 1 \rangle 32$, $\langle 1 \rangle 30$
  $\langle 2 \rangle 4.$ $k \in 1 \,..\, LenP$ BY $\langle 2 \rangle 1$, $\langle 1 \rangle 28$, $SMTT(10)$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 3$, $\langle 2 \rangle 4$, $\langle 1 \rangle 29$
$\langle 1 \rangle 33.$ CASE $i \neq k \land i \neq n$

  Since $i \neq k$ and $i \in 1 \,..\, m$, we copied $Q[i]$ to $P[i]$.

  $\langle 2 \rangle 1.$ $i \neq k \land i \in 1 \,..\, m$ BY $\langle 1 \rangle 33$, $\langle 1 \rangle 24$
  $\langle 2 \rangle 2.$ $P[i] = Q[i]$ BY $\langle 2 \rangle 1$ DEF P
  $\langle 2 \rangle 3.$ $P[i] = s$ BY $\langle 2 \rangle 2$, $\langle 1 \rangle 31$, $\langle 1 \rangle 30$
  $\langle 2 \rangle 4.$ $i \in 1 \,..\, LenP$ BY $\langle 2 \rangle 1$, $\langle 1 \rangle 28$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 3$, $\langle 2 \rangle 4$, $\langle 1 \rangle 29$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 31$, $\langle 1 \rangle 32$, $\langle 1 \rangle 33$

Having an exact sequence is the same as being a finite set.

THEOREM $ExactSeqIsFiniteSet \triangleq$
  ASSUME
    NEW S
  PROVE
  $IsFiniteSet(S) \equiv (\exists Q : IsExactSeqFor(Q, S))$
PROOF
  $\langle 1 \rangle 1.$ $IsFiniteSet(S) \Rightarrow (\exists Q : IsExactSeqFor(Q, S))$ BY $ExactSeqExists$
  $\langle 1 \rangle 2.$ $(\exists Q : IsExactSeqFor(Q, S)) \Rightarrow IsFiniteSet(S)$
    $\langle 2 \rangle 1.$ SUFFICES ASSUME NEW Q, $IsExactSeqFor(Q, S)$ PROVE $IsFiniteSet(S)$ OBVIOUS
    $\langle 2 \rangle 2.$ $Q \in Seq(S)$ BY $\langle 2 \rangle 1$ DEF $IsExactSeqFor$
    $\langle 2 \rangle 3.$ $ExactSeq\_Each(Q, S)$ BY $\langle 2 \rangle 1$ DEF $IsExactSeqFor$
    $\langle 2 \rangle 4.$ $\forall s \in S : \exists q \in 1 \,..\, Len(Q) : Q[q] = s$ BY $\langle 2 \rangle 3$ DEF $ExactSeq\_Each$
    $\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 2 \rangle 4$, $CorrectIsFiniteSet$
  $\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$

If S is a finite set, then $ExactSeqFor(S)$ is an exact sequence for S.

THEOREM $ExactSeqForProperties \triangleq$
  ASSUME
    NEW S, $IsFiniteSet(S)$
  PROVE
  $IsExactSeqFor(ExactSeqFor(S), S)$
PROOF
  $\langle 1 \rangle$ QED BY $ExactSeqExists$ DEF $ExactSeqFor$

The exact sequence for the empty set is the empty sequence.

THEOREM $ExactSeqEmpty \triangleq$
  ASSUME
    NEW S,
    NEW Q, $IsExactSeqFor(Q, S)$
  PROVE
  $Q = \langle\rangle \equiv S = \{\}$
PROOF

$\langle 1 \rangle$ DEFINE $LenQ \triangleq Len(Q)$
$\langle 1 \rangle 1.$ $Q \in Seq(S)$ BY DEF $IsExactSeqFor$
$\langle 1 \rangle 2.$ $LenQ \in Nat$ BY $\langle 1 \rangle 1$, $LenInNat$
$\langle 1 \rangle 3.$ $Q \in [1 \,..\, LenQ \rightarrow S]$ BY $\langle 1 \rangle 1$, $LenAxiom$
$\langle 1 \rangle 4.$ $\forall\, s \in S : \exists\, i \in 1 \,..\, LenQ : Q[i] = s$ BY DEF $IsExactSeqFor$, $ExactSeq\_Each$
$\langle 1 \rangle$ HIDE DEF $LenQ$
$\langle 1 \rangle 5.$ $Q = \langle\rangle \;\Rightarrow\; S = \{\}$
  $\langle 2 \rangle 1.$ SUFFICES ASSUME $Q = \langle\rangle$, $S \neq \{\}$ PROVE FALSE OBVIOUS
  $\langle 2 \rangle 2.$ $LenQ = 0$ BY $\langle 2 \rangle 1$, $EmptySeq$ DEF $LenQ$
  $\langle 2 \rangle 3.$ PICK $s \in S :$ TRUE BY $\langle 2 \rangle 1$
  $\langle 2 \rangle 4.$ $\exists\, i \in 1 \,..\, LenQ : Q[i] = s$ BY $\langle 1 \rangle 4$, $\langle 2 \rangle 3$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 2 \rangle 4$, $SMTT(10)$
$\langle 1 \rangle 6.$ $S = \{\} \;\Rightarrow\; Q = \langle\rangle$
  $\langle 2 \rangle 1.$ SUFFICES ASSUME $S = \{\}$ PROVE $Q = \langle\rangle$ OBVIOUS
  $\langle 2 \rangle 2.$ $LenQ = 0$
    $\langle 3 \rangle 1.$ SUFFICES ASSUME $LenQ \neq 0$ PROVE FALSE OBVIOUS
    $\langle 3 \rangle 2.$ $LenQ > 0$ BY $\langle 1 \rangle 2$, $\langle 3 \rangle 1$, $SMTT(10)$
    $\langle 3 \rangle$ PICK $i \in 1 \,..\, LenQ :$ TRUE BY $\langle 1 \rangle 2$, $\langle 3 \rangle 2$, $SMTT(10)$
    $\langle 3 \rangle 3.$ $Q[i] \in \{\}$ BY $\langle 1 \rangle 3$, $\langle 2 \rangle 1$
    $\langle 3 \rangle$ QED BY $\langle 3 \rangle 3$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 2 \rangle 2$, $EmptySeq$ DEF $LenQ$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 5$, $\langle 1 \rangle 6$

Removing one element from an exact sequence yields a smaller exact sequence.

THEOREM $ExactSeqRemoveAt \triangleq$
  ASSUME
    NEW S,
    NEW Q, $IsExactSeqFor(Q, S)$,
    NEW $n \in 1 \,..\, Len(Q)$
  PROVE
  $IsExactSeqFor(RemoveAt(Q, n), S \setminus \{Q[n]\})$
PROOF
$\langle 1 \rangle$ DEFINE $s0 \triangleq Q[n]$
$\langle 1 \rangle$ DEFINE $S1 \triangleq S \setminus \{s0\}$
$\langle 1 \rangle$ DEFINE $Q1 \triangleq RemoveAt(Q, n)$
$\langle 1 \rangle$ DEFINE $LenQ \triangleq Len(Q)$
$\langle 1 \rangle$ DEFINE $LenQ1 \triangleq Len(Q1)$
$\langle 1 \rangle$ HIDE DEF $s0$, $S1$, $Q1$, $LenQ$, $LenQ1$

First establish some preliminary facts.

$\langle 1 \rangle 1.\ Q \in Seq(S)$ BY DEF $IsExactSeqFor$
$\langle 1 \rangle 2.\ ExactSeq\_Each(Q, S)$ BY DEF $IsExactSeqFor$
$\langle 1 \rangle 3.\ ExactSeq\_Once(Q)$ BY DEF $IsExactSeqFor$
$\langle 1 \rangle 4.\ Q \in [1 \,..\, LenQ \rightarrow S]$ BY $\langle 1 \rangle 1$, $LenAxiom$ DEF $LenQ$
$\langle 1 \rangle 5.\ LenQ \in Nat$ BY $\langle 1 \rangle 1$, $LenInNat$ DEF $LenQ$
$\langle 1 \rangle 6.\ n \in 1 \,..\, LenQ$ BY DEF $LenQ$
$\langle 1 \rangle 7.\ LenQ \geq 1$ BY $\langle 1 \rangle 5$, $\langle 1 \rangle 6$, $SMTT(10)$
$\langle 1 \rangle 8.\ s0 \ \in S$ BY $\langle 1 \rangle 4$, $\langle 1 \rangle 6$ DEF $s0$
$\langle 1 \rangle 9.\ Q1 \in Seq(S)$ BY $\langle 1 \rangle 1$, $RemoveAtProperties$ DEF $Q1$
$\langle 1 \rangle 10.\ LenQ1 \in Nat$ BY $\langle 1 \rangle 9$, $LenInNat$ DEF $LenQ1$
$\langle 1 \rangle 11.\ Q1 \in [1 \,..\, LenQ1 \rightarrow S]$ BY $\langle 1 \rangle 9$, $LenAxiom$ DEF $LenQ1$
$\langle 1 \rangle 12.\ LenQ1 = LenQ - 1$ BY $\langle 1 \rangle 1$, $RemoveAtProperties$ DEF $LenQ$, $Q1$, $LenQ1$

Now proceed to prove each of the three conjuncts in $IsExactSeqFor$.

$\langle 1 \rangle 13.\ Q1 \in Seq(S1)$
  $\langle 2 \rangle 1.\ \forall q1 \in 1 \,..\, LenQ1 : Q1[q1] \in S1$
    $\langle 3 \rangle 1.$ SUFFICES ASSUME $\exists q1 \in 1 \,..\, LenQ1 : Q1[q1] \notin S1$ PROVE FALSE OBVIOUS
    $\langle 3 \rangle 2.$ PICK $q1 : q1 \in 1 \,..\, LenQ1 \wedge Q1[q1] \notin S1$ BY $\langle 3 \rangle 1$
    $\langle 3 \rangle 3.\ Q1[q1] = s0$
      $\langle 4 \rangle 1.\ Q1[q1] \in S$ BY $\langle 1 \rangle 11$, $\langle 3 \rangle 2$
      $\langle 4 \rangle$ QED BY $\langle 1 \rangle 8$, $\langle 3 \rangle 2$, $\langle 4 \rangle 1$ DEF $S1$
    $\langle 3 \rangle 4.\ \exists q \in 1 \,..\, LenQ : q \neq n \wedge Q[q] = s0$
      $\langle 4 \rangle 1.\ RemoveAt\_EachBackward(Q, n)$ BY $\langle 1 \rangle 1$, $RemoveAtProperties$
      $\langle 4 \rangle$ QED BY $\langle 3 \rangle 2$, $\langle 3 \rangle 3$, $\langle 4 \rangle 1$ DEF $RemoveAt\_EachBackward$, $LenQ$, $LenQ1$, $Q1$
    $\langle 3 \rangle 7.\ \neg ExactSeq\_Once(Q)$ BY $\langle 1 \rangle 6$, $\langle 3 \rangle 4$ DEF $ExactSeq\_Once$, $s0$, $LenQ$
    $\langle 3 \rangle$ QED BY $\langle 1 \rangle 3$, $\langle 3 \rangle 7$
  $\langle 2 \rangle 2.\ Q1 \in [1 \,..\, LenQ1 \rightarrow S1]$ BY $\langle 1 \rangle 11$, $\langle 2 \rangle 1$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 10$, $\langle 2 \rangle 2$, $IsASeq$

$\langle 1 \rangle 14.\ ExactSeq\_Each(Q1, S1)$
  $\langle 2 \rangle 1.$ SUFFICES ASSUME NEW $s1$, $s1 \in S1$ PROVE $\exists q1 \in 1 \,..\, LenQ1 : Q1[q1] = s1$
      BY DEF $ExactSeq\_Each$, $LenQ1$
  $\langle 2 \rangle 2.\ s1 \in S$ BY $\langle 2 \rangle 1$ DEF $S1$
  $\langle 2 \rangle 3.$ PICK $q : q \in 1 \,..\, LenQ \wedge Q[q] = s1$ BY $\langle 1 \rangle 2$, $\langle 2 \rangle 2$ DEF $ExactSeq\_Each$, $LenQ$
  $\langle 2 \rangle 4.\ s1 \neq s0$ BY $\langle 2 \rangle 1$ DEF $S1$
  $\langle 2 \rangle 5.\ q \neq n$ BY $\langle 2 \rangle 3$, $\langle 2 \rangle 4$ DEF $s0$
  $\langle 2 \rangle 6.\ RemoveAt\_EachForward(Q, n)$ BY $\langle 1 \rangle 1$, $RemoveAtProperties$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 3$, $\langle 2 \rangle 5$, $\langle 2 \rangle 6$ DEF $RemoveAt\_EachForward$, $LenQ$, $LenQ1$, $Q1$

$\langle 1 \rangle 15.\ ExactSeq\_Once(Q1)$
  $\langle 2 \rangle 1.$ SUFFICES ASSUME $\neg ExactSeq\_Once(Q1)$ PROVE FALSE OBVIOUS
  $\langle 2 \rangle 2.$ PICK $q1a, q1b \in 1 \,..\, LenQ1 : q1a \neq q1b \wedge Q1[q1a] = Q1[q1b]$
      BY $\langle 2 \rangle 1$ DEF $ExactSeq\_Once$, $LenQ1$
  $\langle 2 \rangle 3.\ \exists qa, qb \in 1 \,..\, LenQ : qa \neq qb \wedge Q[qa] = Q[qb]$
    $\langle 3 \rangle 1.$ PICK $qa, qb \in 1 \,..\, LenQ$:
        $qa \neq qb \wedge qa \neq n \wedge qb \neq n \wedge Q[qa] = Q1[q1a] \wedge Q[qb] = Q1[q1b]$
      $\langle 4 \rangle 1.\ RemoveAt\_DistinctBackward(Q, n)$ BY $\langle 1 \rangle 1$, $RemoveAtProperties$

$\langle 4\rangle$ QED BY $\langle 2\rangle 2$, $\langle 4\rangle 1$ DEF $RemoveAt\_DistinctBackward$, $LenQ$, $LenQ1$, $Q1$
$\langle 3\rangle 2$. $qa \neq qb \land Q[qa] = Q[qb]$ BY $\langle 3\rangle 1$, $\langle 2\rangle 2$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1$, $\langle 3\rangle 2$
$\langle 2\rangle 4$. $\neg ExactSeq\_Once(Q)$ BY $\langle 2\rangle 3$ DEF $ExactSeq\_Once$, $LenQ$
$\langle 2\rangle$ QED BY $\langle 1\rangle 3$, $\langle 2\rangle 4$

$\langle 1\rangle$ QED BY $\langle 1\rangle 13$, $\langle 1\rangle 14$, $\langle 1\rangle 15$ DEF $IsExactSeqFor$, $Q1$, $S1$, $s0$

Every exact sequence for a given set has the same length.

THEOREM $ExactSeqLength \triangleq$
ASSUME
NEW S,
NEW Q, $IsExactSeqFor(Q, S)$,
NEW R, $IsExactSeqFor(R, S)$
PROVE
$Len(Q) = Len(R)$
PROOF

A counterexample to this theorem is a set $S1$ with exact sequences $Q1$ and $R1$ that have different lengths.

$\langle 1\rangle$ DEFINE $IsCounterexample(S1, Q1, R1) \triangleq$
$\land IsExactSeqFor(Q1, S1)$
$\land IsExactSeqFor(R1, S1)$
$\land Len(Q1) \neq Len(R1)$
$\langle 1\rangle$ HIDE DEF $IsCounterexample$

Let N be the set of all natural numbers n such that there is a counterexample and the length of one of the exact sequences is n.

$\langle 1\rangle$ DEFINE $N \triangleq \{n \in Nat : \exists S1, Q1, R1 : IsCounterexample(S1, Q1, R1) \land n = Len(Q1)\}$
$\langle 1\rangle$ HIDE DEF N
$\langle 1\rangle 1$. SUFFICES $N = \{\}$
$\langle 2\rangle 1$. SUFFICES ASSUME $Len(Q) \neq Len(R)$ PROVE FALSE OBVIOUS
$\langle 2\rangle 2$. $IsCounterexample(S, Q, R)$ BY $\langle 2\rangle 1$ DEF $IsCounterexample$
$\langle 2\rangle 3$. $Q \in Seq(S)$ BY DEF $IsExactSeqFor$
$\langle 2\rangle 4$. $Len(Q) \in Nat$ BY $\langle 2\rangle 3$, $LenInNat$
$\langle 2\rangle 5$. $Len(Q) \in N$ BY $\langle 2\rangle 2$, $\langle 2\rangle 4$ DEF N
$\langle 2\rangle$ QED BY $\langle 1\rangle 1$, $\langle 2\rangle 5$
$\langle 1\rangle 2$. SUFFICES ASSUME $N \neq \{\}$ PROVE FALSE OBVIOUS

If there is a counterexample, there must be a smallest one.

$\langle 1\rangle 3$. PICK $n \in N : \forall m \in N : n \leq m$ BY $\langle 1\rangle 2$, $NatWellFounded$ DEF N
$\langle 1\rangle 4$. PICK $S1, Q1, R1 : IsCounterexample(S1, Q1, R1) \land n = Len(Q1)$ BY $\langle 1\rangle 3$ DEF N
$\langle 1\rangle$ DEFINE $LenQ1 \triangleq Len(Q1)$
$\langle 1\rangle$ DEFINE $LenR1 \triangleq Len(R1)$

$\langle 1 \rangle$ HIDE DEF $LenQ1$, $LenR1$

Based on this "smallest" counterexample, we will construct a smaller one, thus establishing a contradiction.

First we establish various useful facts about $S1$, $Q1$, and $R1$.

$\langle 1 \rangle 5.$ $IsExactSeqFor(Q1, S1)$ BY $\langle 1 \rangle 4$ DEF $IsCounterexample$
$\langle 1 \rangle 6.$ $Q1 \in Seq(S1)$ BY $\langle 1 \rangle 5$ DEF $IsExactSeqFor$
$\langle 1 \rangle 7.$ $ExactSeq\_Each(Q1, S1)$ BY $\langle 1 \rangle 5$ DEF $IsExactSeqFor$
$\langle 1 \rangle 8.$ $LenQ1 \in Nat$ BY $\langle 1 \rangle 6$, $LenInNat$ DEF $LenQ1$

$\langle 1 \rangle 9.$ $IsExactSeqFor(R1, S1)$ BY $\langle 1 \rangle 4$ DEF $IsCounterexample$
$\langle 1 \rangle 10.$ $R1 \in Seq(S1)$ BY $\langle 1 \rangle 9$ DEF $IsExactSeqFor$
$\langle 1 \rangle 11.$ $ExactSeq\_Each(R1, S1)$ BY $\langle 1 \rangle 9$ DEF $IsExactSeqFor$
$\langle 1 \rangle 12.$ $LenR1 \in Nat$ BY $\langle 1 \rangle 10$, $LenInNat$ DEF $LenR1$

$\langle 1 \rangle 13.$ $LenQ1 \neq LenR1$ BY $\langle 1 \rangle 4$ DEF $LenQ1$, $LenR1$, $IsCounterexample$
$\langle 1 \rangle 14.$ $n = LenQ1$ BY $\langle 1 \rangle 4$ DEF $LenQ1$

$\langle 1 \rangle 15.$ $S1 \neq \{\}$
  $\langle 2 \rangle 1.$ SUFFICES ASSUME $S1 = \{\}$ PROVE FALSE OBVIOUS
  $\langle 2 \rangle 2.$ $Q1 = \langle\rangle$ BY $\langle 1 \rangle 5$, $\langle 2 \rangle 1$, $ExactSeqEmpty$
  $\langle 2 \rangle 3.$ $R1 = \langle\rangle$ BY $\langle 1 \rangle 9$, $\langle 2 \rangle 1$, $ExactSeqEmpty$
  $\langle 2 \rangle 4.$ $Q1 = R1$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$
  $\langle 2 \rangle 5.$ $LenQ1 = LenR1$ BY $\langle 2 \rangle 4$ DEF $LenQ1$, $LenR1$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 13$, $\langle 2 \rangle 5$

Since $S1 \neq \{\}$, we pick some element $s1 \in S1$ and remove it from $S1$, $Q1$, and $R1$. This creates a smaller counterexample.

$\langle 1 \rangle 16.$ PICK $s1 : s1 \in S1$ BY $\langle 1 \rangle 15$
$\langle 1 \rangle 17.$ PICK $q1 : q1 \in 1 \,..\, LenQ1 \land Q1[q1] = s1$ BY $\langle 1 \rangle 7$, $\langle 1 \rangle 16$ DEF $ExactSeq\_Each$, $LenQ1$
$\langle 1 \rangle 18.$ PICK $r1 : r1 \in 1 \,..\, LenR1 \land R1[r1] = s1$ BY $\langle 1 \rangle 11$, $\langle 1 \rangle 16$ DEF $ExactSeq\_Each$, $LenR1$

$\langle 1 \rangle$ DEFINE $S2 \triangleq S1 \setminus \{s1\}$
$\langle 1 \rangle$ DEFINE $Q2 \triangleq RemoveAt(Q1, q1)$
$\langle 1 \rangle$ DEFINE $R2 \triangleq RemoveAt(R1, r1)$
$\langle 1 \rangle$ DEFINE $LenQ2 \triangleq Len(Q2)$
$\langle 1 \rangle$ DEFINE $LenR2 \triangleq Len(R2)$
$\langle 1 \rangle$ HIDE DEF $S2$, $Q2$, $R2$, $LenQ2$, $LenR2$

$\langle 1 \rangle 19.$ $LenQ2 = LenQ1 - 1$ BY $\langle 1 \rangle 6$, $\langle 1 \rangle 17$, $RemoveAtProperties$ DEF $Q2$, $LenQ2$, $LenQ1$
$\langle 1 \rangle 20.$ $LenR2 = LenR1 - 1$ BY $\langle 1 \rangle 10$, $\langle 1 \rangle 18$, $RemoveAtProperties$ DEF $R2$, $LenR2$, $LenR1$

$\langle 1 \rangle 21.$ $IsExactSeqFor(Q2, S2)$ BY $\langle 1 \rangle 5$, $\langle 1 \rangle 17$, $ExactSeqRemoveAt$ DEF $Q2$, $S2$, $LenQ1$
$\langle 1 \rangle 22.$ $IsExactSeqFor(R2, S2)$ BY $\langle 1 \rangle 9$, $\langle 1 \rangle 18$, $ExactSeqRemoveAt$ DEF $R2$, $S2$, $LenR1$

$\langle 1 \rangle 23.$ $Q2 \in Seq(S2)$ BY $\langle 1 \rangle 21$ DEF $IsExactSeqFor$
$\langle 1 \rangle 24.$ $LenQ2 \in Nat$ BY $\langle 1 \rangle 23$, $LenInNat$ DEF $LenQ2$
$\langle 1 \rangle 25.$ $LenQ2 \neq LenR2$ BY $\langle 1 \rangle 8$, $\langle 1 \rangle 12$, $\langle 1 \rangle 13$, $\langle 1 \rangle 19$, $\langle 1 \rangle 20$, $SMTT(10)$

$\langle 1 \rangle 26.$ $IsCounterexample(S2, Q2, R2)$ BY $\langle 1 \rangle 21$, $\langle 1 \rangle 22$, $\langle 1 \rangle 25$ DEF $LenQ2$, $LenR2$, $IsCounterexample$
$\langle 1 \rangle 27.$ $LenQ2 \in N$ BY $\langle 1 \rangle 24$, $\langle 1 \rangle 26$ DEF $LenQ2$, N

$\langle 1 \rangle 28.$ $\neg(LenQ1 \leq LenQ2)$ BY $\langle 1 \rangle 8$, $\langle 1 \rangle 19$, $SMTT(10)$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 3$, $\langle 1 \rangle 14$, $\langle 1 \rangle 27$, $\langle 1 \rangle 28$

C.7 Facts about partial orders

MODULE $NaiadClockProofPartialOrders$

EXTENDS $NaiadClockProofExactSeqs$

Facts about partial orders.

This really ought to be a library of theorems.

Although most of these theorems follow immediately from the definition, appealing to the theorem name in subsequent proofs rather than to the definition makes the subsequent proofs easier to understand.

A partial order is reflexive. This follows immediately from the definition.

THEOREM $PartialOrderReflexive \triangleq$
  ASSUME
    NEW $leq \in PointRelationType$, $IsPartialOrder(leq)$,
    NEW $s \in Point$
  PROVE
  LET
    $a \preceq b \triangleq leq[a][b]$
    $a \prec b \triangleq a \preceq b \land a \neq b$
  IN
  $s \preceq s$
PROOF
  $\langle 1 \rangle$ QED BY DEF $IsPartialOrder$

A partial order is antisymmetric. This follows immediately from the definition.

THEOREM $PartialOrderAntisymmetric \triangleq$
  ASSUME
    NEW $leq \in PointRelationType$, $IsPartialOrder(leq)$,
    NEW $s \in Point$,
    NEW $t \in Point$
  PROVE

LET
$a \preceq b \triangleq leq[a][b]$
$a \prec b \triangleq a \preceq b \land a \neq b$
IN
$s \preceq t \land t \preceq s \Rightarrow s = t$
PROOF
$\langle 1 \rangle$ QED BY DEF $IsPartialOrder$

A partial order is transitive. This follows immediately from the definition.

THEOREM $PartialOrderTransitive \triangleq$
ASSUME
NEW $leq \in PointRelationType$, $IsPartialOrder(leq)$,
NEW $s \in Point$,
NEW $t \in Point$,
NEW $u \in Point$
PROVE
LET
$a \preceq b \triangleq leq[a][b]$
$a \prec b \triangleq a \preceq b \land a \neq b$
IN
$s \preceq t \land t \preceq u \Rightarrow s \preceq u$
PROOF
$\langle 1 \rangle$ QED BY DEF $IsPartialOrder$

A partial order is strictly transitive.

THEOREM $PartialOrderStrictlyTransitive \triangleq$
ASSUME
NEW $leq \in PointRelationType$, $IsPartialOrder(leq)$,
NEW $s \in Point$,
NEW $t \in Point$,
NEW $u \in Point$
PROVE
LET
$a \preceq b \triangleq leq[a][b]$

$a \prec b \triangleq a \preceq b \land a \neq b$
IN
$\land\ s \preceq t \land t \prec u \Rightarrow s \prec u$
$\land\ s \prec t \land t \preceq u \Rightarrow s \prec u$

PROOF
$\langle 1 \rangle$ DEFINE $a \preceq b \triangleq leq[a][b]$
$\langle 1 \rangle$ DEFINE $a \prec b \triangleq a \preceq b \land a \neq b$

$\langle 1 \rangle 1$. SUFFICES ASSUME $s \preceq t,\ t \preceq u,\ s \neq t \lor t \neq u$ PROVE $s \prec u$ OBVIOUS

$\langle 1 \rangle 2$. $s \preceq u$ BY $\langle 1 \rangle 1$, *PartialOrderTransitive*
$\langle 1 \rangle 3$. SUFFICES ASSUME $s = u$ PROVE FALSE BY $\langle 1 \rangle 2$

$\langle 1 \rangle 4$. $u \preceq s$ BY $\langle 1 \rangle 3$, *PartialOrderReflexive*
$\langle 1 \rangle 5$. $u \preceq t$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 4$, *PartialOrderTransitive*
$\langle 1 \rangle 6$. $u = t$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 5$, *PartialOrderAntisymmetric*
$\langle 1 \rangle 7$. $s = t$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 6$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 6$, $\langle 1 \rangle 7$

C.8 Facts about delta vectors

MODULE $NaiadClockProofDeltaVecs$

EXTENDS $NaiadClockProofPartialOrders$

Facts about delta vectors.

Addition of delta vectors is closed.

THEOREM $DeltaVecAddType \triangleq$
ASSUME
NEW $a \in DeltaVecType$,
NEW $b \in DeltaVecType$
PROVE
$DeltaVecAdd(a, b) \in DeltaVecType$
PROOF
$\langle 1 \rangle$ QED BY Isa DEF $DeltaVecType$, $DeltaVecAdd$

Zero is a delta vec.

THEOREM $DeltaVecZeroType \triangleq$
$DeltaVecZero \in DeltaVecType$
PROOF
$\langle 1 \rangle$ QED BY Isa DEF $DeltaVecType$, $DeltaVecZero$

Zero is the identity.

THEOREM $DeltaVecAddZero \triangleq$
ASSUME
NEW $a \in DeltaVecType$
PROVE
$\wedge DeltaVecAdd(a, DeltaVecZero) = a$
$\wedge DeltaVecAdd(DeltaVecZero, a) = a$

PROOF
  $\langle 1 \rangle$ QED BY Isa DEF $DeltaVecType$, $DeltaVecAdd$, $DeltaVecZero$

Addition of delta vectors is commutative.

THEOREM $DeltaVecAddCommutative \triangleq$
  ASSUME
    NEW $a \in DeltaVecType$,
    NEW $b \in DeltaVecType$
  PROVE
  $DeltaVecAdd(a, b) = DeltaVecAdd(b, a)$
PROOF
  $\langle 1 \rangle$ SUFFICES ASSUME NEW $t \in Point$
      PROVE $a[t] + b[t] = b[t] + a[t]$
      BY DEF $DeltaVecAdd$
  $\langle 1 \rangle$ QED BY $SMTT(10)$ DEF $DeltaVecType$

Addition of delta vectors is associative.

THEOREM $DeltaVecAddAssociative \triangleq$
  ASSUME
    NEW $a \in DeltaVecType$,
    NEW $b \in DeltaVecType$,
    NEW $c \in DeltaVecType$
  PROVE
  $DeltaVecAdd(DeltaVecAdd(a, b), c) = DeltaVecAdd(a, DeltaVecAdd(b, c))$
PROOF
  $\langle 1 \rangle$ SUFFICES ASSUME NEW $t \in Point$
      PROVE $(a[t] + b[t]) + c[t] = a[t] + (b[t] + c[t])$
      BY DEF $DeltaVecAdd$
  $\langle 1 \rangle$ QED BY $SMTT(10)$ DEF $DeltaVecType$

Negation of delta vectors is closed.

THEOREM $DeltaVecNegType \triangleq$

ASSUME
NEW $a \in DeltaVecType$
PROVE
$DeltaVecNeg(a) \in DeltaVecType$
PROOF
$\langle 1 \rangle$ QED BY Isa DEF $DeltaVecType$, $DeltaVecNeg$

Negation of a delta vector creates the additive inverse.

THEOREM $DeltaVecAddNeg \triangleq$
ASSUME
NEW $a \in DeltaVecType$
PROVE
$\land DeltaVecAdd(a, DeltaVecNeg(a)) = DeltaVecZero$
$\land DeltaVecAdd(DeltaVecNeg(a), a) = DeltaVecZero$
PROOF
$\langle 1 \rangle$ SUFFICES ASSUME NEW $t \in Point$
PROVE $a[t] + (0 - a[t]) = 0 \land (0 - a[t]) + a[t] = 0$
BY DEF $DeltaVecAdd$, $DeltaVecNeg$, $DeltaVecZero$
$\langle 1 \rangle$ QED BY $SMTT(10)$ DEF $DeltaVecType$

C.9 Facts about summing up sequences of delta vectors

MODULE $NaiadClockProofDeltaVecSeqs$

EXTENDS $NaiadClockProofDeltaVecs$

Facts about summing up sequences of delta vectors.

This really ought to be a library of theorems.

Let $Prop$ be any predicate satisfied by $Zero$ and preserved by Add. Let Q be a sequence of delta vectors in which each element after the first k satisfies $Prop$. Then the skip k sum of Q is a delta vector that satisfies $Prop$.

We define explicit operators to capture the hypothesis and conclusion. Otherwise the provers seem to have difficulty figuring out how to apply this theorem.

$DeltaVecSeqSkipSumProp\_Hypothesis(Prop(\_), Q, k) \triangleq$
$\quad \land Prop(DeltaVecZero)$
$\quad \land \forall a, b \in DeltaVecType : Prop(a) \land Prop(b) \Rightarrow Prop(DeltaVecAdd(a, b))$
$\quad \land Q \in Seq(DeltaVecType)$
$\quad \land k \in Nat$
$\quad \land \forall i \in Nat : k < i \land i \leq Len(Q) \Rightarrow Prop(Q[i])$

$DeltaVecSeqSkipSumProp\_Conclusion(Prop(\_), Q, k) \triangleq$
$\quad \land DeltaVecSeqSkipSum(k, Q) \in DeltaVecType$
$\quad \land Prop(DeltaVecSeqSkipSum(k, Q))$

THEOREM $DeltaVecSeqSkipSumProp \triangleq$
ASSUME NEW $Prop(\_)$, NEW Q, NEW k, $DeltaVecSeqSkipSumProp\_Hypothesis(Prop, Q, k)$
PROVE $DeltaVecSeqSkipSumProp\_Conclusion(Prop, Q, k)$
PROOF
$\langle 1 \rangle$ DEFINE $Type \triangleq DeltaVecType$
$\langle 1 \rangle$ DEFINE $Zero \triangleq DeltaVecZero$
$\langle 1 \rangle$ DEFINE $Add(a, b) \triangleq DeltaVecAdd(a, b)$

$\langle 1 \rangle$ USE DEF $DeltaVecSeqSkipSumProp\_Hypothesis$
$\langle 1 \rangle 1.$ $Prop(Zero)$ OBVIOUS
$\langle 1 \rangle 2.$ $\forall a, b \in Type : Prop(a) \land Prop(b) \Rightarrow Prop(Add(a, b))$ OBVIOUS
$\langle 1 \rangle 3.$ $Q \in Seq(Type)$ OBVIOUS
$\langle 1 \rangle 4.$ $k \in Nat$ OBVIOUS
$\langle 1 \rangle 5.$ $\forall i \in Nat : k < i \land i \leq Len(Q) \Rightarrow Prop(Q[i])$ OBVIOUS
$\langle 1 \rangle$ HIDE DEF $DeltaVecSeqSkipSumProp\_Hypothesis$

$\langle 1 \rangle$ DEFINE $TypeProp(a) \triangleq a \in Type \land Prop(a)$

Show the definition of the recursive function used to define the sum.

$\langle 1 \rangle$ DEFINE $Elem(i) \triangleq DeltaVecSeqSkipSum(k, Q)!:!Elem(i)$
$\langle 1 \rangle$ DEFINE $f0 \triangleq Zero$
$\langle 1 \rangle$ DEFINE $Def(v, i) \triangleq Add(v, Elem(i))$
$\langle 1 \rangle$ DEFINE $f \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0$ ELSE $Def(f[i-1], i)]$
$\langle 1 \rangle$ DEFINE $LenQ \triangleq Len(Q)$
$\langle 1 \rangle 6.$ $LenQ \in Nat$ BY $\langle 1 \rangle 3$, $LenInNat$
$\langle 1 \rangle 7.$ $DeltaVecSeqSkipSum(k, Q) = f[LenQ]$ BY DEF $DeltaVecSeqSkipSum$

$\langle 1 \rangle 8.$ $\forall i \in Nat : f[i] =$ IF $i = 0$ THEN $f0$ ELSE $Def(f[i-1], i)$
  $\langle 2 \rangle$ HIDE DEF $f0$, Def, f
  $\langle 2 \rangle$ SUFFICES $NatInductiveDefConclusion(f, f0, Def)$ BY DEF $NatInductiveDefConclusion$
  $\langle 2 \rangle$ SUFFICES $NatInductiveDefHypothesis(f, f0, Def)$ BY $NatInductiveDef$
  $\langle 2 \rangle$ QED BY DEF $NatInductiveDefHypothesis$, f

Each $Elem$ has $TypeProp$.

$\langle 1 \rangle 9.$ $\forall i \in Nat \setminus \{0\} : TypeProp(Elem(i))$
  $\langle 2 \rangle$ SUFFICES ASSUME NEW $i \in Nat \setminus \{0\}$ PROVE $TypeProp(Elem(i))$ OBVIOUS
  $\langle 2 \rangle$ HIDE DEF $LenQ$
  $\langle 2 \rangle 1.$ CASE $k < i \land i \leq LenQ$
    $\langle 3 \rangle$ SUFFICES $TypeProp(Q[i])$ BY $\langle 2 \rangle 1$ DEF $LenQ$
    $\langle 3 \rangle 1.$ $i \in 1\,..\,LenQ$BY $\langle 2 \rangle 1$, $\langle 1 \rangle 6$, $DotDotDef$, $SMTT(10)$
    $\langle 3 \rangle 2.$ $Q[i] \in Type$ BY $\langle 3 \rangle 1$, $\langle 1 \rangle 3$, $LenAxiom$ DEF $LenQ$
    $\langle 3 \rangle 3.$ $Prop(Q[i])$ BY $\langle 2 \rangle 1$, $\langle 1 \rangle 5$ DEF $LenQ$
    $\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$, $\langle 3 \rangle 3$
  $\langle 2 \rangle 2.$ CASE $\neg(k < i \land i \leq LenQ)$
    $\langle 3 \rangle$ $Elem(i) = Zero$ BY $\langle 2 \rangle 2$ DEF $LenQ$
    $\langle 3 \rangle$ QED BY $\langle 1 \rangle 1$, $DeltaVecZeroType$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$

The sum of the sequence evaluates its recursive function at the length of the sequence. Showing that this satisfies $Prop$ requires induction.

$\langle 1 \rangle 10.$ $TypeProp(f[LenQ])$
  $\langle 2 \rangle$ DEFINE $P(i) \triangleq TypeProp(f[i])$
  $\langle 2 \rangle$ HIDE DEF $LenQ$, f
  $\langle 2 \rangle$ SUFFICES $\forall i \in Nat : P(i)$ BY $\langle 1 \rangle 6$
  $\langle 2 \rangle 1.$ $P(0)$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 8$, $DeltaVecZeroType$ DEF f, $f0$
  $\langle 2 \rangle 2.$ $\forall i \in Nat : P(i) \Rightarrow P(i+1)$
    $\langle 3 \rangle 1.$ SUFFICES ASSUME NEW $i \in Nat$, $P(i)$ PROVE $P(i+1)$ OBVIOUS
    $\langle 3 \rangle 2.$ Trivial facts to help the prover match known facts or proof obligations.
      $\land\ i + 1 \in Nat$
      $\land\ i + 1 \in Nat \setminus \{0\}$
      $\land\ i + 1 \neq 0$
      $\land\ (i + 1) - 1 = i$
      BY $SMTT(10)$
    $\langle 3 \rangle 3.$ $f[i+1] = Add(f[i], Elem(i+1))$ BY $\langle 3 \rangle 2$, $\langle 1 \rangle 8$

$\langle 3\rangle 4.$ $TypeProp(Elem(i+1))$ BY $\langle 3\rangle 2,\ \langle 1\rangle 9$
$\langle 3\rangle$ QED BY $\langle 3\rangle 3,\ \langle 3\rangle 4,\ \langle 3\rangle 1,\ \langle 1\rangle 2,\ DeltaVecAddType$
$\langle 2\rangle$ HIDE DEF P
$\langle 2\rangle$ QED BY ONLY $\langle 2\rangle 1,\ \langle 2\rangle 2,\ NatInduction,\ Isa$
$\langle 1\rangle$ QED BY $\langle 1\rangle 7,\ \langle 1\rangle 10$ DEF $DeltaVecSeqSkipSumProp\_Conclusion$

The skip k sum of a sequence of delta vectors is a delta vector.

THEOREM $DeltaVecSeqSkipSumType \triangleq$
ASSUME
NEW $Q \in Seq(DeltaVecType)$,
NEW $k \in Nat$
PROVE
$DeltaVecSeqSkipSum(k,\ Q) \in DeltaVecType$
PROOF
$\langle 1\rangle$ DEFINE $Prop(a) \triangleq$ TRUE
$\langle 1\rangle$ $DeltaVecSeqSkipSumProp\_Conclusion(Prop,\ Q,\ k)$
$\langle 2\rangle$ $DeltaVecSeqSkipSumProp\_Hypothesis(Prop,\ Q,\ k)$ BY DEF $DeltaVecSeqSkipSumProp\_Hypothesis$
$\langle 2\rangle$ QED BY $DeltaVecSeqSkipSumProp$
$\langle 1\rangle$ QED BY DEF $DeltaVecSeqSkipSumProp\_Conclusion$

The skip k sum of a sequence of zero delta vectors is zero.

THEOREM $DeltaVecSeqSkipSumAllZero \triangleq$
ASSUME
NEW $Q \in Seq(DeltaVecType)$,
NEW $k \in Nat$,
$\forall\, i \in$ DOMAIN $Q : Q[i] = DeltaVecZero$
PROVE
$DeltaVecSeqSkipSum(k,\ Q) = DeltaVecZero$
PROOF
$\langle 1\rangle$ DEFINE $Prop(a) \triangleq a = DeltaVecZero$
$\langle 1\rangle$ HIDE DEF $Prop$

$\langle 1\rangle$ SUFFICES $Prop(DeltaVecSeqSkipSum(k,\ Q))$ BY DEF $Prop$
$\langle 1\rangle$ $DeltaVecSeqSkipSumProp\_Hypothesis(Prop,\ Q,\ k)$
$\langle 2\rangle$ $Prop(DeltaVecZero)$ BY DEF $Prop$
$\langle 2\rangle$ $\forall\, a,\ b \in DeltaVecType : Prop(a) \land Prop(b) \Rightarrow Prop(DeltaVecAdd(a,\ b))$

BY $\mathit{DeltaVecZeroType}, \mathit{DeltaVecAddZero}$ DEF Prop
$\langle 2 \rangle$ DEFINE $\mathit{LenQ} \triangleq \mathit{Len}(Q)$
$\langle 2 \rangle$ $\forall i \in \mathit{Nat} : k < i \land i \leq \mathit{LenQ} \Rightarrow \mathit{Prop}(Q[i])$
$\langle 3 \rangle$ TAKE $i \in \mathit{Nat}$
$\langle 3 \rangle$ HAVE $k < i \land i \leq \mathit{LenQ}$
$\langle 3 \rangle$ SUFFICES $i \in$ DOMAIN Q OBVIOUS
$\langle 3 \rangle$ SUFFICES $i \in 1 \,..\, \mathit{LenQ}$ BY LenDef
$\langle 3 \rangle$ $\mathit{LenQ} \in \mathit{Nat}$ BY $\mathit{LenInNat}$
$\langle 3 \rangle$ HIDE DEF LenQ
$\langle 3 \rangle$ QED BY $\mathit{SMTT}(10)$
$\langle 2 \rangle$ HIDE DEF Prop
$\langle 2 \rangle$ QED BY DEF $\mathit{DeltaVecSeqSkipSumProp\_Hypothesis}$
$\langle 1 \rangle$ $\mathit{DeltaVecSeqSkipSumProp\_Conclusion}(\mathit{Prop}, Q, k)$ BY $\mathit{Isa}, \mathit{DeltaVecSeqSkipSumProp}$
$\langle 1 \rangle$ QED BY DEF $\mathit{DeltaVecSeqSkipSumProp\_Conclusion}$

The skip k sum of a sequence of delta vectors is zero when you skip all of the elements of the sequence.

THEOREM $\mathit{DeltaVecSeqSkipSumSkipAll} \triangleq$
ASSUME
NEW $Q \in \mathit{Seq}(\mathit{DeltaVecType})$,
NEW $k \in \mathit{Nat}, k \geq \mathit{Len}(Q)$
PROVE
$\mathit{DeltaVecSeqSkipSum}(k, Q) = \mathit{DeltaVecZero}$
PROOF

Show the definition of the recursive function used to define the sum.

$\langle 1 \rangle$ DEFINE $\mathit{Elem}(i) \triangleq \mathit{DeltaVecSeqSkipSum}(k, Q)! : !\mathit{Elem}(i)$
$\langle 1 \rangle$ DEFINE $f0 \triangleq \mathit{DeltaVecZero}$
$\langle 1 \rangle$ DEFINE $\mathit{Def}(v, i) \triangleq \mathit{DeltaVecAdd}(v, \mathit{Elem}(i))$
$\langle 1 \rangle$ DEFINE $f \triangleq$ CHOOSE $f : f = [i \in \mathit{Nat} \mapsto$ IF $i = 0$ THEN $f0$ ELSE $\mathit{Def}(f[i-1], i)]$
$\langle 1 \rangle$ DEFINE $\mathit{LenQ} \triangleq \mathit{Len}(Q)$
$\langle 1 \rangle 1.$ $\mathit{LenQ} \in \mathit{Nat}$ BY $\mathit{LenInNat}$
$\langle 1 \rangle 2.$ $\mathit{DeltaVecSeqSkipSum}(k, Q) = f[\mathit{LenQ}]$ BY DEF $\mathit{DeltaVecSeqSkipSum}$

$\langle 1 \rangle 3.$ $\forall i \in \mathit{Nat} : f[i] =$ IF $i = 0$ THEN $f0$ ELSE $\mathit{Def}(f[i-1], i)$
$\langle 2 \rangle$ HIDE DEF $f0, \mathit{Def}, f$
$\langle 2 \rangle$ SUFFICES $\mathit{NatInductiveDefConclusion}(f, f0, \mathit{Def})$ BY DEF $\mathit{NatInductiveDefConclusion}$
$\langle 2 \rangle$ SUFFICES $\mathit{NatInductiveDefHypothesis}(f, f0, \mathit{Def})$ BY $\mathit{NatInductiveDef}$
$\langle 2 \rangle$ QED BY DEF $\mathit{NatInductiveDefHypothesis}, f$

The sum of the sequence evaluates its recursive function at the length of the sequence. Showing that this is zero requires induction.

$\langle 1 \rangle 4.$ $f[\mathit{LenQ}] = \mathit{DeltaVecZero}$
$\langle 2 \rangle$ DEFINE $P(i) \triangleq f[i] = \mathit{DeltaVecZero}$

$\langle 2 \rangle$ HIDE DEF $LenQ,\ f$
$\langle 2 \rangle$ SUFFICES $\forall\, i \in Nat : P(i)$ BY $\langle 1 \rangle 1,\ SMTT(10)$
$\langle 2 \rangle 1.\ P(0)$ BY $\langle 1 \rangle 3$ DEF $f,\ f0$
$\langle 2 \rangle 2.$ ASSUME NEW $i \in Nat,\ P(i)$ PROVE $P(i+1)$
$\langle 3 \rangle 1.\ i+1 \in Nat \wedge i+1 \neq 0 \wedge (i+1)-1 = i$ BY $SMTT(10)$
$\langle 3 \rangle 2.\ f[i+1] = DeltaVecAdd(f[i],\ Elem(i+1))$ BY $\langle 1 \rangle 3,\ \langle 3 \rangle 1$
$\langle 3 \rangle 3.\ Elem(i+1) = DeltaVecZero$
$\langle 4 \rangle$ SUFFICES $\neg(k < i+1 \wedge i+1 \leq LenQ)$ BY DEF $LenQ$
$\langle 4 \rangle 1.\ k \geq LenQ$ BY DEF $LenQ$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1,\ \langle 1 \rangle 1,\ SMTT(10)$
$\langle 3 \rangle 4.\ f[i] \in DeltaVecType$ BY $\langle 2 \rangle 2,\ DeltaVecZeroType$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 2,\ \langle 3 \rangle 3,\ \langle 3 \rangle 4,\ \langle 2 \rangle 2,\ DeltaVecAddZero$
$\langle 2 \rangle$ HIDE DEF P
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 2,\ NatInduction,\ Isa$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2,\ \langle 1 \rangle 4$

The skip k sum of an empty sequence of delta vectors is zero. This is a simple corollary of the previous theorem.

THEOREM $DeltaVecSeqSkipSumEmpty \triangleq$
ASSUME
NEW $Q \in Seq(DeltaVecType),\quad Q = \langle\rangle,$
NEW $k \in Nat$
PROVE
$DeltaVecSeqSkipSum(k,\ Q) = DeltaVecZero$
PROOF
$\langle 1 \rangle$ DEFINE $LenQ \triangleq Len(Q)$
$\langle 1 \rangle 1.\ LenQ = 0$ BY $EmptySeq$
$\langle 1 \rangle 2.\ k \geq LenQ$
$\langle 2 \rangle$ HIDE DEF $LenQ$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 1,\ SMTT(10)$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2,\ DeltaVecSeqSkipSumSkipAll$

The skip k sum of a sequence Q is the same as adding delta to the skip $k+1$ sum, where delta is $Q[k+1]$ if $k+1 \leq Len(Q)$ and $DeltaVecZero$ otherwise.

THEOREM $DeltaVecSeqSkipSumNext \triangleq$
ASSUME

NEW $Q \in Seq(DeltaVecType)$,
NEW $k \in Nat$
PROVE
LET
$delta \triangleq$ IF $k+1 \leq Len(Q)$ THEN $Q[k+1]$ ELSE $DeltaVecZero$
$SSk \triangleq DeltaVecSeqSkipSum(k, Q)$
$SSk1 \triangleq DeltaVecSeqSkipSum(k+1, Q)$
IN
$SSk = DeltaVecAdd(SSk1, delta)$
PROOF
$\langle 1 \rangle$ DEFINE $delta \triangleq$ IF $k+1 \leq Len(Q)$ THEN $Q[k+1]$ ELSE $DeltaVecZero$
$\langle 1 \rangle$ DEFINE $SSk \triangleq DeltaVecSeqSkipSum(k, Q)$
$\langle 1 \rangle$ DEFINE $SSk1 \triangleq DeltaVecSeqSkipSum(k+1, Q)$

$XXXa$ definitions are related to the skip k sum.

$\langle 1 \rangle$ DEFINE $Qa \triangleq Q$
$\langle 1 \rangle$ DEFINE $Elema(i) \triangleq DeltaVecSeqSkipSum(k, Qa)!:!Elem(i)$
$\langle 1 \rangle$ DEFINE $f0a \triangleq DeltaVecZero$
$\langle 1 \rangle$ DEFINE $Defa(v, i) \triangleq DeltaVecAdd(v, Elema(i))$
$\langle 1 \rangle$ DEFINE $fa \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0a$ ELSE $Defa(f[i-1], i)]$
$\langle 1 \rangle$ DEFINE $LenQa \triangleq Len(Qa)$
$\langle 1 \rangle 1.$ $Qa \in Seq(DeltaVecType)$ OBVIOUS
$\langle 1 \rangle 2.$ $Qa \in [1 \,..\, LenQa \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 1$, $LenAxiom$
$\langle 1 \rangle 3.$ $LenQa \in Nat$ BY $LenInNat$
$\langle 1 \rangle 4.$ $DeltaVecSeqSkipSum(k, Qa) = fa[LenQa]$ BY DEF $DeltaVecSeqSkipSum$

$\langle 1 \rangle 5.$ $\forall i \in Nat : fa[i] =$ IF $i = 0$ THEN $f0a$ ELSE $Defa(fa[i-1], i)$
$\langle 2 \rangle$ HIDE DEF $f0a, Defa, fa$
$\langle 2 \rangle$ SUFFICES $NatInductiveDefConclusion(fa, f0a, Defa)$ BY DEF $NatInductiveDefConclusion$
$\langle 2 \rangle$ SUFFICES $NatInductiveDefHypothesis(fa, f0a, Defa)$ BY $NatInductiveDef$
$\langle 2 \rangle$ QED BY DEF $NatInductiveDefHypothesis, fa$

$XXXb$ definitions are related to the skip $k+1$ sum.

$\langle 1 \rangle$ DEFINE $Qb \triangleq Q$
$\langle 1 \rangle$ DEFINE $Elemb(i) \triangleq DeltaVecSeqSkipSum(k+1, Qb)!:!Elem(i)$
$\langle 1 \rangle$ DEFINE $f0b \triangleq DeltaVecZero$
$\langle 1 \rangle$ DEFINE $Defb(v, i) \triangleq DeltaVecAdd(v, Elemb(i))$
$\langle 1 \rangle$ DEFINE $fb \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0b$ ELSE $Defb(f[i-1], i)]$
$\langle 1 \rangle$ DEFINE $LenQb \triangleq Len(Qb)$
$\langle 1 \rangle 6.$ $Qb \in Seq(DeltaVecType)$ OBVIOUS
$\langle 1 \rangle 7.$ $Qb \in [1 \,..\, LenQb \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 6$, $LenAxiom$
$\langle 1 \rangle 8.$ $LenQb \in Nat$ BY $\langle 1 \rangle 6$, $LenInNat$
$\langle 1 \rangle 9.$ $DeltaVecSeqSkipSum(k+1, Qb) = fb[LenQb]$ BY DEF $DeltaVecSeqSkipSum$

$\langle 1 \rangle 10.$ $\forall i \in Nat : fb[i] =$ IF $i = 0$ THEN $f0b$ ELSE $Defb(fb[i-1], i)$
$\langle 2 \rangle$ HIDE DEF $f0b, Defb, fb$

$\langle 2 \rangle$ SUFFICES $NatInductiveDefConclusion(fb, f0b, Defb)$ BY DEF $NatInductiveDefConclusion$
$\langle 2 \rangle$ SUFFICES $NatInductiveDefHypothesis(fb, f0b, Defb)$ BY $NatInductiveDef$
$\langle 2 \rangle$ QED BY DEF $NatInductiveDefHypothesis, fb$

Lengths are equal.

$\langle 1 \rangle 11.$ $LenQa = LenQb$ OBVIOUS

Suffices to assume $k < LenQa$.

$\langle 1 \rangle 12.$ SUFFICES ASSUME $k < LenQa$ PROVE $SSk = DeltaVecAdd(SSk1, delta)$
$\langle 2 \rangle 1.$ SUFFICES ASSUME $\neg(k < LenQa)$ PROVE $SSk = DeltaVecAdd(SSk1, delta)$ OBVIOUS
$\langle 2 \rangle 2.$ $DeltaVecSeqSkipSum(k, Q) = DeltaVecZero$
$\langle 3 \rangle 1.$ $k \geq LenQa$
$\langle 4 \rangle$ HIDE DEF $LenQa$
$\langle 4 \rangle$ QED BY $\langle 2 \rangle 1, \langle 1 \rangle 3, SMTT(10)$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1, DeltaVecSeqSkipSumSkipAll$
$\langle 2 \rangle 3.$ $DeltaVecSeqSkipSum(k+1, Q) = DeltaVecZero$
$\langle 3 \rangle 1.$ $k+1 \geq LenQa$
$\langle 4 \rangle$ HIDE DEF $LenQa$
$\langle 4 \rangle$ QED BY $\langle 2 \rangle 1, \langle 1 \rangle 3, SMTT(10)$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1, DeltaVecSeqSkipSumSkipAll$
$\langle 2 \rangle 4.$ $delta = DeltaVecZero$
$\langle 3 \rangle 1.$ $\neg(k+1 \leq LenQa)$
$\langle 4 \rangle$ HIDE DEF $LenQa$
$\langle 4 \rangle$ QED BY $\langle 2 \rangle 1, \langle 1 \rangle 3, SMTT(10)$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$
$\langle 2 \rangle 5.$ $DeltaVecAdd(DeltaVecSeqSkipSum(k+1, Q), delta) = DeltaVecZero$
BY $\langle 2 \rangle 3, \langle 2 \rangle 4, DeltaVecAddZero, DeltaVecZeroType$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2, \langle 2 \rangle 5$

delta a delta vec.

$\langle 1 \rangle 13.$ $delta \in DeltaVecType$
$\langle 2 \rangle 1.$ $delta = Q[k+1]$
$\langle 3 \rangle 1.$ $k+1 \leq LenQa$
$\langle 4 \rangle$ HIDE DEF $LenQa$
$\langle 4 \rangle$ QED BY $\langle 1 \rangle 3, \langle 1 \rangle 12, SMTT(10)$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$
$\langle 2 \rangle 2.$ $Q[k+1] \in DeltaVecType$
$\langle 3 \rangle 1.$ $k\ +1 \in 1 \dots LenQa$
$\langle 4 \rangle$ HIDE DEF $LenQa$
$\langle 4 \rangle$ QED BY $\langle 1 \rangle 3, \langle 1 \rangle 12, DotDotDef, SMTT(10)$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1, LenAxiom$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1, \langle 2 \rangle 2$

Each *Elema* is a delta vec.

$\langle 1 \rangle 14.$ ASSUME NEW $i \in Nat \setminus \{0\}$ PROVE $Elema(i) \in DeltaVecType$
$\langle 2 \rangle 1.$ CASE $k < i \land i \leq LenQa$

$\langle 3\rangle 1.\ i \in 1\,..\,LenQa$
$\langle 4\rangle$ HIDE DEF $LenQa$
$\langle 4\rangle$ QED BY $\langle 2\rangle 1,\ \langle 1\rangle 3,\ DotDotDef,\ SMTT(10)$
$\langle 3\rangle 2.\ Qa[i] \in DeltaVecType$ BY $\langle 3\rangle 1,\ \langle 1\rangle 6,\ LenAxiom$
$\langle 3\rangle$ QED BY $\langle 3\rangle 2,\ \langle 2\rangle 1$
$\langle 2\rangle 2.$ CASE $\neg(k < i \land i \leq LenQa)$
$\langle 3\rangle 1.\ Elema(i) = DeltaVecZero$ BY $\langle 2\rangle 2$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ DeltaVecZeroType,\ DeltaVecAddZero$
$\langle 2\rangle$ QED BY $\langle 2\rangle 1,\ \langle 2\rangle 2$

Each *Elemb* is a delta vec.

$\langle 1\rangle 15.$ ASSUME NEW $i \in Nat \setminus \{0\}$ PROVE $Elemb(i) \in DeltaVecType$
$\langle 2\rangle 1.$ CASE $k + 1 < i \land i \leq LenQb$
$\langle 3\rangle 1.\ i \in 1\,..\,LenQb$
$\langle 4\rangle$ HIDE DEF $LenQb$
$\langle 4\rangle$ QED BY $\langle 2\rangle 1,\ \langle 1\rangle 8,\ DotDotDef,\ SMTT(10)$
$\langle 3\rangle 2.\ Qb[i] \in DeltaVecType$ BY $\langle 3\rangle 1,\ \langle 1\rangle 6,\ LenAxiom$
$\langle 3\rangle$ QED BY $\langle 3\rangle 2,\ \langle 2\rangle 1$
$\langle 2\rangle 2.$ CASE $\neg(k + 1 < i \land i \leq LenQb)$
$\langle 3\rangle 1.\ Elemb(i) \quad = DeltaVecZero$ BY $\langle 2\rangle 2$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ DeltaVecZeroType,\ DeltaVecAddZero$
$\langle 2\rangle$ QED BY $\langle 2\rangle 1,\ \langle 2\rangle 2$

Each $Elema(i) = Elemb(i)$ for all $i > 0$ except $k + 1$, where we have $Elema(k + 1) = Elemb(k + 1) + delta$.

$\langle 1\rangle 16.$ ASSUME NEW $i \in Nat \setminus \{0\}$
PROVE $Elema(i) =$ IF $i = k + 1$ THEN $DeltaVecAdd(Elemb(i),\ delta)$ ELSE $Elemb(i)$
$\langle 2\rangle 1.$ ASSUME $i = k + 1$ PROVE $Elema(i) = DeltaVecAdd(Elemb(i),\ delta)$
$\langle 3\rangle 1.\ Elema(k + 1) = delta$
$\langle 4\rangle 1.\ k + 1 \leq LenQa$
$\langle 5\rangle$ HIDE DEF $LenQa$
$\langle 5\rangle$ QED BY $\langle 1\rangle 3,\ \langle 1\rangle 12,\ SMTT(10)$
$\langle 4\rangle 2.\ Elema(k + 1) = Q[k + 1]$
$\langle 5\rangle 1.\ k < k \ \ + 1$ BY $SMTT(10)$
$\langle 5\rangle$ QED BY $\langle 5\rangle 1,\ \langle 4\rangle 1$
$\langle 4\rangle 3.\ delta = Q[k + 1]$ BY $\langle 4\rangle 1$
$\langle 4\rangle$ QED BY $\langle 4\rangle 2,\ \langle 4\rangle 3$
$\langle 3\rangle 2.\ Elemb(k + 1) = DeltaVecZero$
$\langle 4\rangle 1.\ \neg(k + 1 < k + 1)$ BY $SMTT(10)$
$\langle 4\rangle$ QED BY $\langle 4\rangle 1$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ \langle 3\rangle 2,\ \langle 2\rangle 1,\ \langle 1\rangle 13,\ DeltaVecAddZero$
$\langle 2\rangle 2.$ ASSUME $i \neq k + 1$ PROVE $Elema(i) = Elemb(i)$
$\langle 3\rangle$ HIDE DEF $Qa,\ Qb,\ LenQa,\ LenQb,\ Elema,\ Elemb$
$\langle 3\rangle 1.$ CASE $k < i \land i \leq LenQa$
$\langle 4\rangle 1.\ k < i \land i \leq LenQa \land i \in 1\,..\,LenQa$ BY $\langle 3\rangle 1,\ \langle 1\rangle 3,\ DotDotDef,\ SMTT(10)$
$\langle 4\rangle 2.\ k + 1 < i \land i \leq LenQb \land i \in 1\,..\,LenQb$ BY $\langle 4\rangle 1,\ \langle 1\rangle 11,\ \langle 2\rangle 2,\ SMTT(10)$
$\langle 4\rangle 3.\ Elema(i) = Q[i]$ BY $\langle 4\rangle 1$ DEF $LenQa,\ Elema,\ Qa$

$\langle 4\rangle 4.\ Elemb(i) = Q[i]$ BY $\langle 4\rangle 2$ DEF $LenQb,\ Elemb,\ Qb$
$\langle 4\rangle$ QED BY $\langle 4\rangle 3,\ \langle 4\rangle 4$
$\langle 3\rangle 2.$ CASE $\neg(k < i \land i \leq LenQa)$
$\langle 4\rangle 1.\ \neg(k < i \land i \leq LenQa)$ BY $\langle 3\rangle 2$
$\langle 4\rangle 2.\ \neg(k+1 < i \land i \leq LenQb)$ BY $\langle 4\rangle 1,\ \langle 1\rangle 11,\ \langle 2\rangle 2,\ SMTT(10)$
$\langle 4\rangle 3.\ Elema(i) = DeltaVecZero$ BY $\langle 4\rangle 1$ DEF $LenQa,\ Elema$
$\langle 4\rangle 4.\ Elemb(i) = DeltaVecZero$ BY $\langle 4\rangle 2$ DEF $LenQb,\ Elemb$
$\langle 4\rangle$ QED BY $\langle 4\rangle 3,\ \langle 4\rangle 4$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ \langle 3\rangle 2$
$\langle 2\rangle$ QED BY $\langle 2\rangle 1,\ \langle 2\rangle 2$

$fa[i]$ is a delta vector

$\langle 1\rangle 17.\ \forall\, i \in Nat : fa[i] \in DeltaVecType$
$\langle 2\rangle$ DEFINE $P(i) \triangleq fa[i] \in DeltaVecType$
$\langle 2\rangle$ HIDE DEF $LenQa,\ LenQb,\ fa,\ fb$
$\langle 2\rangle$ SUFFICES $\forall\, i \in Nat : P(i)$ OBVIOUS
$\langle 2\rangle 1.\ P(0)$
$\langle 3\rangle 1.\ fa[0] = DeltaVecZero$ BY $\langle 1\rangle 5$ DEF $fa,\ f0a$
$\langle 3\rangle 2.\ fa[0] \in DeltaVecType$ BY $\langle 3\rangle 1,\ DeltaVecZeroType,\ DeltaVecAddZero$
$\langle 3\rangle$ QED BY $\langle 3\rangle 2$
$\langle 2\rangle 2.\ \forall\, i \in Nat : P(i) \Rightarrow P(i+1)$
$\langle 3\rangle 1.$ SUFFICES ASSUME NEW $i \in Nat,\ P(i)$ PROVE $P(i+1)$ OBVIOUS
$\langle 3\rangle 2.$ Trivial facts to help the prover match known facts or proof obligations.
$\land\ i+1 \in Nat$
$\land\ i+1 \in Nat \setminus \{0\}$
$\land\ i+1 \neq 0$
$\land\ (i+1)-1 = i$
BY $SMTT(10)$
$\langle 3\rangle 3.\ fa[i+1] = DeltaVecAdd(fa[i],\ Elema(i+1))$ BY $\langle 3\rangle 2,\ \langle 1\rangle 5$
$\langle 3\rangle 4.\ fa[i] \in DeltaVecType$ BY $\langle 3\rangle 1$
$\langle 3\rangle 5.\ Elema(i+1) \in DeltaVecType$ BY $\langle 3\rangle 2,\ \langle 1\rangle 14$
$\langle 3\rangle 6.\ fa[i+1] \in DeltaVecType$ BY $\langle 3\rangle 3,\ \langle 3\rangle 4,\ \langle 3\rangle 5,\ DeltaVecAddType$
$\langle 3\rangle$ QED BY $\langle 3\rangle 6$
$\langle 2\rangle$ HIDE DEF P
$\langle 2\rangle$ QED BY ONLY $\langle 2\rangle 1,\ \langle 2\rangle 2,\ NatInduction,\ Isa$

$fb[i]$ is a delta vector

$\langle 1\rangle 18.\ \forall\, i \in Nat : fb[i] \in DeltaVecType$
$\langle 2\rangle$ DEFINE $P(i) \triangleq fb[i] \in DeltaVecType$
$\langle 2\rangle$ HIDE DEF $LenQa,\ LenQb,\ fa,\ fb$
$\langle 2\rangle$ SUFFICES $\forall\, i \in Nat : P(i)$ OBVIOUS
$\langle 2\rangle 1.\ P(0)$
$\langle 3\rangle 1.\ fb[0] = DeltaVecZero$ BY $\langle 1\rangle 10$ DEF $fb,\ f0b$
$\langle 3\rangle 2.\ fb[0] \in DeltaVecType$ BY $\langle 3\rangle 1,\ DeltaVecZeroType,\ DeltaVecAddZero$
$\langle 3\rangle$ QED BY $\langle 3\rangle 2$

$\langle 2 \rangle 2.\ \forall\, i \in Nat : P(i) \Rightarrow P(i+1)$
$\langle 3 \rangle 1.$ SUFFICES ASSUME NEW $i \in Nat,\ P(i)$ PROVE $P(i+1)$ OBVIOUS
$\langle 3 \rangle 2.$ Trivial facts to help the prover match known facts or proof obligations.
$\wedge\ i+1 \in Nat$
$\wedge\ i+1 \in Nat \setminus \{0\}$
$\wedge\ i+1 \neq 0$
$\wedge\ (i+1)-1 = i$
BY $SMTT(10)$
$\langle 3 \rangle 3.\ fb[i+1] = DeltaVecAdd(fb[i],\ Elemb(i+1))$ BY $\langle 3 \rangle 2,\ \langle 1 \rangle 10$
$\langle 3 \rangle 4.\ fb[i] \in DeltaVecType$ BY $\langle 3 \rangle 1$
$\langle 3 \rangle 5.\ Elemb(i+1) \in DeltaVecType$ BY $\langle 3 \rangle 2,\ \langle 1 \rangle 15$
$\langle 3 \rangle 6.\ fb[i+1] \in DeltaVecType$ BY $\langle 3 \rangle 3,\ \langle 3 \rangle 4,\ \langle 3 \rangle 5,\ DeltaVecAddType$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 6$
$\langle 2 \rangle$ HIDE DEF P
$\langle 2 \rangle$ QED BY ONLY $\langle 2 \rangle 1,\ \langle 2 \rangle 2,\ NatInduction,\ Isa$

Each sum evaluates its recursive function at the length of its sequence.
$\langle 1 \rangle$ DEFINE $AddD(v) \triangleq DeltaVecAdd(v,\ delta)$
$\langle 1 \rangle 19.\ fa[LenQa] = AddD(fb[LenQb])$
$\langle 2 \rangle$ DEFINE $P(i) \triangleq fa[i] =$ IF $i < k+1$ THEN $fb[i]$ ELSE $AddD(fb[i])$
$\langle 2 \rangle$ HIDE DEF $LenQa,\ LenQb,\ fa,\ fb,\ Elema,\ Elemb,\ AddD,\ delta$
$\langle 2 \rangle$ SUFFICES $\forall\, i \in Nat : P(i)$
$\langle 3 \rangle\ LenQa \in Nat$ BY $\langle 1 \rangle 3$
$\langle 3 \rangle\ LenQb \in Nat$ BY $\langle 1 \rangle 8$
$\langle 3 \rangle\ LenQa = LenQb$ BY $\langle 1 \rangle 11$
$\langle 3 \rangle\ k+1 \leq LenQa$ BY $\langle 1 \rangle 12,\ SMTT(10)$
$\langle 3 \rangle$ QED BY $SMTT(10)$
$\langle 2 \rangle 1.\ P(0)$
$\langle 3 \rangle 1.\ 0 < k+1$ BY $SMTT(10)$
$\langle 3 \rangle 2.\ fb[0] = DeltaVecZero$ BY $\langle 1 \rangle 10$ DEF $fb,\ f0b$
$\langle 3 \rangle 3.\ fa[0] = DeltaVecZero$ BY $\langle 1 \rangle 5$ DEF $fa,\ f0a$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 3 \rangle 2,\ \langle 3 \rangle 3$
$\langle 2 \rangle 2.\ \forall\, i \in Nat : P(i) \Rightarrow P(i+1)$
$\langle 3 \rangle 1.$ SUFFICES ASSUME NEW $i \in Nat,\ P(i)$ PROVE $P(i+1)$ OBVIOUS
$\langle 3 \rangle 2.$ Trivial facts to help the prover match known facts or proof obligations.
$\wedge\ i+1 \in Nat$
$\wedge\ i+1 \in Nat \setminus \{0\}$
$\wedge\ i+1 \neq 0$
$\wedge\ (i+1)-1 = i$
BY $SMTT(10)$
$\langle 3 \rangle$ DEFINE $fai \triangleq fa[i]$
$\langle 3 \rangle$ DEFINE $fbi \triangleq fb[i]$
$\langle 3 \rangle$ DEFINE $fai1 \triangleq fa[i+1]$
$\langle 3 \rangle$ DEFINE $fbi1 \triangleq fb[i+1]$
$\langle 3 \rangle$ DEFINE $vai1 \triangleq Elema(i+1)$

$\langle 3 \rangle$ DEFINE $vbi1 \triangleq Elemb(i+1)$
$\langle 3 \rangle 3.$ $fai1 = DeltaVecAdd(fai, vai1)$ BY $\langle 3 \rangle 2$, $\langle 1 \rangle 5$
$\langle 3 \rangle 4.$ $fbi1 = DeltaVecAdd(fbi, vbi1)$ BY $\langle 3 \rangle 2$, $\langle 1 \rangle 10$
$\langle 3 \rangle 5.$ $fbi \in DeltaVecType$ BY $\langle 3 \rangle 2$, $\langle 1 \rangle 18$
$\langle 3 \rangle 6.$ $vbi1 \in DeltaVecType$ BY $\langle 3 \rangle 2$, $\langle 1 \rangle 15$
$\langle 3 \rangle 7.$ CASE $i+1 \neq k+1$
  $\langle 4 \rangle 1.$ $vai1 = vbi1$ BY $\langle 3 \rangle 2$, $\langle 3 \rangle 7$, $\langle 1 \rangle 16$
  $\langle 4 \rangle 2.$ CASE $i < k+1$
    $\langle 5 \rangle 1.$ $fbi = fai$ BY $\langle 4 \rangle 2$, $\langle 3 \rangle 1$
    $\langle 5 \rangle 2.$ $i+1 < k+1$ BY $\langle 4 \rangle 2$, $\langle 3 \rangle 7$, $SMTT(10)$
    $\langle 5 \rangle$ SUFFICES $fai1 = fbi1$ BY $\langle 5 \rangle 2$
    $\langle 5 \rangle$ HIDE DEF fai, fbi, $fai1$, $fbi1$, $vai1$, $vbi1$
    $\langle 5 \rangle 3.$ $fbi1 = DeltaVecAdd(fai, vai1)$ BY $\langle 3 \rangle 4$, $\langle 5 \rangle 1$, $\langle 4 \rangle 1$
    $\langle 5 \rangle 4.$ $fbi1 = fai1$ BY $\langle 5 \rangle 3$, $\langle 3 \rangle 3$
    $\langle 5 \rangle$ QED BY $\langle 5 \rangle 4$
  $\langle 4 \rangle 3.$ CASE $\neg(i < k+1)$
    $\langle 5 \rangle 1.$ $fai = AddD(fbi)$ BY $\langle 4 \rangle 3$, $\langle 3 \rangle 1$
    $\langle 5 \rangle 2.$ $\neg(i+1 < k+1)$ BY $\langle 4 \rangle 3$, $\langle 3 \rangle 7$, $SMTT(10)$
    $\langle 5 \rangle$ SUFFICES $fai1 = AddD(fbi1)$ BY $\langle 5 \rangle 2$
    $\langle 5 \rangle$ HIDE DEF fai, fbi, $fai1$, $fbi1$, $vai1$, $vbi1$
    $\langle 5 \rangle$ $fbi \in DeltaVecType$ BY $\langle 3 \rangle 5$
    $\langle 5 \rangle$ $vbi1 \in DeltaVecType$ BY $\langle 3 \rangle 6$
    $\langle 5 \rangle$ $delta \in DeltaVecType$ BY $\langle 1 \rangle 13$
    $\langle 5 \rangle 3.$ $fai1 = DeltaVecAdd(DeltaVecAdd(fbi, delta), vbi1)$ BY $\langle 3 \rangle 3$, $\langle 5 \rangle 1$, $\langle 4 \rangle 1$ DEF $AddD$
    $\langle 5 \rangle 4.$ $fai1 = DeltaVecAdd(fbi, DeltaVecAdd(delta, vbi1))$ BY $\langle 5 \rangle 3$, $DeltaVecAddAssociative$
    $\langle 5 \rangle 5.$ $fai1 = DeltaVecAdd(fbi, DeltaVecAdd(vbi1, delta))$ BY $\langle 5 \rangle 4$, $DeltaVecAddCommutative$
    $\langle 5 \rangle 6.$ $fai1 = DeltaVecAdd(DeltaVecAdd(fbi, vbi1), delta)$ BY $\langle 5 \rangle 5$, $DeltaVecAddAssociative$
    $\langle 5 \rangle 7.$ $fai1 = AddD(fbi1)$ BY $\langle 5 \rangle 6$, $\langle 3 \rangle 4$ DEF $AddD$
    $\langle 5 \rangle$ QED BY $\langle 5 \rangle 7$
  $\langle 4 \rangle$ QED BY $\langle 4 \rangle 2$, $\langle 4 \rangle 3$
$\langle 3 \rangle 8.$ CASE $i+1 = k+1$
  $\langle 4 \rangle 1.$ $vai1 = AddD(vbi1)$ BY $\langle 3 \rangle 2$, $\langle 3 \rangle 8$, $\langle 1 \rangle 16$ DEF $AddD$
  $\langle 4 \rangle 2.$ $i < k+1$ BY $\langle 3 \rangle 8$, $SMTT(10)$
  $\langle 4 \rangle 3.$ $fai = fbi$ BY $\langle 4 \rangle 2$, $\langle 3 \rangle 1$
  $\langle 4 \rangle 4.$ $\neg(i+1 < k+1)$ BY $\langle 3 \rangle 8$, $SMTT(10)$
  $\langle 4 \rangle$ SUFFICES $fai1 = AddD(fbi1)$ BY $\langle 4 \rangle 4$
  $\langle 4 \rangle$ HIDE DEF fai, fbi, $fai1$, $fbi1$, $vai1$, $vbi1$
  $\langle 4 \rangle$ $fbi \in DeltaVecType$ BY $\langle 3 \rangle 5$
  $\langle 4 \rangle$ $vbi1 \in DeltaVecType$ BY $\langle 3 \rangle 6$
  $\langle 4 \rangle$ $delta \in DeltaVecType$ BY $\langle 1 \rangle 13$
  $\langle 4 \rangle 5.$ $fai1 = DeltaVecAdd(fbi, DeltaVecAdd(vbi1, delta))$ BY $\langle 3 \rangle 3$, $\langle 4 \rangle 3$, $\langle 4 \rangle 1$ DEF $AddD$
  $\langle 4 \rangle 6.$ $fai1 = DeltaVecAdd(DeltaVecAdd(fbi, vbi1), delta)$ BY $\langle 4 \rangle 5$, $DeltaVecAddAssociative$
  $\langle 4 \rangle 7.$ $fai1 = AddD(fbi1)$ BY $\langle 4 \rangle 6$, $\langle 3 \rangle 4$ DEF $AddD$
  $\langle 4 \rangle$ QED BY $\langle 4 \rangle 7$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 7$, $\langle 3 \rangle 8$
$\langle 2 \rangle$ HIDE DEF P

$\langle 2 \rangle$ QED BY ONLY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $NatInduction$, Isa
$\langle 1 \rangle$ HIDE DEF fa, fb
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 4$, $\langle 1 \rangle 9$, $\langle 1 \rangle 19$

When you append a value d to a sequence Q of delta vecs, the sums increase by d for all skip counts $k \leq Len(Q)$.

THEOREM $DeltaVecSeqSkipSumAppend \triangleq$
ASSUME
NEW $Q \in Seq(DeltaVecType)$,
NEW $d \in DeltaVecType$,
NEW $k \in Nat$, $k \leq Len(Q)$
PROVE
$DeltaVecSeqSkipSum(k, Append(Q, d)) = DeltaVecAdd(DeltaVecSeqSkipSum(k, Q), d)$
PROOF

$XXXa$ definitions are related to the sum based on the original Q.

$\langle 1 \rangle$ DEFINE $Qa \triangleq Q$
$\langle 1 \rangle$ DEFINE $Elema(i) \triangleq DeltaVecSeqSkipSum(k, Qa)! : !Elem(i)$
$\langle 1 \rangle$ DEFINE $f0a \triangleq DeltaVecZero$
$\langle 1 \rangle$ DEFINE $Defa(v, i) \triangleq DeltaVecAdd(v, Elema(i))$
$\langle 1 \rangle$ DEFINE $fa \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0a$ ELSE $Defa(f[i-1], i)]$
$\langle 1 \rangle$ DEFINE $LenQa \triangleq Len(Qa)$
$\langle 1 \rangle 1.$ $LenQa \in Nat$ BY $LenInNat$
$\langle 1 \rangle 2.$ $DeltaVecSeqSkipSum(k, Qa) = fa[LenQa]$ BY DEF $DeltaVecSeqSkipSum$

$\langle 1 \rangle 3.$ $\forall i \in Nat : fa[i] =$ IF $i = 0$ THEN $f0a$ ELSE $Defa(fa[i-1], i)$
$\langle 2 \rangle$ HIDE DEF $f0a$, $Defa$, fa
$\langle 2 \rangle$ SUFFICES $NatInductiveDefConclusion(fa, f0a, Defa)$ BY DEF $NatInductiveDefConclusion$
$\langle 2 \rangle$ SUFFICES $NatInductiveDefHypothesis(fa, f0a, Defa)$ BY $NatInductiveDef$
$\langle 2 \rangle$ QED BY DEF $NatInductiveDefHypothesis$, fa

$XXXb$ definitions are related to the sum after appending d to Q.

$\langle 1 \rangle$ DEFINE $Qb \triangleq Append(Q, d)$
$\langle 1 \rangle$ DEFINE $Elemb(i) \triangleq DeltaVecSeqSkipSum(k, Qb)! : !Elem(i)$
$\langle 1 \rangle$ DEFINE $f0b \triangleq DeltaVecZero$
$\langle 1 \rangle$ DEFINE $Defb(v, i) \triangleq DeltaVecAdd(v, Elemb(i))$
$\langle 1 \rangle$ DEFINE $fb \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0b$ ELSE $Defb(f[i-1], i)]$
$\langle 1 \rangle$ DEFINE $LenQb \triangleq Len(Qb)$
$\langle 1 \rangle 4.$ $LenQb \in Nat$ BY $LenInNat$, $IsaT(120)$
$\langle 1 \rangle 5.$ $DeltaVecSeqSkipSum(k, Qb) = fb[LenQb]$ BY DEF $DeltaVecSeqSkipSum$
$\langle 1 \rangle 6.$ $\forall i \in Nat : fb[i] =$ IF $i = 0$ THEN $f0b$ ELSE $Defb(fb[i-1], i)$
$\langle 2 \rangle$ HIDE DEF $f0b$, $Defb$, fb

$\langle 2 \rangle$ SUFFICES $NatInductiveDefConclusion(fb, f0b, Defb)$ BY DEF $NatInductiveDefConclusion$
$\langle 2 \rangle$ SUFFICES $NatInductiveDefHypothesis(fb, f0b, Defb)$ BY $NatInductiveDef$
$\langle 2 \rangle$ QED BY DEF $NatInductiveDefHypothesis, fb$

Now relate the two sums. We show that Qb is one element longer than Qa , that the extra element on the end of Qb is d , and that $Elemb$ and $Elema$ are identical for the length of Qa .

$\langle 1 \rangle 7.$ $LenQb = LenQa + 1$ BY $AppendProperties$
$\langle 1 \rangle 8.$ $\forall i \in 1 \,..\, LenQa : Qb[i] = Qa[i]$ BY $AppendProperties, IsaT(120)$
$\langle 1 \rangle 9.$ $Qb[LenQa + 1] = d$ BY $AppendProperties, IsaT(120)$
$\langle 1 \rangle 10.$ $\forall i \in 1 \,..\, LenQa : Elemb(i) = Elema(i)$
$\langle 2 \rangle$ HIDE DEF Qa, Qb
$\langle 2 \rangle$ SUFFICES ASSUME NEW $i \in 1 \,..\, LenQa$ PROVE $Elemb(i) = Elema(i)$ OBVIOUS
$\langle 2 \rangle 1.$ $Qb[i] = Qa[i]$ BY $\langle 1 \rangle 8$
$\langle 2 \rangle 2.$ $i \leq LenQa \wedge i \leq LenQb$
$\langle 3 \rangle$ HIDE DEF $LenQa, LenQb$
$\langle 3 \rangle$ QED BY $\langle 1 \rangle 1, \langle 1 \rangle 4, \langle 1 \rangle 7, SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1, \langle 2 \rangle 2$

The sum of the sequence evaluates its recursive function at the length of the sequence. Since Qb is one element longer than Qa , we use the recursive definition of fb to express $fb[LenQb]$ in terms of $fb[LenQa]$.

$\langle 1 \rangle 11.$ $fb[LenQa + 1] = DeltaVecAdd(fb[LenQa], Elemb(LenQa + 1))$
$\langle 2 \rangle$ HIDE DEF $LenQa, fb, Defb$
$\langle 2 \rangle 1.$ $LenQa + 1 \in Nat$ BY $\langle 1 \rangle 1, SMTT(10)$
$\langle 2 \rangle 2.$ $LenQa + 1 \neq 0$ BY $\langle 1 \rangle 1, SMTT(10)$
$\langle 2 \rangle 3.$ $(LenQa + 1) - 1 = LenQa$ BY $\langle 1 \rangle 1, SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 6, \langle 2 \rangle 1, \langle 2 \rangle 2, \langle 2 \rangle 3$ DEF $Defb$

Now we show that evaluating fb at the length of Qa is the same as evaluating fa at the length of Qa . Proving this requires induction.

$\langle 1 \rangle 12.$ $fb[LenQa] = fa[LenQa]$
$\langle 2 \rangle$ DEFINE $P(i) \triangleq i \leq LenQa \Rightarrow fb[i] = fa[i]$
$\langle 2 \rangle$ HIDE DEF $LenQa, fa, fb$
$\langle 2 \rangle$ SUFFICES $\forall i \in Nat : P(i)$ BY $\langle 1 \rangle 1, SMTT(10)$
$\langle 2 \rangle 1.$ $P(0)$ BY $\langle 1 \rangle 3, \langle 1 \rangle 6$ DEF $fa, f0a, fb, f0b$
$\langle 2 \rangle 2.$ ASSUME NEW $i \in Nat, P(i)$ PROVE $P(i + 1)$
$\langle 3 \rangle 1.$ Trivial facts to help the prover match known facts or proof obligations.
$\wedge\ i + 1 \in Nat$
$\wedge\ i + 1 \neq 0$
$\wedge\ (i + 1) - 1 = i$
BY $SMTT(10)$
$\langle 3 \rangle 2.$ SUFFICES ASSUME $i + 1 \leq LenQa$ PROVE $fb[i + 1] = fa[i + 1]$ OBVIOUS
$\langle 3 \rangle 3.$ $fa[i + 1] = DeltaVecAdd(fa[i], Elema(i + 1))$ BY $\langle 3 \rangle 1, \langle 1 \rangle 3$
$\langle 3 \rangle 4.$ $fb[i + 1] = DeltaVecAdd(fb[i], Elemb(i + 1))$ BY $\langle 3 \rangle 1, \langle 1 \rangle 6$
$\langle 3 \rangle 5.$ $fb[i] = fa[i]$
$\langle 4 \rangle 1.$ $i \leq LenQa$ BY $\langle 3 \rangle 2, \langle 1 \rangle 1, SMTT(10)$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1, \langle 2 \rangle 2$
$\langle 3 \rangle 6.$ $Elemb(i + 1) = Elema(i + 1)$
$\langle 4 \rangle 1.$ $i + 1 \in 1 \,..\, LenQa$ BY $\langle 3 \rangle 2, \langle 1 \rangle 1, DotDotDef, SMTT(10)$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1, \langle 1 \rangle 10$

$\langle 3 \rangle$ QED BY $\langle 3 \rangle 3,\ \langle 3 \rangle 4,\ \langle 3 \rangle 5,\ \langle 3 \rangle 6$
$\langle 2 \rangle$ HIDE DEF P
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 2,\ NatInduction,\ Isa$

Now we show that $Elemb(LenQb)$ is in fact d , the additional element that was appended to Qa . Proving this requires that $k \leq Len(Q)$.

$\langle 1 \rangle 13.$ $Elemb(LenQa + 1) = d$
$\langle 2 \rangle$ SUFFICES $k < LenQa + 1 \land LenQa + 1 \leq LenQb$ BY $\langle 1 \rangle 9$
$\langle 2 \rangle 1.$ $k \leq LenQa$ OBVIOUS
$\langle 2 \rangle$ HIDE DEF $LenQa,\ LenQb$
$\langle 2 \rangle 2.$ $k < LenQa\ \ + 1$ BY $\langle 1 \rangle 1,\ \langle 2 \rangle 1,\ SMTT(10)$
$\langle 2 \rangle 3.$ $LenQa + 1 \leq LenQb$ BY $\langle 1 \rangle 1,\ \langle 1 \rangle 7,\ SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2,\ \langle 2 \rangle 3$
$\langle 1 \rangle$ HIDE DEF $fa,\ fb$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2,\ \langle 1 \rangle 5,\ \langle 1 \rangle 11,\ \langle 1 \rangle 12,\ \langle 1 \rangle 13,\ IsaT(120)$

For a non-empty sequence Q of delta vecs, the sums skipping k of $Tail(Q)$ are the same as the sums skipping $k + 1$ of Q.

THEOREM $DeltaVecSeqSkipSumTail \triangleq$
ASSUME
NEW $Q \in Seq(DeltaVecType),\ Q \neq \langle \rangle,$
NEW $k\ \in Nat$
PROVE
$DeltaVecSeqSkipSum(k,\ Tail(Q)) = DeltaVecSeqSkipSum(k + 1,\ Q)$
PROOF

$XXXa$ definitions are related to the sum of Q skipping $k + 1$.

$\langle 1 \rangle$ DEFINE $Qa \triangleq Q$
$\langle 1 \rangle$ DEFINE $Elema(i) \triangleq DeltaVecSeqSkipSum(k + 1,\ Qa)! : !Elem(i)$
$\langle 1 \rangle$ DEFINE $f0a \triangleq DeltaVecZero$
$\langle 1 \rangle$ DEFINE $Defa(v,\ i) \triangleq DeltaVecAdd(v,\ Elema(i))$
$\langle 1 \rangle$ DEFINE $fa \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0a$ ELSE $Defa(f[i - 1],\ i)]$
$\langle 1 \rangle$ DEFINE $LenQa \triangleq Len(Qa)$
$\langle 1 \rangle 1.$ $k + 1 \in Nat$ BY $SMTT(10)$
$\langle 1 \rangle 2.$ $LenQa \in Nat$ BY $LenInNat$
$\langle 1 \rangle 3.$ $DeltaVecSeqSkipSum(k + 1,\ Qa) = fa[LenQa]$ BY $\langle 1 \rangle 1$ DEF $DeltaVecSeqSkipSum$

$\langle 1 \rangle 4.$ $\forall\, i \in Nat : fa[i] =$ IF $i = 0$ THEN $f0a$ ELSE $Defa(fa[i - 1],\ i)$
$\langle 2 \rangle$ HIDE DEF $f0a,\ Defa,\ fa$
$\langle 2 \rangle$ SUFFICES $NatInductiveDefConclusion(fa,\ f0a,\ Defa)$ BY DEF $NatInductiveDefConclusion$
$\langle 2 \rangle$ SUFFICES $NatInductiveDefHypothesis(fa,\ f0a,\ Defa)$ BY $NatInductiveDef$
$\langle 2 \rangle$ QED BY DEF $NatInductiveDefHypothesis,\ fa$

$XXXb$ definitions are related to the sum of $Tail(Q)$ skipping k.

$\langle 1\rangle$ DEFINE $Qb \triangleq Tail(Q)$
$\langle 1\rangle$ DEFINE $Elemb(i) \triangleq DeltaVecSeqSkipSum(k, Qb)!:!Elem(i)$
$\langle 1\rangle$ DEFINE $f0b \triangleq DeltaVecZero$
$\langle 1\rangle$ DEFINE $Defb(v, i) \triangleq DeltaVecAdd(v, Elemb(i))$
$\langle 1\rangle$ DEFINE $fb \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0b$ ELSE $Defb(f[i-1], i)]$
$\langle 1\rangle$ DEFINE $LenQb \triangleq Len(Qb)$
$\langle 1\rangle$5. $Qb \in Seq(DeltaVecType)$ BY $TailProp$
$\langle 1\rangle$6. $LenQb \in Nat$ BY $\langle 1\rangle 5$, $LenInNat$
$\langle 1\rangle$7. $DeltaVecSeqSkipSum(k, Qb) = fb[LenQb]$ BY DEF $DeltaVecSeqSkipSum$

$\langle 1\rangle$8. $\forall i \in Nat : fb[i] =$ IF $i = 0$ THEN $f0b$ ELSE $Defb(fb[i-1], i)$
  $\langle 2\rangle$ HIDE DEF $f0b$, $Defb$, fb
  $\langle 2\rangle$ SUFFICES $NatInductiveDefConclusion(fb, f0b, Defb)$ BY DEF $NatInductiveDefConclusion$
  $\langle 2\rangle$ SUFFICES $NatInductiveDefHypothesis(fb, f0b, Defb)$ BY $NatInductiveDef$
  $\langle 2\rangle$ QED BY DEF $NatInductiveDefHypothesis$, fb

Now relate the two sums. We show that $Elemb(i)$ is the same as $Elema(i+1)$ and that $Elema(1)$ is zero.

$\langle 1\rangle$9. $LenQb = LenQa - 1$ BY $TailProp$
$\langle 1\rangle$10. $\forall i \in Nat \setminus \{0\} : Elemb(i) = Elema(i+1)$
  $\langle 2\rangle$1. $\forall i \in 1 \,..\, LenQb : Qb[i] = Qa[i+1]$ BY $TailProp$
  $\langle 2\rangle$ HIDE DEF Qa, Qb, $LenQa$, $LenQb$
  $\langle 2\rangle$ SUFFICES ASSUME NEW $i \in Nat \setminus \{0\}$ PROVE $Elemb(i) = Elema(i+1)$ OBVIOUS
  $\langle 2\rangle$2. $(i \in 1 \,..\, LenQb) \lor (i > LenQb)$ BY $\langle 1\rangle 6$, $DotDotDef$, $SMTT(10)$
  $\langle 2\rangle$3. CASE $i \in 1 \,..\, LenQb$
    $\langle 3\rangle$1. $Qb[i] = Qa[i+1]$ BY $\langle 2\rangle 1$, $\langle 2\rangle 3$
    $\langle 3\rangle$2. $i \leq LenQb \land i + 1 \leq LenQa$ BY $\langle 2\rangle 3$, $\langle 1\rangle 2$, $\langle 1\rangle 6$, $\langle 1\rangle 9$, $DotDotDef$, $SMTT(10)$
    $\langle 3\rangle$3. $k < i \equiv k + 1 < i + 1$ BY $SMTT(10)$
    $\langle 3\rangle$ QED BY $\langle 3\rangle 1$, $\langle 3\rangle 2$, $\langle 3\rangle 3$ DEF $LenQa$, $LenQb$
  $\langle 2\rangle$4. CASE $i > LenQb$
    $\langle 3\rangle$1. $\neg(i \leq LenQb) \land \neg(i + 1 \leq LenQa)$ BY $\langle 2\rangle 4$, $\langle 1\rangle 2$, $\langle 1\rangle 6$, $\langle 1\rangle 9$, $SMTT(10)$
    $\langle 3\rangle$ QED BY $\langle 3\rangle 1$ DEF $LenQa$, $LenQb$
  $\langle 2\rangle$ QED BY $\langle 2\rangle 2$, $\langle 2\rangle 3$, $\langle 2\rangle 4$
$\langle 1\rangle$11. $Elema(1) = DeltaVecZero$
  $\langle 2\rangle$1. $\neg(k + 1 < 1)$ BY $SMTT(10)$
  $\langle 2\rangle$ QED BY $\langle 2\rangle 1$

Each sum evaluates its recursive function at the length of its sequence. Now we show that the results are the same for each sum. Proving this requires induction.

$\langle 1\rangle$12. $fb[LenQb] = fa[LenQa]$
  $\langle 2\rangle$ DEFINE $P(i) \triangleq fb[i] = fa[i+1]$
  $\langle 2\rangle$ HIDE DEF $LenQa$, $LenQb$, fa, fb
  $\langle 2\rangle$ SUFFICES $\forall i \in Nat : P(i)$ BY $\langle 1\rangle 2$, $\langle 1\rangle 6$, $\langle 1\rangle 9$, $SMTT(10)$
  $\langle 2\rangle$1. $P(0)$
    $\langle 3\rangle$1. Trivial facts to help the prover match known facts or proof obligations.
      $\land\ 0 + 1 \in Nat$
      $\land\ 0 + 1 \neq 0$
      $\land\ (0 + 1) - 1 = 0$

$\land\ 0 + 1 = 1$
BY $SMTT(10)$
$\langle 3\rangle 2.\ fb[0] = DeltaVecZero$ BY $\langle 1\rangle 8$ DEF $fb,\ f0b$
$\langle 3\rangle 3.\ fa[1] = DeltaVecZero$
$\langle 4\rangle 2.\ fa[1] = DeltaVecAdd(fa[0],\ Elema(1))$ BY $\langle 3\rangle 1,\ \langle 1\rangle 4$ DEF $fa,\ Defa$
$\langle 4\rangle 3.\ fa[0] = DeltaVecZero$ BY $\langle 1\rangle 4$ DEF $fa,\ f0a$
$\langle 4\rangle 4.\ Elema(1) = DeltaVecZero$ BY $\langle 3\rangle 1,\ \langle 1\rangle 11$
$\langle 4\rangle$ QED BY $\langle 4\rangle 2,\ \langle 4\rangle 3,\ \langle 4\rangle 4,\ DeltaVecZeroType,\ DeltaVecAddZero$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ \langle 3\rangle 2,\ \langle 3\rangle 3$
$\langle 2\rangle 2.$ ASSUME NEW $i \in Nat,\ P(i)$ PROVE $P(i+1)$
$\langle 3\rangle 1.$ Trivial facts to help the prover match known facts or proof obligations.
$\land\ i + 1 \in Nat$
$\land\ i + 1 \in Nat \setminus \{0\}$
$\land\ i + 1 \neq 0$
$\land\ (i + 1) - 1 = i$
$\land\ (i + 1) + 1 = i + 2$
$\land\ i + 2 \in Nat$
$\land\ i + 2 \in Nat \setminus \{0\}$
$\land\ i + 2 \neq 0$
$\land\ (i + 2) - 1 = i + 1$
BY $SMTT(10)$
$\langle 3\rangle 2.\ fb[i+1] = DeltaVecAdd(fb[i],\ Elemb(i+1))$ BY $\langle 3\rangle 1,\ \langle 1\rangle 8$
$\langle 3\rangle 3.\ fa[i+2] = DeltaVecAdd(fa[i+1],\ Elema(i+2))$ BY $\langle 3\rangle 1,\ \langle 1\rangle 4$
$\langle 3\rangle 4.\ fb[i] = fa[i+1]$ BY $\langle 2\rangle 2$
$\langle 3\rangle 5.\ Elemb(i+1) = Elema(i+2)$ BY $\langle 1\rangle 10,\ \langle 3\rangle 1$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ \langle 3\rangle 2,\ \langle 3\rangle 3,\ \langle 3\rangle 4,\ \langle 3\rangle 5$
$\langle 2\rangle$ HIDE DEF P
$\langle 2\rangle$ QED BY $\langle 2\rangle 1,\ \langle 2\rangle 2,\ NatInduction,\ Isa$
$\langle 1\rangle$ HIDE DEF $fa,\ fb$
$\langle 1\rangle$ QED BY $\langle 1\rangle 3,\ \langle 1\rangle 7,\ \langle 1\rangle 12$

For a non-empty sequence Q of delta vectors, $Head(Q)$ plus the sum of all delta vectors on $Tail(Q)$ is the same as the sum of all delta vecs on Q.

THEOREM $DeltaVecSeqSkipSumHeadTail \triangleq$
ASSUME
NEW $Q \in Seq(DeltaVecType),\ Q \neq \langle\rangle$
PROVE
$DeltaVecAdd(Head(Q),\ DeltaVecSeqSkipSum(0,\ Tail(Q))) = DeltaVecSeqSkipSum(0,\ Q)$
PROOF
$XXXa$ definitions are related to the sum of Q skipping 0.

$\langle 1\rangle$ DEFINE $Qa \triangleq Q$
$\langle 1\rangle$ DEFINE $Elema(i) \triangleq DeltaVecSeqSkipSum(0, Qa)!:!Elem(i)$
$\langle 1\rangle$ DEFINE $f0a \triangleq DeltaVecZero$
$\langle 1\rangle$ DEFINE $Defa(v, i) \triangleq DeltaVecAdd(v, Elema(i))$
$\langle 1\rangle$ DEFINE $fa \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0a$ ELSE $Defa(f[i-1], i)]$
$\langle 1\rangle$ DEFINE $LenQa \triangleq Len(Qa)$
$\langle 1\rangle 1.$ $LenQa \in Nat$ BY $LenInNat$
$\langle 1\rangle 2.$ $DeltaVecSeqSkipSum(0, Qa) = fa[LenQa]$ BY DEF $DeltaVecSeqSkipSum$

$\langle 1\rangle 3.$ $\forall i \in Nat : fa[i] =$ IF $i = 0$ THEN $f0a$ ELSE $Defa(fa[i-1], i)$
  $\langle 2\rangle$ HIDE DEF $f0a, Defa, fa$
  $\langle 2\rangle$ SUFFICES $NatInductiveDefConclusion(fa, f0a, Defa)$ BY DEF $NatInductiveDefConclusion$
  $\langle 2\rangle$ SUFFICES $NatInductiveDefHypothesis(fa, f0a, Defa)$ BY $NatInductiveDef$
  $\langle 2\rangle$ QED BY DEF $NatInductiveDefHypothesis, fa$

$XXXb$ definitions are related to the sum of $Tail(Q)$ skipping 0.

$\langle 1\rangle$ DEFINE $Qb \triangleq Tail(Q)$
$\langle 1\rangle$ DEFINE $Elemb(i) \triangleq DeltaVecSeqSkipSum(0, Qb)!:!Elem(i)$
$\langle 1\rangle$ DEFINE $f0b \triangleq DeltaVecZero$
$\langle 1\rangle$ DEFINE $Defb(v, i) \triangleq DeltaVecAdd(v, Elemb(i))$
$\langle 1\rangle$ DEFINE $fb \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0b$ ELSE $Defb(f[i-1], i)]$
$\langle 1\rangle$ DEFINE $LenQb \triangleq Len(Qb)$
$\langle 1\rangle 4.$ $Qb \in Seq(DeltaVecType)$ BY $TailProp$
$\langle 1\rangle 5.$ $LenQb \in Nat$ BY $\langle 1\rangle 4$, $LenInNat$
$\langle 1\rangle 6.$ $DeltaVecSeqSkipSum(0, Qb) = fb[LenQb]$ BY DEF $DeltaVecSeqSkipSum$

$\langle 1\rangle 7.$ $\forall i \in Nat : fb[i] =$ IF $i = 0$ THEN $f0b$ ELSE $Defb(fb[i-1], i)$
  $\langle 2\rangle$ HIDE DEF $f0b, Defb, fb$
  $\langle 2\rangle$ SUFFICES $NatInductiveDefConclusion(fb, f0b, Defb)$ BY DEF $NatInductiveDefConclusion$
  $\langle 2\rangle$ SUFFICES $NatInductiveDefHypothesis(fb, f0b, Defb)$ BY $NatInductiveDef$
  $\langle 2\rangle$ QED BY DEF $NatInductiveDefHypothesis, fb$

Each $Elemb$ is a delta vec.

$\langle 1\rangle 8.$ $\forall i \in Nat \setminus \{0\} : Elemb(i) \in DeltaVecType$
  $\langle 2\rangle$ SUFFICES ASSUME NEW $i \in Nat \setminus \{0\}$ PROVE $Elemb(i) \in DeltaVecType$ OBVIOUS
  $\langle 2\rangle$ HIDE DEF $LenQb$
  $\langle 2\rangle 1.$ CASE $0 < i \land i \leq LenQb$
    $\langle 3\rangle 1.$ $i \in 1 \,..\, LenQb$ BY $\langle 2\rangle 1$, $\langle 1\rangle 5$, $DotDotDef$, $SMTT(10)$
    $\langle 3\rangle 2.$ $Qb[i] \in DeltaVecType$ BY $\langle 3\rangle 1$, $\langle 1\rangle 4$, $LenAxiom$ DEF $LenQb$
    $\langle 3\rangle$ QED BY $\langle 3\rangle 2$, $\langle 2\rangle 1$ DEF $LenQb$
  $\langle 2\rangle 2.$ CASE $\neg(0 < i \land i \leq LenQb)$
    $\langle 3\rangle 1.$ $Elemb(i) = DeltaVecZero$ BY $\langle 2\rangle 2$ DEF $LenQb$
    $\langle 3\rangle$ QED BY $\langle 3\rangle 1$, $DeltaVecZeroType$
  $\langle 2\rangle$ QED BY $\langle 2\rangle 1$, $\langle 2\rangle 2$

Now relate the two sums. We show that $Elemb(i)$ is the same as $Elema(i+1)$.

$\langle 1 \rangle 9.$ $LenQb = LenQa - 1$ BY $TailProp$
$\langle 1 \rangle 10.$ $\forall\, i \in Nat \setminus \{0\} : Elemb(i) = Elema(i+1)$
  $\langle 2 \rangle 1.$ $\forall\, i \in 1 \,..\, LenQb : Qb[i] = Qa[i+1]$ BY $TailProp$
  $\langle 2 \rangle 2.$ $\forall\, i \in 1 \,..\, LenQb : Qb[i] \in DeltaVecType$ BY $TailProp$, $ElementOfSeq$
  $\langle 2 \rangle$ HIDE DEF Qa, Qb, $LenQa$, $LenQb$
  $\langle 2 \rangle 3.$ SUFFICES ASSUME NEW $i \in Nat \setminus \{0\}$
      PROVE $Elemb(i) = Elema(i+1)$
      OBVIOUS
  $\langle 2 \rangle 4.$ $(i \in 1 \,..\, LenQb) \lor (i > LenQb)$ BY $\langle 1 \rangle 5$, $DotDotDef$, $SMTT(10)$
  $\langle 2 \rangle 5.$ CASE $i \in 1 \,..\, LenQb$
    $\langle 3 \rangle 1.$ $Qb[i] = Qa[i+1]$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 5$
    $\langle 3 \rangle 2.$ $i \leq LenQb \land i+1 \leq LenQa$ BY $\langle 2 \rangle 5$, $\langle 1 \rangle 1$, $\langle 1 \rangle 5$, $\langle 1 \rangle 9$, $DotDotDef$, $SMTT(10)$
    $\langle 3 \rangle 3.$ $0 < i \land 0 < i+1$ BY $SMTT(10)$
    $\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 3 \rangle 3$ DEF $LenQa$, $LenQb$
  $\langle 2 \rangle 6.$ CASE $i > LenQb$
    $\langle 3 \rangle 1.$ $\neg(i \leq LenQb) \land \neg(i+1 \leq LenQa)$ BY $\langle 2 \rangle 6$, $\langle 1 \rangle 1$, $\langle 1 \rangle 5$, $\langle 1 \rangle 9$, $SMTT(10)$
    $\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$ DEF $LenQa$, $LenQb$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 4$, $\langle 2 \rangle 5$, $\langle 2 \rangle 6$

We show that $Elema(1)$ is $Head(Q)$.

$\langle 1 \rangle 11.$ $Head(Q) \in DeltaVecType$ BY $HeadType$
$\langle 1 \rangle 12.$ $LenQa > 0$
  $\langle 2 \rangle 2.$ $LenQa \neq 0$ BY $EmptySeq$
  $\langle 2 \rangle$ HIDE DEF $LenQa$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 1 \rangle 1$, $SMTT(10)$
$\langle 1 \rangle 13.$ $Elema(1) = Head(Q)$
  $\langle 2 \rangle$ HIDE DEF $LenQa$
  $\langle 2 \rangle 1.$ $Qa[1] = Head(Q)$ BY $HeadDef$
  $\langle 2 \rangle 3.$ $0 < 1 \land 1 \leq LenQa$ BY $\langle 1 \rangle 12$, $\langle 1 \rangle 1$, $SMTT(10)$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 3$ DEF $LenQa$

Each sum evaluates its recursive function at the length of its sequence. Now we show that adding $Head(Q)$ to the sum of $Tail(Q)$ is the same as the sum of Q. Proving this requires induction.

$\langle 1 \rangle$ DEFINE $AddHeadQ(v) \triangleq DeltaVecAdd(Head(Q),\, v)$
$\langle 1 \rangle 14.$ $AddHeadQ(fb[LenQb]) = fa[LenQa]$
  $\langle 2 \rangle$ DEFINE $P(i) \triangleq AddHeadQ(fb[i]) = fa[i+1] \land fb[i] \in DeltaVecType$
  $\langle 2 \rangle$ HIDE DEF $LenQa$, $LenQb$, fa, fb, $AddHeadQ$
  $\langle 2 \rangle$ SUFFICES $\forall\, i \in Nat : P(i)$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 5$, $\langle 1 \rangle 9$, $\langle 1 \rangle 12$, $SMTT(10)$
  $\langle 2 \rangle 1.$ $P(0)$
    $\langle 3 \rangle 1.$ Trivial facts to help the prover match known facts or proof obligations.
      $\land\ 0 + 1 \in Nat$
      $\land\ 0 + 1 \neq 0$
      $\land\ (0 + 1) - 1 = 0$
      $\land\ 0 + 1 = 1$
      BY $SMTT(10)$
    $\langle 3 \rangle 2.$ $fb[0] = DeltaVecZero$ BY $\langle 1 \rangle 7$ DEF fb, $f0b$
    $\langle 3 \rangle 3.$ $fb[0] \in DeltaVecType$ BY $\langle 3 \rangle 2$, $DeltaVecZeroType$

$\langle 3\rangle 4.$ $DeltaVecAdd(Head(Q), fb[0]) = Head(Q)$ BY $\langle 3\rangle 2$, $\langle 1\rangle 11$, $DeltaVecAddZero$
$\langle 3\rangle 5.$ $AddHeadQ(fb[0]) = Head(Q)$ BY $\langle 3\rangle 4$ DEF $AddHeadQ$
$\langle 3\rangle 6.$ $fa[0] = DeltaVecZero$ BY $\langle 1\rangle 3$ DEF fa, $f0a$
$\langle 3\rangle 7.$ $fa[1] = DeltaVecAdd(fa[0], Elema(1))$ BY $\langle 3\rangle 1$, $\langle 1\rangle 3$ DEF fa, $Defa$
$\langle 3\rangle 8.$ $fa[1] = Head(Q)$ BY $\langle 3\rangle 7$, $\langle 3\rangle 6$, $\langle 1\rangle 13$, $\langle 1\rangle 11$, $DeltaVecAddZero$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1$, $\langle 3\rangle 3$, $\langle 3\rangle 5$, $\langle 3\rangle 8$
$\langle 2\rangle 2.$ ASSUME NEW $i \in Nat$, $P(i)$ PROVE $P(i+1)$
$\langle 3\rangle 1.$ Trivial facts to help the prover match known facts or proof obligations.

$\wedge\ i+1 \in Nat$
$\wedge\ i+1 \in Nat \setminus \{0\}$
$\wedge\ i+1 \neq 0$
$\wedge\ (i+1)-1 = i$
$\wedge\ (i+1)+1 = i+2$
$\wedge\ i+2 \in Nat$
$\wedge\ i+2 \in Nat \setminus \{0\}$
$\wedge\ i+2 \neq 0$
$\wedge\ (i+2)-1 = i+1$
BY $SMTT(10)$

Write in terms of definitions that can be hidden.

$\langle 3\rangle$ DEFINE $hq \triangleq Head(Q)$
$\langle 3\rangle$ DEFINE $fbi \triangleq fb[i]$
$\langle 3\rangle$ DEFINE $fbi1 \triangleq fb[i+1]$
$\langle 3\rangle$ DEFINE $vbi1 \triangleq Elemb(i+1)$
$\langle 3\rangle$ DEFINE $fai1 \triangleq fa[i+1]$
$\langle 3\rangle$ DEFINE $fai2 \triangleq fa[i+2]$
$\langle 3\rangle$ DEFINE $vai2 \triangleq Elema(i+2)$

Expose one level of recursion and re-associate adding $Head(Q)$.

$\langle 3\rangle 2.$ $fai2 = DeltaVecAdd(fai1, vai2)$ BY $\langle 3\rangle 1$, $\langle 1\rangle 3$
$\langle 3\rangle 3.$ $vbi1 = vai2$ BY $\langle 3\rangle 1$, $\langle 1\rangle 10$
$\langle 3\rangle 4.$ $fai2 = DeltaVecAdd(fai1, vbi1)$ BY $\langle 3\rangle 2$, $\langle 3\rangle 3$
$\langle 3\rangle 5.$ $AddHeadQ(fbi) = fai1$ BY $\langle 2\rangle 2$
$\langle 3\rangle 6.$ $DeltaVecAdd(hq, fbi) = fai1$ BY $\langle 3\rangle 5$ DEF $AddHeadQ$
$\langle 3\rangle 7.$ $fai2 = DeltaVecAdd(DeltaVecAdd(hq, fbi), vbi1)$ BY $\langle 3\rangle 4$, $\langle 3\rangle 6$
$\langle 3\rangle 8.$ $hq \in DeltaVecType$ BY $\langle 1\rangle 11$
$\langle 3\rangle 9.$ $fbi \in DeltaVecType$ BY $\langle 2\rangle 2$
$\langle 3\rangle 10.$ $vbi1 \in DeltaVecType$ BY $\langle 3\rangle 1$, $\langle 1\rangle 8$
$\langle 3\rangle 11.$ $fai2 = DeltaVecAdd(hq, DeltaVecAdd(fbi, vbi1))$
$\langle 4\rangle$ HIDE DEF $fai2$, hq, fbi, $vbi1$
$\langle 4\rangle$ QED BY $\langle 3\rangle 7$, $\langle 3\rangle 8$, $\langle 3\rangle 9$, $\langle 3\rangle 10$, $DeltaVecAddAssociative$
$\langle 3\rangle 12.$ $fbi1 = DeltaVecAdd(fbi, vbi1)$ BY $\langle 3\rangle 1$, $\langle 1\rangle 7$
$\langle 3\rangle 13.$ $fbi1 \in DeltaVecType$
$\langle 4\rangle$ HIDE DEF $fbi1$, fbi, $vbi1$
$\langle 4\rangle$ QED BY $\langle 3\rangle 9$, $\langle 3\rangle 10$, $\langle 3\rangle 12$, $DeltaVecAddType$
$\langle 3\rangle 14.$ $fai2 = DeltaVecAdd(hq, fbi1)$ BY $\langle 3\rangle 11$, $\langle 3\rangle 12$
$\langle 3\rangle 15.$ $fai2 = AddHeadQ(fbi1)$ BY $\langle 3\rangle 14$ DEF $AddHeadQ$

$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 13$, $\langle 3 \rangle 15$
$\langle 2 \rangle$ HIDE DEF P
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $NatInduction$, Isa
$\langle 1 \rangle$ HIDE DEF fa, fb
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2$, $\langle 1 \rangle 6$, $\langle 1 \rangle 14$

For a sequence Q of delta vectors and an index $n \in 1 \,..\, Len(Q)$, $Q[n]$ plus the sum of all delta vectors on $RemoveAt(Q, n)$ is the same as the sum of all delta vectors on Q.

THEOREM $DeltaVecSeqSkipSumRemoveAt \triangleq$
ASSUME
NEW $Q \in Seq(DeltaVecType)$,
NEW $n \in 1 \,..\, Len(Q)$
PROVE
$DeltaVecAdd(Q[n], DeltaVecSeqSkipSum(0, RemoveAt(Q, n))) = DeltaVecSeqSkipSum(0, Q)$
PROOF

$XXXa$ definitions are related to the sum of Q skipping 0.

$\langle 1 \rangle$ DEFINE $Qa \triangleq Q$
$\langle 1 \rangle$ DEFINE $Elema(i) \triangleq DeltaVecSeqSkipSum(0, Qa)!:!Elem(i)$
$\langle 1 \rangle$ DEFINE $f0a \triangleq DeltaVecZero$
$\langle 1 \rangle$ DEFINE $Defa(v, i) \triangleq DeltaVecAdd(v, Elema(i))$
$\langle 1 \rangle$ DEFINE $fa \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0a$ ELSE $Defa(f[i-1], i)]$
$\langle 1 \rangle$ DEFINE $LenQa \triangleq Len(Qa)$
$\langle 1 \rangle$ HIDE DEF Qa, $Elema$, $f0a$, $Defa$, fa, $LenQa$

$\langle 1 \rangle 1.$ $Qa \in Seq(DeltaVecType)$ BY DEF Qa
$\langle 1 \rangle 2.$ $Qa \in [1 \,..\, LenQa \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 1$, $LenAxiom$ DEF $LenQa$
$\langle 1 \rangle 3.$ $LenQa \in Nat$ BY $\langle 1 \rangle 1$, $LenInNat$ DEF $LenQa$
$\langle 1 \rangle 4.$ $DeltaVecSeqSkipSum(0, Qa) = fa[LenQa]$ BY DEF $DeltaVecSeqSkipSum$, fa, $f0a$, $Defa$, $Elema$, $LenQa$

$\langle 1 \rangle 5.$ $\forall i \in Nat : fa[i] =$ IF $i = 0$ THEN $f0a$ ELSE $Defa(fa[i-1], i)$
$\langle 2 \rangle$ SUFFICES $NatInductiveDefConclusion(fa, f0a, Defa)$ BY DEF $NatInductiveDefConclusion$
$\langle 2 \rangle$ SUFFICES $NatInductiveDefHypothesis(fa, f0a, Defa)$ BY $NatInductiveDef$
$\langle 2 \rangle$ QED BY DEF $NatInductiveDefHypothesis$, fa

$\langle 1 \rangle 6.$ $\forall i \in Nat \setminus \{0\} : Elema(i) \in DeltaVecType$
$\langle 2 \rangle$ SUFFICES ASSUME NEW $i \in Nat \setminus \{0\}$ PROVE $Elema(i) \in DeltaVecType$ OBVIOUS
$\langle 2 \rangle 1.$ CASE $0 < i \wedge i \leq LenQa$
$\langle 3 \rangle 1.$ $i \in 1 \,..\, LenQa$ BY $\langle 2 \rangle 1$, $\langle 1 \rangle 3$, $DotDotDef$, $SMTT(10)$
$\langle 3 \rangle 2.$ $Qa[i] \in DeltaVecType$ BY $\langle 3 \rangle 1$, $\langle 1 \rangle 2$

$\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$, $\langle 2 \rangle 1$ DEF $LenQa$, $Elema$
$\langle 2 \rangle 2$. CASE $\neg(0 < i \land i \leq LenQa)$
$\langle 3 \rangle 1$. $Elema(i) = DeltaVecZero$ BY $\langle 2 \rangle 2$ DEF $LenQa$, $Elema$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $DeltaVecZeroType$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$

$\langle 1 \rangle 7$. $\forall i \in Nat : fa[i] \in DeltaVecType$
$\langle 2 \rangle$ DEFINE $P(i) \triangleq fa[i] \in DeltaVecType$
$\langle 2 \rangle$ HIDE DEF P
$\langle 2 \rangle$ SUFFICES $\forall i \in Nat : P(i)$ BY DEF P
$\langle 2 \rangle 1$. $P(0)$
$\langle 3 \rangle 1$. $fa[0] = DeltaVecZero$ BY $\langle 1 \rangle 5$ DEF $f0a$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $DeltaVecZeroType$ DEF P
$\langle 2 \rangle 2$. ASSUME NEW $i \in Nat$, $P(i)$ PROVE $P(i+1)$

Rewrite $(i, i+1)$ to $(j-1, j)$ to match inductive def.

$\langle 3 \rangle 1$. PICK $j : j = i + 1$ OBVIOUS
$\langle 3 \rangle 2$. $j \in Nat \setminus \{0\}$ BY $\langle 3 \rangle 1$, $SMTT(10)$
$\langle 3 \rangle 3$. $j - 1 = i$ BY $\langle 3 \rangle 1$, $SMTT(10)$
$\langle 3 \rangle 4$. $fa[j] = DeltaVecAdd(fa[j-1], Elema(j))$ BY $\langle 1 \rangle 5$, $\langle 3 \rangle 2$ DEF $Defa$
$\langle 3 \rangle 5$. $fa[j-1] \in DeltaVecType$ BY $\langle 2 \rangle 2$, $\langle 3 \rangle 3$ DEF P
$\langle 3 \rangle 6$. $Elema(j) \in DeltaVecType$ BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 1 \rangle 6$
$\langle 3 \rangle 7$. $fa[j] \in DeltaVecType$ BY $\langle 3 \rangle 4$, $\langle 3 \rangle 5$, $\langle 3 \rangle 6$, $DeltaVecAddType$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 7$ DEF P
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $NatInduction$, Isa

$XXXb$ definitions are related to the sum of $RemoveAt(Q, n)$ skipping 0.

$\langle 1 \rangle$ DEFINE $Qb \triangleq RemoveAt(Q, n)$
$\langle 1 \rangle$ DEFINE $Elemb(i) \triangleq DeltaVecSeqSkipSum(0, Qb)!:!Elem(i)$
$\langle 1 \rangle$ DEFINE $f0b \triangleq DeltaVecZero$
$\langle 1 \rangle$ DEFINE $Defb(v, i) \triangleq DeltaVecAdd(v, Elemb(i))$
$\langle 1 \rangle$ DEFINE $fb \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0b$ ELSE $Defb(f[i-1], i)]$
$\langle 1 \rangle$ DEFINE $LenQb \triangleq Len(Qb)$
$\langle 1 \rangle$ HIDE DEF Qb, $Elemb$, $f0b$, $Defb$, fb, $LenQb$

$\langle 1 \rangle 8$. $Qb \in Seq(DeltaVecType)$ BY $RemoveAtProperties$ DEF Qb
$\langle 1 \rangle 9$. $Qb \in [1 \,..\, LenQb \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 8$, $LenAxiom$ DEF $LenQb$
$\langle 1 \rangle 10$. $LenQb \in Nat$ BY $\langle 1 \rangle 8$, $LenInNat$ DEF $LenQb$
$\langle 1 \rangle 11$. $DeltaVecSeqSkipSum(0, Qb) = fb[LenQb]$ BY DEF $DeltaVecSeqSkipSum$, fb, $f0b$, $Defb$, $Elemb$, $LenQb$

$\langle 1 \rangle 12$. $\forall i \in Nat : fb[i] =$ IF $i = 0$ THEN $f0b$ ELSE $Defb(fb[i-1], i)$
$\langle 2 \rangle$ SUFFICES $NatInductiveDefConclusion(fb, f0b, Defb)$ BY DEF $NatInductiveDefConclusion$
$\langle 2 \rangle$ SUFFICES $NatInductiveDefHypothesis(fb, f0b, Defb)$ BY $NatInductiveDef$
$\langle 2 \rangle$ QED BY DEF $NatInductiveDefHypothesis$, fb

$\langle 1 \rangle 13$. $\forall i \in Nat \setminus \{0\} : Elemb(i) \in DeltaVecType$
$\langle 2 \rangle$ SUFFICES ASSUME NEW $i \in Nat \setminus \{0\}$ PROVE $Elemb(i) \in DeltaVecType$ OBVIOUS
$\langle 2 \rangle 1$. CASE $0 < i \land i \leq LenQb$

$\langle 3\rangle 1.\ i \in 1 \,..\, \mathit{LenQb}$ BY $\langle 2\rangle 1,\ \langle 1\rangle 10,\ \mathit{DotDotDef},\ \mathit{SMTT}(10)$
$\langle 3\rangle 2.\ \mathit{Qb}[i] \in \mathit{DeltaVecType}$ BY $\langle 3\rangle 1,\ \langle 1\rangle 9$
$\langle 3\rangle$ QED BY $\langle 3\rangle 2,\ \langle 2\rangle 1$ DEF $\mathit{LenQb},\ \mathit{Elemb}$
$\langle 2\rangle 2.$ CASE $\neg(0 < i \land i \leq \mathit{LenQb})$
$\langle 3\rangle 1.\ \mathit{Elemb}(i) = \mathit{DeltaVecZero}$ BY $\langle 2\rangle 2$ DEF $\mathit{LenQb},\ \mathit{Elemb}$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ \mathit{DeltaVecZeroType}$
$\langle 2\rangle$ QED BY $\langle 2\rangle 1,\ \langle 2\rangle 2$

$\langle 1\rangle 14.\ \forall\, i \in \mathit{Nat} : \mathit{fb}[i] \in \mathit{DeltaVecType}$
$\langle 2\rangle$ DEFINE $P(i) \triangleq \mathit{fb}[i] \in \mathit{DeltaVecType}$
$\langle 2\rangle$ HIDE DEF P
$\langle 2\rangle$ SUFFICES $\forall\, i \in \mathit{Nat} : P(i)$ BY DEF P
$\langle 2\rangle 1.\ P(0)$
$\langle 3\rangle 1.\ \mathit{fb}[0] = \mathit{DeltaVecZero}$ BY $\langle 1\rangle 12$ DEF $f0b$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ \mathit{DeltaVecZeroType}$ DEF P
$\langle 2\rangle 2.$ ASSUME NEW $i \in \mathit{Nat},\ P(i)$ PROVE $P(i+1)$

Rewrite $(i, i+1)$ to $(j-1, j)$ to match inductive def.

$\langle 3\rangle 1.$ PICK $j : j = i + 1$ OBVIOUS
$\langle 3\rangle 2.\ j \in \mathit{Nat} \setminus \{0\}$ BY $\langle 3\rangle 1,\ \mathit{SMTT}(10)$
$\langle 3\rangle 3.\ j - 1 = i$ BY $\langle 3\rangle 1,\ \mathit{SMTT}(10)$
$\langle 3\rangle 4.\ \mathit{fb}[j] = \mathit{DeltaVecAdd}(\mathit{fb}[j-1],\ \mathit{Elemb}(j))$ BY $\langle 1\rangle 12,\ \langle 3\rangle 2$ DEF Defb
$\langle 3\rangle 5.\ \mathit{fb}[j-1] \in \mathit{DeltaVecType}$ BY $\langle 2\rangle 2,\ \langle 3\rangle 3$ DEF P
$\langle 3\rangle 6.\ \mathit{Elemb}(j) \in \mathit{DeltaVecType}$ BY $\langle 3\rangle 1,\ \langle 3\rangle 2,\ \langle 1\rangle 13$
$\langle 3\rangle 7.\ \mathit{fb}[j] \in \mathit{DeltaVecType}$ BY $\langle 3\rangle 4,\ \langle 3\rangle 5,\ \langle 3\rangle 6,\ \mathit{DeltaVecAddType}$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ \langle 3\rangle 7$ DEF P
$\langle 2\rangle$ QED BY $\langle 2\rangle 1,\ \langle 2\rangle 2,\ \mathit{NatInduction},\ \mathit{Isa}$

Now relate the two sums.

$\langle 1\rangle 15.\ \mathit{LenQb} = \mathit{LenQa} - 1$ BY $\mathit{RemoveAtProperties}$ DEF $\mathit{Qa},\ \mathit{Qb},\ \mathit{LenQa},\ \mathit{LenQb}$
$\langle 1\rangle 16.\ n \in 1 \,..\, \mathit{LenQa}$ BY DEF $\mathit{Qa},\ \mathit{LenQa}$
$\langle 1\rangle 17.\ 0 < n \land n \leq \mathit{LenQa}$ BY $\langle 1\rangle 3,\ \langle 1\rangle 16,\ \mathit{SMTT}(10)$

$\langle 1\rangle 18.\ \forall\, i \in \mathit{Nat} \setminus \{0\} : i \neq n \Rightarrow \mathit{Elema}(i) = \mathit{Elemb}(\text{IF } i < n \text{ THEN } i \text{ ELSE } i - 1)$
$\langle 2\rangle 1.$ SUFFICES ASSUME
NEW $\mathit{ia},\ \mathit{ia} \in \mathit{Nat} \setminus \{0\},\ \mathit{ia} \neq n,$
NEW $\mathit{ib},\ \mathit{ib} = \text{IF } \mathit{ia} < n \text{ THEN } \mathit{ia} \text{ ELSE } \mathit{ia} - 1$
PROVE $\mathit{Elema}(\mathit{ia}) = \mathit{Elemb}(\mathit{ib})$
OBVIOUS
$\langle 2\rangle 2.\ \forall\, i \in 1 \,..\, \mathit{LenQa} : i \neq n \Rightarrow \mathit{Qa}[i] = \mathit{Qb}[\text{IF } i < n \text{ THEN } i \text{ ELSE } i - 1]$
$\langle 3\rangle 1.\ \forall\, i \in 1 \,..\, \mathit{LenQa} : i \neq n \Rightarrow \mathit{Qa}[i] = \mathit{Qb}[\mathit{RemoveAt\_ForwardIndex}(\mathit{Qa},\ n,\ i)]$
$\langle 4\rangle 1.\ \mathit{RemoveAt\_MapForward}(\mathit{Qa},\ n)$ BY $\mathit{RemoveAtProperties}$ DEF Qa
$\langle 4\rangle$ QED BY $\langle 4\rangle 1$ DEF $\mathit{RemoveAt\_MapForward},\ \mathit{Qa},\ \mathit{Qb},\ \mathit{LenQa}$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1$ DEF $\mathit{RemoveAt\_ForwardIndex}$
$\langle 2\rangle 3.$ CASE $0 < \mathit{ia} \land \mathit{ia} \leq \mathit{LenQa}$
$\langle 3\rangle 1.\ 0 < \mathit{ib} \land \mathit{ib} \leq \mathit{LenQb}$ BY $\langle 1\rangle 3,\ \langle 1\rangle 10,\ \langle 1\rangle 15,\ \langle 1\rangle 16,\ \langle 2\rangle 1,\ \langle 2\rangle 3,\ \mathit{SMTT}(10)$
$\langle 3\rangle 2.\ \mathit{Elema}(\mathit{ia}) = \mathit{Qa}[\mathit{ia}]$ BY $\langle 2\rangle 3$ DEF $\mathit{Elema},\ \mathit{LenQa}$

$\langle 3\rangle 3.$ $Elemb(ib) = Qb[ib]$ BY $\langle 3\rangle 1$ DEF $Elemb$, $LenQb$
$\langle 3\rangle 4.$ $ia \in 1 \,..\, LenQa$ BY $\langle 1\rangle 3$, $\langle 1\rangle 10$, $\langle 2\rangle 1$, $\langle 2\rangle 3$, $DotDotDef$, $SMTT(10)$
$\langle 3\rangle 5.$ $Qa[ia] = Qb[ib]$ BY $\langle 2\rangle 1$, $\langle 2\rangle 2$, $\langle 3\rangle 4$
$\langle 3\rangle$ QED BY $\langle 3\rangle 2$, $\langle 3\rangle 3$, $\langle 3\rangle 5$
$\langle 2\rangle 4.$ CASE $\neg(0 < ia \land ia \leq LenQa)$
$\langle 3\rangle 1.$ $\neg(0 < ib \land ib \leq LenQb)$ BY $\langle 1\rangle 3$, $\langle 1\rangle 10$, $\langle 1\rangle 15$, $\langle 1\rangle 16$, $\langle 2\rangle 1$, $\langle 2\rangle 4$, $SMTT(10)$
$\langle 3\rangle 2.$ $Elema(ia) = DeltaVecZero$ BY $\langle 2\rangle 4$ DEF $Elema$, $LenQa$
$\langle 3\rangle 3.$ $Elemb(ib) = DeltaVecZero$ BY $\langle 3\rangle 1$ DEF $Elemb$, $LenQb$
$\langle 3\rangle$ QED BY $\langle 3\rangle 2$, $\langle 3\rangle 3$
$\langle 2\rangle$ QED BY $\langle 2\rangle 3$, $\langle 2\rangle 4$

$\langle 1\rangle 19.$ $Elema(n) = Qa[n]$
$\langle 2\rangle 1.$ $0 < n \land n \leq LenQa$ BY $\langle 1\rangle 3$, $\langle 1\rangle 16$, $SMTT(10)$
$\langle 2\rangle$ QED BY $\langle 2\rangle 1$ DEF $Elema$, $LenQa$

Each sum evaluates its recursive function at the length of its sequence.

$\langle 1\rangle 20.$ $DeltaVecAdd(Qa[n], fb[LenQb]) = fa[LenQa]$
$\langle 2\rangle$ DEFINE $P(i) \triangleq fa[i] =$ IF $i < n$ THEN $fb[i]$ ELSE $DeltaVecAdd(Qa[n], fb[i-1])$
$\langle 2\rangle$ HIDE DEF P
$\langle 2\rangle 1.$ $\forall\, i \in Nat : P(i)$
$\langle 3\rangle 1.$ $P(0)$
$\langle 4\rangle$ SUFFICES $fa[0] = fb[0]$ BY $\langle 1\rangle 17$ DEF P
$\langle 4\rangle$ QED BY $\langle 1\rangle 5$, $\langle 1\rangle 12$ DEF $f0a$, $f0b$
$\langle 3\rangle 2.$ ASSUME NEW $i \in Nat$, $P(i)$ PROVE $P(i+1)$
$\langle 4\rangle 1.$ PICK $j : j = i + 1$ OBVIOUS
$\langle 4\rangle 2.$ $j \in Nat \setminus \{0\}$ BY $\langle 4\rangle 1$, $SMTT(10)$
$\langle 4\rangle 3.$ $j - 1 = i$ BY $\langle 4\rangle 1$, $SMTT(10)$

$\langle 4\rangle 4.$ CASE $j < n$

Before the removal point.

$\langle 5\rangle 1.$ $j \neq n$ BY $\langle 4\rangle 1$, $\langle 4\rangle 4$, $SMTT(10)$
$\langle 5\rangle 2.$ $j - 1 < n$ BY $\langle 4\rangle 1$, $\langle 4\rangle 4$, $SMTT(10)$
$\langle 5\rangle 3.$ $fa[j-1] = fb[j-1]$ BY $\langle 3\rangle 2$, $\langle 4\rangle 3$, $\langle 5\rangle 2$ DEF P
$\langle 5\rangle 4.$ $Elema(j) = Elemb(j)$ BY $\langle 1\rangle 18$, $\langle 4\rangle 2$, $\langle 4\rangle 4$, $\langle 5\rangle 1$
$\langle 5\rangle 5.$ $fa[j] = DeltaVecAdd(fa[j-1], Elema(j))$ BY $\langle 1\rangle 5$, $\langle 4\rangle 2$ DEF $Defa$
$\langle 5\rangle 6.$ $fb[j] = DeltaVecAdd(fb[j-1], Elemb(j))$ BY $\langle 1\rangle 12$, $\langle 4\rangle 2$ DEF $Defb$
$\langle 5\rangle 7.$ $fa[j] = fb[j]$ BY $\langle 5\rangle 3$, $\langle 5\rangle 4$, $\langle 5\rangle 5$, $\langle 5\rangle 6$
$\langle 5\rangle$ QED BY $\langle 4\rangle 1$, $\langle 4\rangle 4$, $\langle 5\rangle 7$ DEF P

$\langle 4\rangle 5.$ CASE $j = n$

At the removal point.

$\langle 5\rangle 1.$ $\neg(j < n)$ BY $\langle 4\rangle 1$, $\langle 4\rangle 5$, $SMTT(10)$
$\langle 5\rangle 2.$ $j - 1 < n$ BY $\langle 4\rangle 1$, $\langle 4\rangle 5$, $SMTT(10)$
$\langle 5\rangle 3.$ $fa[j-1] = fb[j-1]$ BY $\langle 3\rangle 2$, $\langle 4\rangle 3$, $\langle 5\rangle 2$ DEF P
$\langle 5\rangle 4.$ $Elema(j) = Qa[n]$ BY $\langle 1\rangle 19$, $\langle 4\rangle 5$
$\langle 5\rangle 5.$ $fa[j] = DeltaVecAdd(fa[j-1], Elema(j))$ BY $\langle 1\rangle 5$, $\langle 4\rangle 2$ DEF $Defa$

$\langle 5 \rangle 6.$ $fa[j] = DeltaVecAdd(fb[j-1],\ Qa[n])$ BY $\langle 5 \rangle 3,\ \langle 5 \rangle 4,\ \langle 5 \rangle 5$
$\langle 5 \rangle 7.$ $fa[j] = DeltaVecAdd(Qa[n],\ fb[j-1])$
$\langle 6 \rangle 1.$ $Qa[n] \in DeltaVecType$ BY $\langle 1 \rangle 2,\ \langle 1 \rangle 16$
$\langle 6 \rangle 2.$ $fb[j-1] \in DeltaVecType$ BY $\langle 1 \rangle 14,\ \langle 4 \rangle 3$
$\langle 6 \rangle$ QED BY $\langle 5 \rangle 6,\ \langle 6 \rangle 1,\ \langle 6 \rangle 2,\ DeltaVecAddCommutative$
$\langle 5 \rangle$ QED BY $\langle 4 \rangle 1,\ \langle 5 \rangle 1,\ \langle 5 \rangle 7$ DEF P

$\langle 4 \rangle 6.$ CASE $j > n$

After the removal point.

$\langle 5 \rangle 1.$ $j \neq n$ BY $\langle 4 \rangle 1,\ \langle 4 \rangle 6,\ SMTT(10)$
$\langle 5 \rangle 2.$ $\neg(j < n)$ BY $\langle 4 \rangle 1,\ \langle 4 \rangle 6,\ SMTT(10)$
$\langle 5 \rangle 3.$ $\neg(j - 1 < n)$ BY $\langle 4 \rangle 1,\ \langle 4 \rangle 6,\ SMTT(10)$
$\langle 5 \rangle 4.$ $i \in Nat \setminus \{0\}$ BY $\langle 4 \rangle 3,\ \langle 5 \rangle 3,\ SMTT(10)$
$\langle 5 \rangle 5.$ $i - 1 \in Nat$ BY $\langle 4 \rangle 3,\ \langle 5 \rangle 3,\ SMTT(10)$
$\langle 5 \rangle 6.$ $fa[j-1] = DeltaVecAdd(Qa[n],\ fb[i-1])$ BY $\langle 3 \rangle 2,\ \langle 4 \rangle 3,\ \langle 5 \rangle 3$ DEF P
$\langle 5 \rangle 7.$ $Elema(j) = Elemb(i)$ BY $\langle 1 \rangle 18,\ \langle 4 \rangle 2,\ \langle 4 \rangle 3,\ \langle 5 \rangle 1,\ \langle 5 \rangle 2$
$\langle 5 \rangle 8.$ $fa[j] = DeltaVecAdd(fa[j-1],\ Elema(j))$ BY $\langle 1 \rangle 5,\ \langle 4 \rangle 2$ DEF $Defa$
$\langle 5 \rangle 9.$ $fb[i] = DeltaVecAdd(fb[i-1],\ Elemb(i))$ BY $\langle 1 \rangle 12,\ \langle 5 \rangle 4$ DEF $Defb$
$\langle 5 \rangle 10.$ $fa[j] = DeltaVecAdd(fa[j-1],\ Elemb(i))$ BY $\langle 5 \rangle 8,\ \langle 5 \rangle 7$
$\langle 5 \rangle 11.$ $fa[j] = DeltaVecAdd(DeltaVecAdd(Qa[n],\ fb[i-1]),\ Elemb(i))$ BY $\langle 5 \rangle 10,\ \langle 5 \rangle 6$
$\langle 5 \rangle 12.$ $fa[j] = DeltaVecAdd(Qa[n],\ DeltaVecAdd(fb[i-1],\ Elemb(i)))$
$\langle 6 \rangle 1.$ $Qa[n] \in DeltaVecType$ BY $\langle 1 \rangle 2,\ \langle 1 \rangle 16$
$\langle 6 \rangle 2.$ $fb[i-1] \in DeltaVecType$ BY $\langle 1 \rangle 14,\ \langle 5 \rangle 5$
$\langle 6 \rangle 3.$ $Elemb(i) \in DeltaVecType$ BY $\langle 1 \rangle 13,\ \langle 5 \rangle 4$
$\langle 6 \rangle$ QED BY $\langle 5 \rangle 11,\ \langle 6 \rangle 1,\ \langle 6 \rangle 2,\ \langle 6 \rangle 3,\ DeltaVecAddAssociative,\ Isa$
$\langle 5 \rangle 13.$ $fa[j] = DeltaVecAdd(Qa[n],\ fb[i])$ BY $\langle 5 \rangle 9,\ \langle 5 \rangle 12$
$\langle 5 \rangle$ QED BY $\langle 4 \rangle 1,\ \langle 5 \rangle 2,\ \langle 5 \rangle 13,\ Isa$ DEF P

$\langle 4 \rangle$ QED BY $\langle 4 \rangle 2,\ \langle 4 \rangle 4,\ \langle 4 \rangle 5,\ \langle 4 \rangle 6,\ SMTT(10)$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 3 \rangle 2,\ NatInduction,\ Isa$

$\langle 2 \rangle 2.$ $\neg(LenQa < n)$ BY $\langle 1 \rangle 3,\ \langle 1 \rangle 16,\ SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 3,\ \langle 1 \rangle 15,\ \langle 2 \rangle 1,\ \langle 2 \rangle 2$ DEF P

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 4,\ \langle 1 \rangle 11,\ \langle 1 \rangle 20$ DEF $Qa,\ Qb$

Adding a delta vector to one of a sequence of delta vectors.

$AddAt$ makes a sequence of delta vectors.

THEOREM $DeltaVecSeqAddAtType \triangleq$
  ASSUME
    NEW $Q \in Seq(DeltaVecType)$,
    NEW $n \in 1 \,..\, Len(Q)$,
    NEW $d \in DeltaVecType$
  PROVE
  LET $R \triangleq DeltaVecSeqAddAt(Q, n, d)$ IN
  $\wedge R \in Seq(DeltaVecType)$
  $\wedge Len(Q) = Len(R)$
PROOF
  $\langle 1 \rangle$ DEFINE $R \triangleq DeltaVecSeqAddAt(Q, n, d)$
  $\langle 1 \rangle$ DEFINE $LenQ \triangleq Len(Q)$
  $\langle 1 \rangle$ DEFINE $LenR \triangleq Len(R)$
  $\langle 1 \rangle 1$. $LenQ \in Nat$ BY $LenInNat$
  $\langle 1 \rangle$ SUFFICES $R \in Seq(DeltaVecType) \wedge LenQ = LenR$ OBVIOUS
  $\langle 1 \rangle$ HIDE DEF R, $LenQ$, $LenR$
  $\langle 1 \rangle 2$. $R \in [1 \,..\, LenQ \rightarrow DeltaVecType]$
    $\langle 2 \rangle 1$. $n \in 1 \,..\, LenQ$ BY DEF $LenQ$
    $\langle 2 \rangle 2$. $Q \in [1 \,..\, LenQ \rightarrow DeltaVecType]$ BY $LenAxiom$ DEF $LenQ$
    $\langle 2 \rangle 3$. $Q[n] \in DeltaVecType$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$
    $\langle 2 \rangle 4$. $DeltaVecAdd(Q[n], d) \in DeltaVecType$ BY $\langle 2 \rangle 3$, $DeltaVecAddType$
    $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $\langle 2 \rangle 4$ DEF R, $DeltaVecSeqAddAt$
  $\langle 1 \rangle 3$. $R \in Seq(DeltaVecType)$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $SeqDef$
  $\langle 1 \rangle 4$. $LenQ = LenR$
    $\langle 2 \rangle 1$. DOMAIN $R = 1 \,..\, LenQ$ BY $\langle 1 \rangle 2$
    $\langle 2 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 3$, $\langle 2 \rangle 1$, $LenDomain$ DEF $LenR$
  $\langle 1 \rangle$ QED BY $\langle 1 \rangle 3$, $\langle 1 \rangle 4$

Adding a value d to the sum of a sequence of delta vectors gives the same result as adding d to one of the elements of the sequence and then taking the sum.

THEOREM $DeltaVecSeqSkipSumAddAt \triangleq$
  ASSUME
    NEW $Q \in Seq(DeltaVecType)$,
    NEW $n \in 1 \,..\, Len(Q)$,
    NEW $d \in DeltaVecType$
  PROVE

$DeltaVecAdd(DeltaVecSeqSkipSum(0, Q), d) = DeltaVecSeqSkipSum(0, DeltaVecSeqAddAt(Q, n, d))$

PROOF

$XXXa$ definitions are related to the sum of Q.

$\langle 1 \rangle$ DEFINE $Qa \triangleq Q$
$\langle 1 \rangle$ DEFINE $Elema(i) \triangleq DeltaVecSeqSkipSum(0, Qa)!:!Elem(i)$
$\langle 1 \rangle$ DEFINE $f0a \triangleq DeltaVecZero$
$\langle 1 \rangle$ DEFINE $Defa(v, i) \triangleq DeltaVecAdd(v, Elema(i))$
$\langle 1 \rangle$ DEFINE $fa \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0a$ ELSE $Defa(f[i-1], i)]$
$\langle 1 \rangle$ DEFINE $LenQa \triangleq Len(Qa)$
$\langle 1 \rangle 1.$ $Qa \in Seq(DeltaVecType)$ OBVIOUS
$\langle 1 \rangle 2.$ $Qa \in [1 \,..\, LenQa \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 1$, $LenAxiom$
$\langle 1 \rangle 3.$ $LenQa \in Nat$ BY $LenInNat$
$\langle 1 \rangle 4.$ $DeltaVecSeqSkipSum(0, Qa) = fa[LenQa]$ BY DEF $DeltaVecSeqSkipSum$

$\langle 1 \rangle 5.$ $\forall i \in Nat : fa[i] =$ IF $i = 0$ THEN $f0a$ ELSE $Defa(fa[i-1], i)$
  $\langle 2 \rangle$ HIDE DEF $f0a$, $Defa$, fa
  $\langle 2 \rangle$ SUFFICES $NatInductiveDefConclusion(fa, f0a, Defa)$ BY DEF $NatInductiveDefConclusion$
  $\langle 2 \rangle$ SUFFICES $NatInductiveDefHypothesis(fa, f0a, Defa)$ BY $NatInductiveDef$
  $\langle 2 \rangle$ QED BY DEF $NatInductiveDefHypothesis$, fa

$XXXb$ definitions are related to the sum of Q except d added to $Q[n]$.

$\langle 1 \rangle$ DEFINE $Qb \triangleq DeltaVecSeqAddAt(Qa, n, d)$
$\langle 1 \rangle$ DEFINE $Elemb(i) \triangleq DeltaVecSeqSkipSum(0, Qb)!:!Elem(i)$
$\langle 1 \rangle$ DEFINE $f0b \triangleq DeltaVecZero$
$\langle 1 \rangle$ DEFINE $Defb(v, i) \triangleq DeltaVecAdd(v, Elemb(i))$
$\langle 1 \rangle$ DEFINE $fb \triangleq$ CHOOSE $f : f = [i \in Nat \mapsto$ IF $i = 0$ THEN $f0b$ ELSE $Defb(f[i-1], i)]$
$\langle 1 \rangle$ DEFINE $LenQb \triangleq Len(Qb)$
$\langle 1 \rangle 6.$ $Qb \in Seq(DeltaVecType)$ BY $DeltaVecSeqAddAtType$
$\langle 1 \rangle 7.$ $Qb \in [1 \,..\, LenQb \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 6$, $LenAxiom$
$\langle 1 \rangle 8.$ $LenQb \in Nat$ BY $\langle 1 \rangle 6$, $LenInNat$
$\langle 1 \rangle 9.$ $DeltaVecSeqSkipSum(0, Qb) = fb[LenQb]$ BY DEF $DeltaVecSeqSkipSum$

$\langle 1 \rangle 10.$ $\forall i \in Nat : fb[i] =$ IF $i = 0$ THEN $f0b$ ELSE $Defb(fb[i-1], i)$
  $\langle 2 \rangle$ HIDE DEF $f0b$, $Defb$, fb
  $\langle 2 \rangle$ SUFFICES $NatInductiveDefConclusion(fb, f0b, Defb)$ BY DEF $NatInductiveDefConclusion$
  $\langle 2 \rangle$ SUFFICES $NatInductiveDefHypothesis(fb, f0b, Defb)$ BY $NatInductiveDef$
  $\langle 2 \rangle$ QED BY DEF $NatInductiveDefHypothesis$, fb

Miscellaneous facts about $LenQa$, $LenQb$, and n.

$\langle 1 \rangle 11.$ $LenQb = LenQa$ BY $DeltaVecSeqAddAtType$
$\langle 1 \rangle 12.$ $0 < n \wedge n \leq LenQa$
  $\langle 2 \rangle 1.$ $n \in 1 \,..\, LenQa$ BY DEF $LenQa$, Qa
  $\langle 2 \rangle$ HIDE DEF $LenQa$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 1 \rangle 3$, $DotDotDef$, $SMTT(10)$

Each $Elema$ is a delta vec.

$\langle 1\rangle 13.\ \forall\, i \in Nat \setminus \{0\} : Elema(i) \in DeltaVecType$
$\quad\langle 2\rangle$ SUFFICES ASSUME NEW $i \in Nat \setminus \{0\}$ PROVE $Elema(i) \in DeltaVecType$ OBVIOUS
$\quad\langle 2\rangle$ HIDE DEF $LenQa$
$\quad\langle 2\rangle 1.$ CASE $0 < i \wedge i \leq LenQa$
$\quad\quad\langle 3\rangle 1.\ i \in 1 \,..\, LenQa$ BY $\langle 2\rangle 1,\ \langle 1\rangle 8,\ DotDotDef,\ SMTT(10)$
$\quad\quad\langle 3\rangle 2.\ Qa[i] \in DeltaVecType$ BY $\langle 3\rangle 1,\ \langle 1\rangle 6,\ LenAxiom$ DEF $LenQa$
$\quad\quad\langle 3\rangle$ QED BY $\langle 3\rangle 2,\ \langle 2\rangle 1$ DEF $LenQa$
$\quad\langle 2\rangle 2.$ CASE $\neg(0 < i \wedge i \leq LenQa)$
$\quad\quad\langle 3\rangle 1.\ Elema(i) = DeltaVecZero$ BY $\langle 2\rangle 2$ DEF $LenQa$
$\quad\quad\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ DeltaVecZeroType,\ DeltaVecAddZero$
$\quad\langle 2\rangle$ QED BY $\langle 2\rangle 1,\ \langle 2\rangle 2$

Each *Elemb* is a delta vec.

$\langle 1\rangle 14.\ \forall\, i \in Nat \setminus \{0\} : Elemb(i) \in DeltaVecType$
$\quad\langle 2\rangle$ SUFFICES ASSUME NEW $i \in Nat \setminus \{0\}$ PROVE $Elemb(i) \in DeltaVecType$ OBVIOUS
$\quad\langle 2\rangle$ HIDE DEF $LenQb$
$\quad\langle 2\rangle 1.$ CASE $0 < i \wedge i \leq LenQb$
$\quad\quad\langle 3\rangle 1.\ i \in 1 \,..\, LenQb$ BY $\langle 2\rangle 1,\ \langle 1\rangle 8,\ DotDotDef,\ SMTT(10)$
$\quad\quad\langle 3\rangle 2.\ Qb[i] \in DeltaVecType$ BY $\langle 3\rangle 1,\ \langle 1\rangle 6,\ LenAxiom$ DEF $LenQb$
$\quad\quad\langle 3\rangle$ QED BY $\langle 3\rangle 2,\ \langle 2\rangle 1$ DEF $LenQb$
$\quad\langle 2\rangle 2.$ CASE $\neg(0 < i \wedge i \leq LenQb)$
$\quad\quad\langle 3\rangle 1.\ Elemb(i) = DeltaVecZero$ BY $\langle 2\rangle 2$ DEF $LenQb$
$\quad\quad\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ DeltaVecZeroType,\ DeltaVecAddZero$
$\quad\langle 2\rangle$ QED BY $\langle 2\rangle 1,\ \langle 2\rangle 2$

Each *Elemb* is the same as *Elema* except at $[n]$, where it is $Elema(n) + d$.

$\langle 1\rangle 15.\ \forall\, i \in Nat \setminus \{0\} :$
$\quad\quad Elemb(i) =$ IF $i = n$ THEN $DeltaVecAdd(Elema(i),\ d)$ ELSE $Elema(i)$
$\quad\langle 2\rangle$ SUFFICES ASSUME NEW $i \in Nat \setminus \{0\}$
$\quad\quad$ PROVE $Elemb(i) =$ IF $i \quad = n$ THEN $DeltaVecAdd(Elema(i),\ d)$ ELSE $Elema(i)$
$\quad\quad$ OBVIOUS
$\quad\langle 2\rangle$ HIDE DEF $Qa,\ Qb,\ LenQa,\ LenQb,\ Elema,\ Elemb$
$\quad\langle 2\rangle 1.$ CASE $0 < i \wedge i \leq LenQa$
$\quad\quad\langle 3\rangle 1.\ 0 < i \wedge i \leq LenQa \wedge i \in 1 \,..\, LenQa$ BY $\langle 2\rangle 1,\ \langle 1\rangle 3,\ DotDotDef,\ SMTT(10)$
$\quad\quad\langle 3\rangle 2.\ 0 < i \wedge i \leq LenQb \wedge i \in 1 \,..\, LenQb$ BY $\langle 3\rangle 1,\ \langle 1\rangle 11$
$\quad\quad\langle 3\rangle 3.\ Elema(i) = Qa[i]$ BY $\langle 3\rangle 1$ DEF $LenQa,\ Elema$
$\quad\quad\langle 3\rangle 4.\ Elemb(i) = Qb[i]$ BY $\langle 3\rangle 2$ DEF $LenQb,\ Elemb$
$\quad\quad\langle 3\rangle 5.$ CASE $i = n$
$\quad\quad\quad\langle 4\rangle 1.\ Qb[i] = DeltaVecAdd(Qa[i],\ d)$
$\quad\quad\quad\quad$ BY $\langle 1\rangle 2,\ \langle 1\rangle 3,\ \langle 3\rangle 1,\ \langle 3\rangle 5$ DEF $DeltaVecSeqAddAt,\ Qb$
$\quad\quad\quad\langle 4\rangle$ QED BY $\langle 4\rangle 1,\ \langle 3\rangle 3,\ \langle 3\rangle 4,\ \langle 3\rangle 5$
$\quad\quad\langle 3\rangle 6.$ CASE $i \neq n$
$\quad\quad\quad\langle 4\rangle 1.\ Qb[i] = Qa[i]$
$\quad\quad\quad\quad$ BY $\langle 1\rangle 2,\ \langle 1\rangle 3,\ \langle 3\rangle 1,\ \langle 3\rangle 6$ DEF $DeltaVecSeqAddAt,\ Qb$
$\quad\quad\quad\langle 4\rangle$ QED BY $\langle 4\rangle 1,\ \langle 3\rangle 3,\ \langle 3\rangle 4,\ \langle 3\rangle 6$
$\quad\quad\langle 3\rangle$ QED BY $\langle 3\rangle 5,\ \langle 3\rangle 6$
$\quad\langle 2\rangle 2.$ CASE $\neg(0 < i \wedge i \leq LenQa)$

$\langle 3 \rangle 1.$ $\neg(0 < i \wedge i \leq LenQa)$ BY $\langle 2 \rangle 2$
$\langle 3 \rangle 2.$ $\neg(0 < i \wedge i \leq LenQb)$ BY $\langle 3 \rangle 1$, $\langle 1 \rangle 11$
$\langle 3 \rangle 3.$ $Elema(i) = DeltaVecZero$ BY $\langle 3 \rangle 1$ DEF $LenQa$, $Elema$
$\langle 3 \rangle 4.$ $Elemb(i) = DeltaVecZero$ BY $\langle 3 \rangle 2$ DEF $LenQb$, $Elemb$
$\langle 3 \rangle 5.$ $i \neq n$ BY $\langle 3 \rangle 1$, $\langle 1 \rangle 12$, $SMTT(10)$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 3$, $\langle 3 \rangle 4$, $\langle 3 \rangle 5$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$

$fa[i]$ is a delta vector

$\langle 1 \rangle 16.$ $\forall\, i \in Nat : fa[i] \in DeltaVecType$
$\langle 2 \rangle$ DEFINE $P(i) \triangleq fa[i] \in DeltaVecType$
$\langle 2 \rangle$ HIDE DEF $LenQa$, $LenQb$, fa, fb
$\langle 2 \rangle$ SUFFICES $\forall\, i \in Nat : P(i)$ OBVIOUS
$\langle 2 \rangle 1.$ $P(0)$
$\langle 3 \rangle 1.$ $fa[0] = DeltaVecZero$ BY $\langle 1 \rangle 5$ DEF fa, $f0a$
$\langle 3 \rangle 2.$ $fa[0] \in DeltaVecType$ BY $\langle 3 \rangle 1$, $DeltaVecZeroType$, $DeltaVecAddZero$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$
$\langle 2 \rangle 2.$ $\forall\, i \in Nat : P(i) \Rightarrow P(i+1)$
$\langle 3 \rangle 1.$ SUFFICES ASSUME NEW $i \in Nat$, $P(i)$ PROVE $P(i+1)$ OBVIOUS
$\langle 3 \rangle 2.$ Trivial facts to help the prover match known facts or proof obligations.
$\wedge\ i + 1 \in Nat$
$\wedge\ i + 1 \in Nat \setminus \{0\}$
$\wedge\ i + 1 \neq 0$
$\wedge\ (i + 1) - 1 = i$
BY $SMTT(10)$
$\langle 3 \rangle 3.$ $fa[i+1] = DeltaVecAdd(fa[i], Elema(i+1))$ BY $\langle 3 \rangle 2$, $\langle 1 \rangle 5$
$\langle 3 \rangle 4.$ $fa[i] \in DeltaVecType$ BY $\langle 3 \rangle 1$
$\langle 3 \rangle 5.$ $Elema(i+1) \in DeltaVecType$ BY $\langle 3 \rangle 2$, $\langle 1 \rangle 13$
$\langle 3 \rangle 6.$ $fa[i+1] \in DeltaVecType$ BY $\langle 3 \rangle 3$, $\langle 3 \rangle 4$, $\langle 3 \rangle 5$, $DeltaVecAddType$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 6$
$\langle 2 \rangle$ HIDE DEF P
$\langle 2 \rangle$ QED BY ONLY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $NatInduction$, Isa

$fb[i]$ is a delta vector

$\langle 1 \rangle 17.$ $\forall\, i \in Nat : fb[i] \in DeltaVecType$
$\langle 2 \rangle$ DEFINE $P(i) \triangleq fb[i] \in DeltaVecType$
$\langle 2 \rangle$ HIDE DEF $LenQa$, $LenQb$, fa, fb
$\langle 2 \rangle$ SUFFICES $\forall\, i \in Nat : P(i)$ OBVIOUS
$\langle 2 \rangle 1.$ $P(0)$
$\langle 3 \rangle 1.$ $fb[0] = DeltaVecZero$ BY $\langle 1 \rangle 10$ DEF fb, $f0b$
$\langle 3 \rangle 2.$ $fb[0] \in DeltaVecType$ BY $\langle 3 \rangle 1$, $DeltaVecZeroType$, $DeltaVecAddZero$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$
$\langle 2 \rangle 2.$ $\forall\, i \in Nat : P(i) \Rightarrow P(i+1)$
$\langle 3 \rangle 1.$ SUFFICES ASSUME NEW $i \in Nat$, $P(i)$ PROVE $P(i+1)$ OBVIOUS
$\langle 3 \rangle 2.$ Trivial facts to help the prover match known facts or proof obligations.
$\wedge\ i + 1 \in Nat$
$\wedge\ i + 1 \in Nat \setminus \{0\}$

$\land\ i+1 \neq 0$
$\land\ (i+1)-1 = i$
BY $SMTT(10)$
$\langle 3\rangle 3.\ fb[i+1] = DeltaVecAdd(fb[i],\ Elemb(i+1))$ BY $\langle 3\rangle 2,\ \langle 1\rangle 10$
$\langle 3\rangle 4.\ fb[i] \in DeltaVecType$ BY $\langle 3\rangle 1$
$\langle 3\rangle 5.\ Elemb(i+1) \in DeltaVecType$ BY $\langle 3\rangle 2,\ \langle 1\rangle 14$
$\langle 3\rangle 6.\ fb[i+1] \in DeltaVecType$ BY $\langle 3\rangle 3,\ \langle 3\rangle 4,\ \langle 3\rangle 5,\ DeltaVecAddType$
$\langle 3\rangle$ QED BY $\langle 3\rangle 6$
$\langle 2\rangle$ HIDE DEF P
$\langle 2\rangle$ QED BY ONLY $\langle 2\rangle 1,\ \langle 2\rangle 2,\ NatInduction,\ Isa$

Each sum evaluates its recursive function at the length of its sequence.

$\langle 1\rangle$ DEFINE $AddD(v)\ \triangleq\ DeltaVecAdd(v,\ d)$
$\langle 1\rangle 18.\ fb[LenQb] = AddD(fa[LenQa])$
$\langle 2\rangle$ DEFINE $P(i)\ \triangleq\ fb[i] =$ IF $i < n$ THEN $fa[i]$ ELSE $AddD(fa[i])$
$\langle 2\rangle$ HIDE DEF $LenQa,\ LenQb,\ fa,\ fb,\ Elema,\ Elemb,\ AddD$
$\langle 2\rangle$ SUFFICES $\forall\, i \in Nat : P(i)$
$\langle 3\rangle\ LenQa \in Nat$ BY $\langle 1\rangle 3$
$\langle 3\rangle\ LenQb \in Nat$ BY $\langle 1\rangle 8$
$\langle 3\rangle\ LenQa = LenQb$ BY $\langle 1\rangle 11$
$\langle 3\rangle\ n \leq LenQa$ BY $\langle 1\rangle 12$
$\langle 3\rangle$ QED BY $SMTT(10)$
$\langle 2\rangle 1.\ P(0)$
$\langle 3\rangle 1.\ 0 < n$ BY $\langle 1\rangle 12$
$\langle 3\rangle 2.\ fb[0] = DeltaVecZero$ BY $\langle 1\rangle 10$ DEF $fb,\ f0b$
$\langle 3\rangle 6.\ fa[0] = DeltaVecZero$ BY $\langle 1\rangle 5$ DEF $fa,\ f0a$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ \langle 3\rangle 2,\ \langle 3\rangle 6$
$\langle 2\rangle 2.\ \forall\, i \in Nat : P(i) \Rightarrow P(i+1)$
$\langle 3\rangle 1.$ SUFFICES ASSUME NEW $i \in Nat,\ P(i)$ PROVE $P(i+1)$ OBVIOUS
$\langle 3\rangle 2.$ Trivial facts to help the prover match known facts or proof obligations.
$\land\ i+1 \in Nat$
$\land\ i+1 \in Nat \setminus \{0\}$
$\land\ i+1 \neq 0$
$\land\ (i+1)-1 = i$
BY $SMTT(10)$
$\langle 3\rangle$ DEFINE $fai\ \triangleq\ fa[i]$
$\langle 3\rangle$ DEFINE $fbi\ \triangleq\ fb[i]$
$\langle 3\rangle$ DEFINE $fai1\ \triangleq\ fa[i+1]$
$\langle 3\rangle$ DEFINE $fbi1\ \triangleq\ fb[i+1]$
$\langle 3\rangle$ DEFINE $vai1\ \triangleq\ Elema(i+1)$
$\langle 3\rangle$ DEFINE $vbi1\ \triangleq\ Elemb(i+1)$
$\langle 3\rangle 3.\ fai1 = DeltaVecAdd(fai,\ vai1)$ BY $\langle 3\rangle 2,\ \langle 1\rangle 5$
$\langle 3\rangle 4.\ fbi1 = DeltaVecAdd(fbi,\ vbi1)$ BY $\langle 3\rangle 2,\ \langle 1\rangle 10$
$\langle 3\rangle 5.\ fai \in DeltaVecType$ BY $\langle 3\rangle 2,\ \langle 1\rangle 16$
$\langle 3\rangle 6.\ vai1 \in DeltaVecType$ BY $\langle 3\rangle 2,\ \langle 1\rangle 13$
$\langle 3\rangle 7.$ CASE $i+1 \neq n$

$\langle 4\rangle 1.$ $vai1 = vbi1$ BY $\langle 3\rangle 2$, $\langle 3\rangle 7$, $\langle 1\rangle 15$
$\langle 4\rangle 2.$ CASE $i < n$
$\langle 5\rangle 1.$ $fbi \quad = fai$ BY $\langle 4\rangle 2$, $\langle 3\rangle 1$
$\langle 5\rangle 2.$ $i + 1 < n$ BY $\langle 4\rangle 2$, $\langle 3\rangle 7$, $SMTT(10)$
$\langle 5\rangle$ SUFFICES $fai1 = fbi1$ BY $\langle 5\rangle 2$
$\langle 5\rangle$ HIDE DEF fai, fbi, $fai1$, $fbi1$, $vai1$, $vbi1$
$\langle 5\rangle 3.$ $fbi1 = DeltaVecAdd(fai, vai1)$ BY $\langle 3\rangle 4$, $\langle 5\rangle 1$, $\langle 4\rangle 1$
$\langle 5\rangle 4.$ $fbi1 = fai1$ BY $\langle 5\rangle 3$, $\langle 3\rangle 3$
$\langle 5\rangle$ QED BY $\langle 5\rangle 4$
$\langle 4\rangle 3.$ CASE $\neg(i < n)$
$\langle 5\rangle 1.$ $fbi = AddD(fai)$ BY $\langle 4\rangle 3$, $\langle 3\rangle 1$
$\langle 5\rangle 2.$ $\neg(i + 1 < n)$ BY $\langle 4\rangle 3$, $\langle 3\rangle 7$, $SMTT(10)$
$\langle 5\rangle$ SUFFICES $fbi1 = AddD(fai1)$ BY $\langle 5\rangle 2$
$\langle 5\rangle$ HIDE DEF fai, fbi, $fai1$, $fbi1$, $vai1$, $vbi1$
$\langle 5\rangle 3.$ $fbi1 = DeltaVecAdd(DeltaVecAdd(fai, d), vai1)$ BY $\langle 3\rangle 4$, $\langle 5\rangle 1$, $\langle 4\rangle 1$ DEF $AddD$
$\langle 5\rangle 4.$ $fbi1 = DeltaVecAdd(fai, DeltaVecAdd(d, vai1))$ BY $\langle 5\rangle 3$, $\langle 3\rangle 5$, $\langle 3\rangle 6$, $DeltaVecAddAssociative$
$\langle 5\rangle 5.$ $fbi1 = DeltaVecAdd(fai, DeltaVecAdd(vai1, d))$ BY $\langle 5\rangle 4$, $\langle 3\rangle 5$, $\langle 3\rangle 6$, $DeltaVecAddCommutative$
$\langle 5\rangle 6.$ $fbi1 = DeltaVecAdd(DeltaVecAdd(fai, vai1), d)$ BY $\langle 5\rangle 5$, $\langle 3\rangle 5$, $\langle 3\rangle 6$, $DeltaVecAddAssociative$
$\langle 5\rangle 7.$ $fbi1 = AddD(fai1)$ BY $\langle 5\rangle 6$, $\langle 3\rangle 3$ DEF $AddD$
$\langle 5\rangle$ QED BY $\langle 5\rangle 7$
$\langle 4\rangle$ QED BY $\langle 4\rangle 2$, $\langle 4\rangle 3$
$\langle 3\rangle 8.$ CASE $i + 1 = n$
$\langle 4\rangle 1.$ $vbi1 = AddD(vai1)$ BY $\langle 3\rangle 2$, $\langle 3\rangle 8$, $\langle 1\rangle 15$ DEF $AddD$
$\langle 4\rangle 2.$ $i < n$ BY $\langle 3\rangle 8$, $SMTT(10)$
$\langle 4\rangle 3.$ $fbi = fai$ BY $\langle 4\rangle 2$, $\langle 3\rangle 1$
$\langle 4\rangle 4.$ $\neg(i + 1 < n)$ BY $\langle 3\rangle 8$, $SMTT(10)$
$\langle 4\rangle$ SUFFICES $fbi1 = AddD(fai1)$ BY $\langle 4\rangle 4$
$\langle 4\rangle$ HIDE DEF fai, fbi, $fai1$, $fbi1$, $vai1$, $vbi1$
$\langle 4\rangle 5.$ $fbi1 = DeltaVecAdd(fai, DeltaVecAdd(vai1, d))$ BY $\langle 3\rangle 4$, $\langle 4\rangle 3$, $\langle 4\rangle 1$ DEF $AddD$
$\langle 4\rangle 6.$ $fbi1 = DeltaVecAdd(DeltaVecAdd(fai, vai1), d)$ BY $\langle 4\rangle 5$, $\langle 3\rangle 5$, $\langle 3\rangle 6$, $DeltaVecAddAssociative$
$\langle 4\rangle 7.$ $fbi1 = AddD(fai1)$ BY $\langle 4\rangle 6$, $\langle 3\rangle 3$ DEF $AddD$
$\langle 4\rangle$ QED BY $\langle 4\rangle 7$
$\langle 3\rangle$ QED BY $\langle 3\rangle 7$, $\langle 3\rangle 8$
$\langle 2\rangle$ HIDE DEF P
$\langle 2\rangle$ QED BY ONLY $\langle 2\rangle 1$, $\langle 2\rangle 2$, $NatInduction$, Isa
$\langle 1\rangle$ HIDE DEF fa, fb
$\langle 1\rangle$ QED BY $\langle 1\rangle 4$, $\langle 1\rangle 9$, $\langle 1\rangle 18$

Facts about summing up sequences of delta vectors. Corollaries for the special case of k = 0.

Let $Prop$ be any predicate satisfied by $Zero$ and preserved by Add. Let Q be a sequence of delta vectors in which each element satisfies $Prop$. Then the sum of Q is a delta vector that satisfies $Prop$.

We define explicit operators to capture the hypothesis and conclusion. Otherwise the provers seem to have difficulty figuring out how to apply this theorem.

$DeltaVecSeqSumProp\_Hypothesis(Prop(\_),\ Q)\ \triangleq$
$\quad \land\ Prop(DeltaVecZero)$
$\quad \land\ \forall\, a,\ b \in DeltaVecType : Prop(a) \land Prop(b) \Rightarrow Prop(DeltaVecAdd(a,\ b))$
$\quad \land\ Q \in Seq(DeltaVecType)$
$\quad \land\ \forall\, i \in 1\ ..\ Len(Q) : Prop(Q[i])$

$DeltaVecSeqSumProp\_Conclusion(Prop(\_),\ Q)\ \triangleq$
$\quad \land\ DeltaVecSeqSum(Q) \in DeltaVecType$
$\quad \land\ Prop(DeltaVecSeqSum(Q))$

THEOREM $DeltaVecSeqSumProp\ \triangleq$
  ASSUME NEW $Prop(\_)$, NEW Q, $DeltaVecSeqSumProp\_Hypothesis(Prop,\ Q)$
  PROVE $DeltaVecSeqSumProp\_Conclusion(Prop,\ Q)$
PROOF
  $\langle 1\rangle$ $DeltaVecSeqSkipSum(0,\ Q) \in DeltaVecType \land Prop(DeltaVecSeqSkipSum(0,\ Q))$
    $\langle 2\rangle$ $DeltaVecSeqSkipSumProp\_Conclusion(Prop,\ Q,\ 0)$
      $\langle 3\rangle$ $DeltaVecSeqSkipSumProp\_Hypothesis(Prop,\ Q,\ 0)$
        $\langle 4\rangle$ USE DEF $DeltaVecSeqSumProp\_Hypothesis$
        $\langle 4\rangle$ $\forall\, i \in Nat : 0 < i \land i \leq Len(Q) \Rightarrow Prop(Q[i])$
          $\langle 5\rangle$ DEFINE $LenQ\ \triangleq\ Len(Q)$
          $\langle 5\rangle$ HIDE DEF $LenQ$
          $\langle 5\rangle$ $\forall\, i \in 1\ ..\ LenQ : Prop(Q[i])$ BY DEF $LenQ$
          $\langle 5\rangle$ $LenQ \in Nat$ BY $LenInNat$ DEF $LenQ$
          $\langle 5\rangle$ $\forall\, i \in Nat : 0 < i \land i \leq LenQ \Rightarrow i \in 1\ ..\ LenQ$ BY $SMTT(10)$
          $\langle 5\rangle$ QED BY DEF $LenQ$
        $\langle 4\rangle$ QED BY DEF $DeltaVecSeqSkipSumProp\_Hypothesis$
      $\langle 3\rangle$ QED BY $DeltaVecSeqSkipSumProp$
    $\langle 2\rangle$ QED BY DEF $DeltaVecSeqSkipSumProp\_Conclusion$
  $\langle 1\rangle$ QED BY DEF $DeltaVecSeqSum$, $DeltaVecSeqSumProp\_Conclusion$

The sum of a sequence of delta vectors is a delta vector.

COROLLARY $DeltaVecSeqSumType \triangleq$
ASSUME
NEW $Q \in Seq(DeltaVecType)$
PROVE
$DeltaVecSeqSum(Q) \in DeltaVecType$
PROOF
$\langle 1 \rangle$ QED BY $DeltaVecSeqSkipSumType$ DEF $DeltaVecSeqSum$

The sum of a sequence of zero delta vectors is zero.

COROLLARY $DeltaVecSeqSumAllZero \triangleq$
ASSUME
NEW $Q \in Seq(DeltaVecType)$,
$\forall i \in$ DOMAIN $Q : Q[i] = DeltaVecZero$
PROVE
$DeltaVecSeqSum(Q) = DeltaVecZero$
PROOF
$\langle 1 \rangle$ QED BY $DeltaVecSeqSkipSumAllZero$ DEF $DeltaVecSeqSum$

The sum of an empty sequence of delta vectors is zero.

COROLLARY $DeltaVecSeqSumEmpty \triangleq$
ASSUME
NEW $Q \in Seq(DeltaVecType)$, $Q = \langle \rangle$
PROVE
$DeltaVecSeqSum(Q) = DeltaVecZero$
PROOF
$\langle 1 \rangle$ QED BY $DeltaVecSeqSkipSumEmpty$ DEF $DeltaVecSeqSum$

When you append a delta vector d to a sequence of delta vectors, the sum increases by d.

COROLLARY $DeltaVecSeqSumAppend \triangleq$
  ASSUME
    NEW $Q \in Seq(DeltaVecType)$,
    NEW $d \in DeltaVecType$
  PROVE
  $DeltaVecSeqSum(Append(Q, d)) = DeltaVecAdd(DeltaVecSeqSum(Q), d)$
PROOF
  $\langle 1 \rangle 1.\ 0 \leq Len(Q)$
    $\langle 2 \rangle$ DEFINE $LenQ \triangleq Len(Q)$
    $\langle 2 \rangle$ $LenQ \in Nat$ BY $LenInNat$
    $\langle 2 \rangle$ SUFFICES $0 \leq LenQ$ OBVIOUS
    $\langle 2 \rangle$ HIDE DEF $LenQ$
    $\langle 2 \rangle$ QED BY $SMTT(10)$
  $\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$, $DeltaVecSeqSkipSumAppend$, Isa DEF $DeltaVecSeqSum$

For a sequence Q of values and an index $n \in 1 \,..\, Len(Q)$, $Q[n]$ plus the sum of all values on $RemoveAt(Q, n)$ is the same as the sum of all values on Q.

COROLLARY $DeltaVecSeqSumRemoveAt \triangleq$
  ASSUME
    NEW $Q \in Seq(DeltaVecType)$,
    NEW $n \in 1 \,..\, Len(Q)$
  PROVE
  $DeltaVecAdd(Q[n], DeltaVecSeqSum(RemoveAt(Q, n))) = DeltaVecSeqSum(Q)$
PROOF
  $\langle 1 \rangle$ QED BY $DeltaVecSeqSkipSumRemoveAt$ DEF $DeltaVecSeqSum$

Adding a value d to the sum of a sequence of delta vectors gives the same result as adding d to one of the elements of the sequence and then taking the sum.

COROLLARY $DeltaVecSeqSumAddAt \triangleq$
  ASSUME
    NEW $Q \in Seq(DeltaVecType)$,
    NEW $n \in 1 \,..\, Len(Q)$,
    NEW $d \in DeltaVecType$
  PROVE

$DeltaVecAdd(DeltaVecSeqSum(Q),\ d) = DeltaVecSeqSum(DeltaVecSeqAddAt(Q,\ n,\ d))$

PROOF

$\langle 1 \rangle$ QED BY $DeltaVecSeqSkipSumAddAt$ DEF $DeltaVecSeqSum$

C.10 Facts about summing up delta vectors in the range of a function

MODULE $NaiadClockProofDeltaVecFuns$

EXTENDS $NaiadClockProofDeltaVecSeqs$

Facts about summing up delta vectors in the range of a function.

This really ought to be a library of theorems.

Let $Prop$ be any predicate satisfied by $Zero$ and preserved by Add. Let F be a function to delta vectors in which $F[d]$ satisfies $Prop$ for each $d \in$ DOMAIN F. Then any index sum of F satisfies $Prop$.

We define explicit operators to capture the hypothesis and conclusion. Otherwise the provers seem to have difficulty figuring out how to apply this theorem.

$DeltaVecFunIndexSumProp\_Hypothesis(Prop(\_), F, I) \triangleq$
$\quad \land Prop(DeltaVecZero)$
$\quad \land \forall a, b \in DeltaVecType : Prop(a) \land Prop(b) \Rightarrow Prop(DeltaVecAdd(a, b))$
$\quad \land F \in [\text{DOMAIN } F \rightarrow DeltaVecType]$
$\quad \land \forall s \in \text{DOMAIN } F : Prop(F[s])$
$\quad \land I \in Seq(\text{DOMAIN } F)$

$DeltaVecFunIndexSumProp\_Conclusion(Prop(\_), F, I) \triangleq$
$\quad \land DeltaVecFunIndexSum(F, I) \in DeltaVecType$
$\quad \land Prop(DeltaVecFunIndexSum(F, I))$

THEOREM $DeltaVecFunIndexSumProp \triangleq$
ASSUME NEW $Prop(\_)$, NEW F, NEW I, $DeltaVecFunIndexSumProp\_Hypothesis(Prop, F, I)$
PROVE $DeltaVecFunIndexSumProp\_Conclusion(Prop, F, I)$
PROOF
$\langle 1 \rangle$ $I \in Seq(\text{DOMAIN } F)$ BY DEF $DeltaVecFunIndexSumProp\_Hypothesis$
$\langle 1 \rangle$ DEFINE $LenI \triangleq Len(I)$
$\langle 1 \rangle$ DEFINE $Q \triangleq [i \in 1 \,..\, LenI \mapsto F[I[i]]]$
$\langle 1 \rangle$ HIDE DEF $LenI$, Q
$\langle 1 \rangle$ $DeltaVecSeqSum(Q) \in DeltaVecType \land Prop(DeltaVecSeqSum(Q))$
$\quad \langle 2 \rangle$ $DeltaVecSeqSumProp\_Conclusion(Prop, Q)$
$\quad\quad \langle 3 \rangle$ $DeltaVecSeqSumProp\_Hypothesis(Prop, Q)$
$\quad\quad\quad \langle 4 \rangle$ $LenI \in Nat$ BY $LenInNat$ DEF $LenI$
$\quad\quad\quad \langle 4 \rangle$ $Q \in [1 \,..\, LenI \rightarrow DeltaVecType]$
$\quad\quad\quad\quad \langle 5 \rangle$ $I \in [1 \,..\, LenI \rightarrow \text{DOMAIN } F]$ BY $LenAxiom$ DEF $LenI$
$\quad\quad\quad\quad \langle 5 \rangle$ $F \in [\text{DOMAIN } F \rightarrow DeltaVecType]$ BY DEF $DeltaVecFunIndexSumProp\_Hypothesis$

$\langle 5 \rangle$ QED BY DEF Q
$\langle 4 \rangle$ $Q \in Seq(DeltaVecType)$ BY $IsASeq$
$\langle 4 \rangle$ $LenI = Len(Q)$
$\langle 5 \rangle$ $Len(Q) \in Nat$ BY $LenInNat$
$\langle 5 \rangle$ DOMAIN $Q = 1 \,..\, Len(Q)$ BY $LenDef$
$\langle 5 \rangle$ DOMAIN $Q = 1 \,..\, LenI$ OBVIOUS
$\langle 5 \rangle$ QED BY $DotDotOneThruN$
$\langle 4 \rangle$ $\forall\, q \in 1 \,..\, LenI : Prop(Q[q])$
$\langle 5 \rangle$ $\forall\, q \in 1 \,..\, LenI : Q[q] = F[I[q]]$ BY DEF Q
$\langle 5 \rangle$ $I \in Seq($DOMAIN $F)$ BY DEF $DeltaVecFunIndexSumProp\_Hypothesis$
$\langle 5 \rangle$ $\forall\, q \in 1 \,..\, LenI : I[q] \in$ DOMAIN F BY $ElementOfSeq$ DEF $LenI$
$\langle 5 \rangle$ $\forall\, s \in$ DOMAIN $F : Prop(F[s])$ BY DEF $DeltaVecFunIndexSumProp\_Hypothesis$
$\langle 5 \rangle$ QED OBVIOUS
$\langle 4 \rangle$ QED BY DEF $DeltaVecSeqSumProp\_Hypothesis$, $DeltaVecFunIndexSumProp\_Hypothesis$
$\langle 3 \rangle$ QED BY $DeltaVecSeqSumProp$
$\langle 2 \rangle$ QED BY DEF $DeltaVecSeqSumProp\_Conclusion$
$\langle 1 \rangle$ $DeltaVecSeqSum(Q) \in DeltaVecType$ BY $DeltaVecSeqSumType$
$\langle 1 \rangle$ $DeltaVecFunIndexSum(F, I) = DeltaVecSeqSum(Q)$ BY DEF $DeltaVecFunIndexSum$, $LenI$, Q
$\langle 1 \rangle$ QED BY DEF $DeltaVecFunIndexSumProp\_Conclusion$

Let $Prop$ be any predicate satisfied by $Zero$ and preserved by Add. Let F be a function to delta vectors in which $F[d]$ satisfies $Prop$ for each $d \in$ DOMAIN F. Then any finite subset sum of F satisfies $Prop$.

We define explicit operators to capture the hypothesis and conclusion. Otherwise the provers seem to have difficulty figuring out how to apply this theorem.

$DeltaVecFunSubsetSumProp\_Hypothesis(Prop(\_), F, S) \triangleq$
$\land\ Prop(DeltaVecZero)$
$\land\ \forall\, a, b \in DeltaVecType : Prop(a) \land Prop(b) \Rightarrow Prop(DeltaVecAdd(a, b))$
$\land\ F \in [$DOMAIN $F \rightarrow DeltaVecType]$
$\land\ \forall\, s \in$ DOMAIN $F : Prop(F[s])$
$\land\ S \subseteq$ DOMAIN F
$\land\ IsFiniteSet(S)$

$DeltaVecFunSubsetSumProp\_Conclusion(Prop(\_), F, S) \triangleq$
$\land\ DeltaVecFunSubsetSum(F, S) \in DeltaVecType$
$\land\ Prop(DeltaVecFunSubsetSum(F, S))$

THEOREM $DeltaVecFunSubsetSumProp \triangleq$
ASSUME NEW $Prop(\_)$, NEW F, NEW S, $DeltaVecFunSubsetSumProp\_Hypothesis(Prop, F, S)$
PROVE $DeltaVecFunSubsetSumProp\_Conclusion(Prop, F, S)$
PROOF
$\langle 1 \rangle$ DEFINE $I \triangleq ExactSeqFor(S)$

$\langle 1 \rangle$ HIDE DEF I
$\langle 1 \rangle$ $DeltaVecFunIndexSumProp\_Conclusion(Prop, F, I)$
  $\langle 2 \rangle$ $DeltaVecFunIndexSumProp\_Hypothesis(Prop, F, I)$
    $\langle 3 \rangle$ USE DEF $DeltaVecFunSubsetSumProp\_Hypothesis$
    $\langle 3 \rangle$ $I \in Seq(\text{DOMAIN } F)$
      $\langle 4 \rangle$ $I \in Seq(S)$
        $\langle 5 \rangle$ $IsExactSeqFor(I, S)$ BY $ExactSeqForProperties$ DEF I
        $\langle 5 \rangle$ QED BY DEF $IsExactSeqFor$
      $\langle 4 \rangle$ QED BY $SeqSupset$
    $\langle 3 \rangle$ QED BY DEF $DeltaVecFunIndexSumProp\_Hypothesis$
  $\langle 2 \rangle$ QED BY $DeltaVecFunIndexSumProp$
$\langle 1 \rangle$ QED
  $\langle 2 \rangle$ USE DEF $DeltaVecFunSubsetSumProp\_Conclusion$
  $\langle 2 \rangle$ USE DEF $DeltaVecFunIndexSumProp\_Conclusion$
  $\langle 2 \rangle$ USE DEF $DeltaVecFunSubsetSum$
  $\langle 2 \rangle$ USE DEF I
  $\langle 2 \rangle$ QED OBVIOUS

Let $Prop$ be any predicate satisfied by $Zero$ and preserved by Add. Let F be a function to delta vectors with finite non-zero range in which $F[d]$ satisfies $Prop$ for each $d \in \text{DOMAIN } F$. Then the sum of F satisfies $Prop$.

We define explicit operators to capture the hypothesis and conclusion. Otherwise the provers seem to have difficulty figuring out how to apply this theorem.

$DeltaVecFunSumProp\_Hypothesis(Prop(\_), F) \triangleq$
  $\land\ Prop(DeltaVecZero)$
  $\land\ \forall a, b \in DeltaVecType : Prop(a) \land Prop(b) \Rightarrow Prop(DeltaVecAdd(a, b))$
  $\land\ F \in [\text{DOMAIN } F \rightarrow DeltaVecType]$
  $\land\ \forall s \in \text{DOMAIN } F : Prop(F[s])$
  $\land\ DeltaVecFunHasFiniteNonZeroRange(F)$

$DeltaVecFunSumProp\_Conclusion(Prop(\_), F) \triangleq$
  $\land\ DeltaVecFunSum(F) \in DeltaVecType$
  $\land\ Prop(DeltaVecFunSum(F))$

THEOREM $DeltaVecFunSumProp \triangleq$
  ASSUME NEW $Prop(\_)$, NEW F, $DeltaVecFunSumProp\_Hypothesis(Prop, F)$
  PROVE $DeltaVecFunSumProp\_Conclusion(Prop, F)$
PROOF
  $\langle 1 \rangle$ DEFINE $S \triangleq \{s \in \text{DOMAIN } F : F[s] \neq DeltaVecZero\}$
  $\langle 1 \rangle$ HIDE DEF S

$\langle 1 \rangle$ $DeltaVecFunSubsetSumProp\_Conclusion(Prop,\ F,\ S)$
$\langle 2 \rangle$ $DeltaVecFunSubsetSumProp\_Hypothesis(Prop,\ F,\ S)$
$\langle 3 \rangle$ USE DEF $DeltaVecFunSumProp\_Hypothesis$
$\langle 3 \rangle$ $S \subseteq$ DOMAIN F BY DEF S
$\langle 3 \rangle$ $IsFiniteSet(S)$ BY DEF $DeltaVecFunHasFiniteNonZeroRange,\ S$
$\langle 3 \rangle$ QED BY DEF $DeltaVecFunSubsetSumProp\_Hypothesis$
$\langle 2 \rangle$ QED BY $DeltaVecFunSubsetSumProp$
$\langle 1 \rangle$ QED
$\langle 2 \rangle$ USE DEF $DeltaVecFunSumProp\_Conclusion$
$\langle 2 \rangle$ USE DEF $DeltaVecFunSubsetSumProp\_Conclusion$
$\langle 2 \rangle$ USE DEF $DeltaVecFunSum$
$\langle 2 \rangle$ USE DEF S
$\langle 2 \rangle$ QED OBVIOUS

The index sum is a delta vector.

THEOREM $DeltaVecFunIndexSumType \triangleq$
ASSUME
NEW D,
NEW $F \in [D \rightarrow DeltaVecType]$,
NEW $I \in Seq(D)$
PROVE
$DeltaVecFunIndexSum(F,\ I) \in DeltaVecType$
PROOF
$\langle 1 \rangle$ DEFINE $Prop(a) \triangleq$ TRUE
$\langle 1 \rangle$ $DeltaVecFunIndexSumProp\_Hypothesis(Prop,\ F,\ I)$ BY DEF $DeltaVecFunIndexSumProp\_Hypothesis$
$\langle 1 \rangle$ $DeltaVecFunIndexSumProp\_Conclusion(Prop,\ F,\ I)$ BY $DeltaVecFunIndexSumProp$
$\langle 1 \rangle$ QED BY DEF $DeltaVecFunIndexSumProp\_Conclusion$

The subset sum is a delta vector.

THEOREM $DeltaVecFunSubsetSumType \triangleq$
ASSUME
NEW D,
NEW $F \in [D \rightarrow DeltaVecType]$,
NEW $S,\ IsFiniteSet(S),\ S \subseteq D$
PROVE

$DeltaVecFunSubsetSum(F, S) \in DeltaVecType$
PROOF
$\langle 1 \rangle$ DEFINE $Prop(a) \triangleq$ TRUE
$\langle 1 \rangle$ $DeltaVecFunSubsetSumProp\_Hypothesis(Prop, F, S)$ BY DEF $DeltaVecFunSubsetSumProp\_Hypothesis$
$\langle 1 \rangle$ $DeltaVecFunSubsetSumProp\_Conclusion(Prop, F, S)$ BY $DeltaVecFunSubsetSumProp$
$\langle 1 \rangle$ QED BY DEF $DeltaVecFunSubsetSumProp\_Conclusion$

The sum is a delta vector.

THEOREM $DeltaVecFunSumType \triangleq$
ASSUME
NEW D,
NEW $F \in [D \rightarrow DeltaVecType]$, $DeltaVecFunHasFiniteNonZeroRange(F)$
PROVE
$DeltaVecFunSum(F) \in DeltaVecType$
PROOF
$\langle 1 \rangle$ DEFINE $Prop(a) \triangleq$ TRUE
$\langle 1 \rangle$ $DeltaVecFunSumProp\_Hypothesis(Prop, F)$ BY DEF $DeltaVecFunSumProp\_Hypothesis$
$\langle 1 \rangle$ $DeltaVecFunSumProp\_Conclusion(Prop, F)$ BY $DeltaVecFunSumProp$
$\langle 1 \rangle$ QED BY DEF $DeltaVecFunSumProp\_Conclusion$

Removing an index from an index sum produces the expected partial sum.

THEOREM $DeltaVecFunIndexSumRemoveAt \triangleq$
ASSUME
NEW D,
NEW $F \in [D \rightarrow DeltaVecType]$,
NEW $I \in Seq(D)$,
NEW $i \in 1 \,..\, Len(I)$
PROVE
$DeltaVecFunIndexSum(F, I) = DeltaVecAdd(F[I[i]], DeltaVecFunIndexSum(F, RemoveAt(I, i)))$
PROOF
$\langle 1 \rangle$ DEFINE $LenI \triangleq Len(I)$
$\langle 1 \rangle$ DEFINE $Q \triangleq [k \in 1 \,..\, LenI \mapsto F[I[k]]]$
$\langle 1 \rangle$ DEFINE $LenQ \triangleq Len(Q)$
$\langle 1 \rangle$ HIDE DEF $LenI, Q, LenQ$
$\langle 1 \rangle 1.$ $I \in Seq(D)$ OBVIOUS

$\langle 1\rangle 2.\ i\ \in 1\ ..\ LenI$ BY DEF $LenI$
$\langle 1\rangle 3.\ LenI \in Nat$ BY $LenInNat$ DEF $LenI$
$\langle 1\rangle 4.\ I\ \in [1\ ..\ LenI \rightarrow D]$ BY $LenAxiom$ DEF $LenI$
$\langle 1\rangle 5.\ Q \in [1\ ..\ LenI \rightarrow DeltaVecType]$ BY $\langle 1\rangle 4$ DEF Q
$\langle 1\rangle 6.\ Q \in Seq(DeltaVecType)$ BY $\langle 1\rangle 3,\ \langle 1\rangle 5,\ IsASeq$
$\langle 1\rangle 7.\ LenQ = LenI$ BY $\langle 1\rangle 3,\ \langle 1\rangle 5,\ \langle 1\rangle 6,\ LenDomain$ DEF $LenQ$
$\langle 1\rangle 8.\ i \in 1\ ..\ LenQ$ BY $\langle 1\rangle 2,\ \langle 1\rangle 7$
$\langle 1\rangle 9.\ DeltaVecSeqSum(Q) = DeltaVecAdd(Q[i],\ DeltaVecSeqSum(RemoveAt(Q,\ i)))$ BY $\langle 1\rangle 6,\ \langle 1\rangle 8,\ DeltaVecSeqSumRe$
$\langle 1\rangle 10.\ DeltaVecFunIndexSum(F,\ I) = DeltaVecSeqSum(Q)$ BY DEF $DeltaVecFunIndexSum,\ Q,\ LenI$
$\langle 1\rangle 11.\ F[I[i]] = Q[i]$ BY $\langle 1\rangle 2$ DEF Q
$\langle 1\rangle 12.\ DeltaVecFunIndexSum(F,\ RemoveAt(I,\ i)) = DeltaVecSeqSum(RemoveAt(Q,\ i))$
$\langle 2\rangle$ DEFINE $I1\ \triangleq\ RemoveAt(I,\ i)$
$\langle 2\rangle$ DEFINE $Q1\ \triangleq\ RemoveAt(Q,\ i)$
$\langle 2\rangle$ DEFINE $LenI1\ \triangleq\ Len(I1)$
$\langle 2\rangle$ DEFINE $LenQ1\ \triangleq\ Len(Q1)$
$\langle 2\rangle$ HIDE DEF $I1,\ Q1,\ LenI1,\ LenQ1$
$\langle 2\rangle 1.\ I1\ \in Seq(D)$ BY $\langle 1\rangle 1,\ \langle 1\rangle 2,\ RemoveAtProperties$ DEF $I1,\ LenI$
$\langle 2\rangle 2.\ Q1 \in Seq(DeltaVecType)$ BY $\langle 1\rangle 6,\ \langle 1\rangle 8,\ RemoveAtProperties$ DEF $Q1,\ LenQ$
$\langle 2\rangle 3.\ LenI1 = LenQ1$
$\langle 3\rangle 3.\ LenI1\ = LenI\ - 1$ BY $\langle 1\rangle 1,\ \langle 1\rangle 2,\ RemoveAtProperties$ DEF $I1,\ LenI,\ LenI1$
$\langle 3\rangle 4.\ LenQ1 = LenQ - 1$ BY $\langle 1\rangle 6,\ \langle 1\rangle 8,\ RemoveAtProperties$ DEF $Q1,\ LenQ,\ LenQ1$
$\langle 3\rangle$ QED BY $\langle 1\rangle 7,\ \langle 3\rangle 3,\ \langle 3\rangle 4$
$\langle 2\rangle 4.\ DeltaVecFunIndexSum(F,\ I1) = DeltaVecSeqSum([k \in 1\ ..\ LenI1 \mapsto F[I1[k]]])$ BY DEF $DeltaVecFunIndexSum,$

$\langle 2\rangle$ SUFFICES $Q1 = [k \in 1\ ..\ LenQ1 \mapsto F[I1[k]]]$ BY $\langle 2\rangle 3,\ \langle 2\rangle 4$ DEF $I1,\ Q1$
$\langle 2\rangle 5.\ Q1 \in [1\ ..\ LenQ1 \rightarrow DeltaVecType]$ BY $\langle 2\rangle 2,\ LenAxiom$ DEF $LenQ1$

$\langle 2\rangle 6.$ SUFFICES ASSUME NEW $k,\ k \in 1\ ..\ LenQ1$ PROVE $Q1[k] = F[I1[k]]$ BY $\langle 2\rangle 5$
$\langle 2\rangle 7.\ k \in 1\ ..\ LenI1$ BY $\langle 2\rangle 3,\ \langle 2\rangle 6$
$\langle 2\rangle 8.\ k \in 1\ ..\ LenQ1$ OBVIOUS
$\langle 2\rangle 9.\ RemoveAt\_MapBackward(I,\ i)$ BY $\langle 1\rangle 1,\ \langle 1\rangle 2,\ RemoveAtProperties$ DEF $I1,\ LenI$
$\langle 2\rangle 10.\ RemoveAt\_MapBackward(Q,\ i)$ BY $\langle 1\rangle 6,\ \langle 1\rangle 8,\ RemoveAtProperties$ DEF $Q1,\ LenQ$
$\langle 2\rangle$ DEFINE $Iik\ \triangleq\ RemoveAt\_BackwardIndex(I,\ i,\ k)$
$\langle 2\rangle$ DEFINE $Qik\ \triangleq\ RemoveAt\_BackwardIndex(Q,\ i,\ k)$
$\langle 2\rangle 11.\ Iik\ \in 1\ ..\ LenI\ \wedge I1[k]\ = I[Iik]$ BY $\langle 2\rangle 7,\ \langle 2\rangle 9$ DEF $RemoveAt\_MapBackward,\ I1,\ LenI,\ LenI1$
$\langle 2\rangle 12.\ Qik \in 1\ ..\ LenQ \wedge Q1[k] = Q[Qik]$ BY $\langle 2\rangle 8,\ \langle 2\rangle 10$ DEF $RemoveAt\_MapBackward,\ Q1,\ LenQ,\ LenQ1$
$\langle 2\rangle 13.\ Iik\ = Qik$ BY DEF $RemoveAt\_BackwardIndex$
$\langle 2\rangle$ QED BY $\langle 1\rangle 7,\ \langle 2\rangle 11,\ \langle 2\rangle 12,\ \langle 2\rangle 13$ DEF Q
$\langle 1\rangle$ QED BY $\langle 1\rangle 9,\ \langle 1\rangle 10,\ \langle 1\rangle 11,\ \langle 1\rangle 12$

All exact sequences for a given subset produce the same index sum.

We define explicit operators to capture the hypothesis and conclusion. Otherwise the provers seem to have difficulty figuring out how to apply this theorem.

$DeltaVecFunIndexSumAnyExactSeq\_Hypothesis(F, S, I, J) \triangleq$
LET
$D \triangleq$ DOMAIN F
IN
$\land F \in [D \to DeltaVecType]$
$\land S \subseteq D$
$\land IsExactSeqFor(I, S)$
$\land IsExactSeqFor(J, S)$

$DeltaVecFunIndexSumAnyExactSeq\_Conclusion(F, S, I, J) \triangleq$
$DeltaVecFunIndexSum(F, I) = DeltaVecFunIndexSum(F, J)$

THEOREM $DeltaVecFunIndexSumAnyExactSeq \triangleq$
ASSUME NEW F, NEW S, NEW I, NEW J, $DeltaVecFunIndexSumAnyExactSeq\_Hypothesis(F, S, I, J)$
PROVE $DeltaVecFunIndexSumAnyExactSeq\_Conclusion(F, S, I, J)$
PROOF
$\langle 1 \rangle$ DEFINE $D \triangleq$ DOMAIN F
$\langle 1 \rangle$ $F \in [D \to DeltaVecType]$ BY DEF $DeltaVecFunIndexSumAnyExactSeq\_Hypothesis$
$\langle 1 \rangle$ $S \subseteq D$ BY DEF $DeltaVecFunIndexSumAnyExactSeq\_Hypothesis$
$\langle 1 \rangle$ $IsExactSeqFor(I, S)$ BY DEF $DeltaVecFunIndexSumAnyExactSeq\_Hypothesis$
$\langle 1 \rangle$ $IsExactSeqFor(J, S)$ BY DEF $DeltaVecFunIndexSumAnyExactSeq\_Hypothesis$
$\langle 1 \rangle$ HIDE DEF D
$\langle 1 \rangle$ USE DEF $DeltaVecFunIndexSumAnyExactSeq\_Conclusion$

A counterexample to this theorem is a subset $S1$ with exact sequences $I1$ and $J1$ that produce different index sums.

$\langle 1 \rangle$ DEFINE $IsCounterexample(S1, I1, J1) \triangleq$
$\land S1 \subseteq D$
$\land IsExactSeqFor(I1, S1)$
$\land IsExactSeqFor(J1, S1)$
$\land DeltaVecFunIndexSum(F, I1) \neq DeltaVecFunIndexSum(F, J1)$
$\langle 1 \rangle$ HIDE DEF $IsCounterexample$

Let N be the set of all natural numbers n such that there is a counterexample and the length of one of the exact sequences is n.

$\langle 1 \rangle$ DEFINE $N \triangleq \{n \in Nat : \exists S1, I1, J1 : IsCounterexample(S1, I1, J1) \land n = Len(I1)\}$
$\langle 1 \rangle$ HIDE DEF N
$\langle 1 \rangle 1.$ SUFFICES $N = \{\}$
$\langle 2 \rangle 1.$ SUFFICES ASSUME $DeltaVecFunIndexSum(F, I) \neq DeltaVecFunIndexSum(F, J)$ PROVE FALSE OBVIOUS
$\langle 2 \rangle 2.$ $IsCounterexample(S, I, J)$ BY $\langle 2 \rangle 1$ DEF $IsCounterexample$
$\langle 2 \rangle 3.$ $I \in Seq(S)$ BY DEF $IsExactSeqFor$
$\langle 2 \rangle 4.$ $Len(I) \in Nat$ BY $\langle 2 \rangle 3$, $LenInNat$
$\langle 2 \rangle 5.$ $Len(I) \in N$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 4$ DEF N
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 2 \rangle 5$
$\langle 1 \rangle 2.$ SUFFICES ASSUME $N \neq \{\}$ PROVE FALSE OBVIOUS

If there is a counterexample, there must be a smallest one.

$\langle 1 \rangle 3.$ PICK $n \in N : \forall m \in N : n \leq m$ BY $\langle 1 \rangle 2$, $NatWellFounded$ DEF N
$\langle 1 \rangle 4.$ PICK $S1,\ I1,\ J1 : IsCounterexample(S1,\ I1,\ J1) \land n = Len(I1)$ BY $\langle 1 \rangle 3$ DEF N
$\langle 1 \rangle$ DEFINE $LenI1 \triangleq Len(I1)$
$\langle 1 \rangle$ DEFINE $LenJ1 \triangleq Len(J1)$
$\langle 1 \rangle$ HIDE DEF $LenI1,\ LenJ1$

Based on this "smallest" counterexample, we will construct a smaller one, thus establishing a contradiction.

First we establish various useful facts about $S1$, $I1$, and $J1$.

$\langle 1 \rangle 5.$ $S1 \subseteq D$ BY $\langle 1 \rangle 4$ DEF $IsCounterexample$
$\langle 1 \rangle 6.$ $IsExactSeqFor(I1,\ S1)$ BY $\langle 1 \rangle 4$ DEF $IsCounterexample$
$\langle 1 \rangle 7.$ $IsExactSeqFor(J1,\ S1)$ BY $\langle 1 \rangle 4$ DEF $IsCounterexample$

$\langle 1 \rangle 8.$ $I1 \in Seq(S1)$ BY $\langle 1 \rangle 6$ DEF $IsExactSeqFor$
$\langle 1 \rangle 9.$ $J1 \in Seq(S1)$ BY $\langle 1 \rangle 7$ DEF $IsExactSeqFor$

$\langle 1 \rangle 10.$ $ExactSeq\_Each(I1,\ S1)$ BY $\langle 1 \rangle 6$ DEF $IsExactSeqFor$
$\langle 1 \rangle 11.$ $ExactSeq\_Each(J1,\ S1)$ BY $\langle 1 \rangle 7$ DEF $IsExactSeqFor$

$\langle 1 \rangle 12.$ $LenI1 \in Nat$ BY $\langle 1 \rangle 8$, $LenInNat$ DEF $LenI1$
$\langle 1 \rangle 13.$ $LenJ1 \in Nat$ BY $\langle 1 \rangle 9$, $LenInNat$ DEF $LenJ1$

$\langle 1 \rangle 14.$ $I1 \in Seq(D)$ BY $\langle 1 \rangle 5$, $\langle 1 \rangle 8$, $SeqSupset$
$\langle 1 \rangle 15.$ $J1 \in Seq(D)$ BY $\langle 1 \rangle 5$, $\langle 1 \rangle 9$, $SeqSupset$

$\langle 1 \rangle 16.$ $DeltaVecFunIndexSum(F,\ I1) \neq DeltaVecFunIndexSum(F,\ J1)$ BY $\langle 1 \rangle 4$ DEF $IsCounterexample$
$\langle 1 \rangle 17.$ $n = LenI1$ BY $\langle 1 \rangle 4$ DEF $LenI1$

$\langle 1 \rangle 18.$ $S1 \neq \{\}$
  $\langle 2 \rangle 1.$ SUFFICES ASSUME $S1 = \{\}$ PROVE FALSE OBVIOUS
  $\langle 2 \rangle 2.$ $I1 = \langle\rangle$ BY $\langle 1 \rangle 6$, $\langle 2 \rangle 1$, $ExactSeqEmpty$
  $\langle 2 \rangle 3.$ $J1 = \langle\rangle$ BY $\langle 1 \rangle 7$, $\langle 2 \rangle 1$, $ExactSeqEmpty$
  $\langle 2 \rangle 4.$ $I1 = J1$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$
  $\langle 2 \rangle 5.$ $DeltaVecFunIndexSum(F,\ I1) = DeltaVecFunIndexSum(F,\ J1)$ BY $\langle 2 \rangle 4$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 16$, $\langle 2 \rangle 5$

Since $S1 \neq \{\}$, we pick some element $s1 \in S1$ and remove it from $S1$, $I1$, and $J1$. This creates a smaller counterexample.

$\langle 1 \rangle 19.$ PICK $s1 : s1 \in S1$ BY $\langle 1 \rangle 18$
$\langle 1 \rangle 20.$ PICK $i1 : i1 \in 1 \,..\, LenI1 \land I1[i1] = s1$ BY $\langle 1 \rangle 10$, $\langle 1 \rangle 19$ DEF $ExactSeq\_Each$, $LenI1$
$\langle 1 \rangle 21.$ PICK $j1 : j1 \in 1 \,..\, LenJ1 \land J1[j1] = s1$ BY $\langle 1 \rangle 11$, $\langle 1 \rangle 19$ DEF $ExactSeq\_Each$, $LenJ1$

$\langle 1 \rangle$ DEFINE $S2 \triangleq S1 \setminus \{s1\}$
$\langle 1 \rangle$ DEFINE $I2 \triangleq RemoveAt(I1,\ i1)$
$\langle 1 \rangle$ DEFINE $J2 \triangleq RemoveAt(J1,\ j1)$
$\langle 1 \rangle$ DEFINE $LenI2 \triangleq Len(I2)$
$\langle 1 \rangle$ DEFINE $LenJ2 \triangleq Len(J2)$
$\langle 1 \rangle$ HIDE DEF $S2,\ I2,\ J2,\ LenI2,\ LenJ2$

$\langle 1 \rangle 22.$ $LenI2 = LenI1 - 1$ BY $\langle 1 \rangle 8$, $\langle 1 \rangle 20$, $RemoveAtProperties$ DEF $I2$, $LenI2$, $LenI1$

$\langle 1 \rangle 23.$ $LenJ2 = LenJ1 - 1$ BY $\langle 1 \rangle 9$, $\langle 1 \rangle 21$, $RemoveAtProperties$ DEF $J2$, $LenJ2$, $LenJ1$

$\langle 1 \rangle 24.$ $S2 \subseteq D$ BY $\langle 1 \rangle 5$ DEF $S2$
$\langle 1 \rangle 25.$ $IsExactSeqFor(I2, S2)$ BY $\langle 1 \rangle 6$, $\langle 1 \rangle 20$, $ExactSeqRemoveAt$ DEF $I2$, $S2$, $LenI1$
$\langle 1 \rangle 26.$ $IsExactSeqFor(J2, S2)$ BY $\langle 1 \rangle 7$, $\langle 1 \rangle 21$, $ExactSeqRemoveAt$ DEF $J2$, $S2$, $LenJ1$

$\langle 1 \rangle 27.$ $I2 \in Seq(S2)$ BY $\langle 1 \rangle 25$ DEF $IsExactSeqFor$
$\langle 1 \rangle 28.$ $J2 \in Seq(S2)$ BY $\langle 1 \rangle 26$ DEF $IsExactSeqFor$

$\langle 1 \rangle 29.$ $LenI2 \in Nat$ BY $\langle 1 \rangle 27$, $LenInNat$ DEF $LenI2$

$\langle 1 \rangle 30.$ $DeltaVecFunIndexSum(F, I1) = DeltaVecAdd(F[s1], DeltaVecFunIndexSum(F, I2))$
BY $\langle 1 \rangle 14$, $\langle 1 \rangle 20$, $DeltaVecFunIndexSumRemoveAt$ DEF $I2$, $LenI1$

$\langle 1 \rangle 31.$ $DeltaVecFunIndexSum(F, J1) = DeltaVecAdd(F[s1], DeltaVecFunIndexSum(F, J2))$
BY $\langle 1 \rangle 15$, $\langle 1 \rangle 21$, $DeltaVecFunIndexSumRemoveAt$ DEF $J2$, $LenJ1$

$\langle 1 \rangle 32.$ $DeltaVecFunIndexSum(F, I2) \neq DeltaVecFunIndexSum(F, J2)$
$\langle 2 \rangle 1.$ SUFFICES ASSUME $DeltaVecFunIndexSum(F, I2) = DeltaVecFunIndexSum(F, J2)$ PROVE FALSE OBVIOUS
$\langle 2 \rangle 2.$ $DeltaVecFunIndexSum(F, I1) = DeltaVecFunIndexSum(F, J1)$ BY $\langle 1 \rangle 30$, $\langle 1 \rangle 31$, $\langle 2 \rangle 1$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 16$, $\langle 2 \rangle 2$

$\langle 1 \rangle 33.$ $IsCounterexample(S2, I2, J2)$ BY $\langle 1 \rangle 24$, $\langle 1 \rangle 25$, $\langle 1 \rangle 26$, $\langle 1 \rangle 32$ DEF $IsCounterexample$

$\langle 1 \rangle 34.$ $LenI2 \in N$ BY $\langle 1 \rangle 29$, $\langle 1 \rangle 33$ DEF $LenI2$, N
$\langle 1 \rangle 35.$ $\neg(LenI1 \leq LenI2)$ BY $\langle 1 \rangle 12$, $\langle 1 \rangle 22$, $SMTT(10)$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 3$, $\langle 1 \rangle 17$, $\langle 1 \rangle 34$, $\langle 1 \rangle 35$

The index sum of an empty sequence is zero.

THEOREM $DeltaVecFunIndexSumEmpty \triangleq$
ASSUME
NEW D,
NEW $F \in [D \rightarrow DeltaVecType]$
PROVE
$DeltaVecFunIndexSum(F, \langle\rangle) = DeltaVecZero$
PROOF
$\langle 1 \rangle$ DEFINE $I \triangleq \langle\rangle$
$\langle 1 \rangle$ DEFINE $LenI \triangleq Len(I)$
$\langle 1 \rangle$ DEFINE $Q \triangleq [i \in 1 \,..\, LenI \mapsto F[I[i]]]$
$\langle 1 \rangle$ HIDE DEF I, $LenI$, Q
$\langle 1 \rangle 1.$ $DeltaVecFunIndexSum(F, I) = DeltaVecSeqSum(Q)$ BY DEF $DeltaVecFunIndexSum$, $LenI$, Q
$\langle 1 \rangle 2.$ $LenI = 0$ BY $EmptySeq$ DEF I, $LenI$

$\langle 1 \rangle 3.\ 1 \,..\, \mathit{LenI} = \{\}$ BY $\langle 1 \rangle 2,\ \mathit{SMTT}(10)$
$\langle 1 \rangle 4.\ \forall\, i \in 1 \,..\, \mathit{LenI} : I[i] \in D$ BY $\langle 1 \rangle 3$
$\langle 1 \rangle 5.\ Q \in [1 \,..\, \mathit{LenI} \rightarrow \mathit{DeltaVecType}]$ BY $\langle 1 \rangle 4$ DEF Q
$\langle 1 \rangle 6.$ DOMAIN $Q = 1 \,..\, \mathit{LenI}$ BY $\langle 1 \rangle 5$
$\langle 1 \rangle 7.\ Q \in \mathit{Seq}(\mathit{DeltaVecType})$ BY $\langle 1 \rangle 2,\ \langle 1 \rangle 5,\ \mathit{IsASeq}$
$\langle 1 \rangle 8.\ \mathit{Len}(Q) = 0$ BY $\langle 1 \rangle 2,\ \langle 1 \rangle 6,\ \langle 1 \rangle 7,\ \mathit{LenDomain}$
$\langle 1 \rangle 9.\ Q = \langle\rangle$ BY $\langle 1 \rangle 7,\ \langle 1 \rangle 8,\ \mathit{EmptySeq}$
$\langle 1 \rangle 10.\ \mathit{DeltaVecSeqSum}(Q) = \mathit{DeltaVecZero}$ BY $\langle 1 \rangle 7,\ \langle 1 \rangle 9,\ \mathit{DeltaVecSeqSumEmpty}$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1,\ \langle 1 \rangle 10$ DEF I

The subset sum of an empty subset is zero.

THEOREM $\mathit{DeltaVecFunSubsetSumEmpty} \triangleq$
ASSUME
NEW D,
NEW $F \in [D \rightarrow \mathit{DeltaVecType}]$
PROVE
$\mathit{DeltaVecFunSubsetSum}(F,\ \{\}) = \mathit{DeltaVecZero}$
PROOF
$\langle 1 \rangle 1.\ \mathit{DeltaVecFunIndexSum}(F,\ \langle\rangle) = \mathit{DeltaVecZero}$ BY $\mathit{DeltaVecFunIndexSumEmpty}$
$\langle 1 \rangle 2.\ \langle\rangle = \mathit{ExactSeqFor}(\{\})$
BY $\mathit{FiniteSetEmpty},\ \mathit{ExactSeqForProperties},\ \mathit{ExactSeqEmpty}$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1,\ \langle 1 \rangle 2$ DEF $\mathit{DeltaVecFunSubsetSum}$

The sum of an all-zero function is zero.

THEOREM $\mathit{DeltaVecFunSumAllZero} \triangleq$
ASSUME
NEW D,
NEW $F \in [D \rightarrow \mathit{DeltaVecType}]$,
$\forall\, d \in D : F[d] = \mathit{DeltaVecZero}$
PROVE
$\land\ \mathit{DeltaVecFunHasFiniteNonZeroRange}(F)$
$\land\ \mathit{DeltaVecFunSum}(F) = \mathit{DeltaVecZero}$
PROOF
$\langle 1 \rangle$ DEFINE $S \triangleq \{d \in$ DOMAIN $F : F[d] \neq \mathit{DeltaVecZero}\}$
$\langle 1 \rangle$ HIDE DEF S

$\langle 1 \rangle 1.\ \mathit{DeltaVecFunHasFiniteNonZeroRange}(F)$
  $\langle 2 \rangle\ S = \{\}$ BY DEF S
  $\langle 2 \rangle\ \mathit{IsFiniteSet}(S)$ BY $\mathit{FiniteSetEmpty}$
  $\langle 2 \rangle$ QED BY DEF $S,\ \mathit{DeltaVecFunHasFiniteNonZeroRange}$

$\langle 1 \rangle 2.\ \mathit{DeltaVecFunSum}(F) = \mathit{DeltaVecZero}$
  $\langle 2 \rangle\ S = \{\}$ BY DEF S
  $\langle 2 \rangle\ \mathit{DeltaVecFunSum}(F) = \mathit{DeltaVecFunSubsetSum}(F,\ S)$ BY DEF $\mathit{DeltaVecFunSum},\ S$
  $\langle 2 \rangle$ QED BY $\mathit{DeltaVecFunSubsetSumEmpty}$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1,\ \langle 1 \rangle 2$

Adding a new element $x \notin S$ to subset S causes the subset sum to increase by $F[x]$.

We define explicit operators to capture the hypothesis and conclusion. Otherwise the provers seem to have difficulty figuring out how to apply this theorem.

$\mathit{DeltaVecFunSubsetSumNewElem\_Hypothesis}(F,\ S,\ x) \triangleq$
  LET
    $D \triangleq$ DOMAIN F
  IN
  $\land\ F \in [D \to \mathit{DeltaVecType}]$
  $\land\ S \subseteq D$
  $\land\ \mathit{IsFiniteSet}(S)$
  $\land\ x \in D$
  $\land\ x \notin S$

$\mathit{DeltaVecFunSubsetSumNewElem\_Conclusion}(F,\ S,\ x) \triangleq$
  LET
    $D \triangleq$ DOMAIN F
    $T \triangleq S \cup \{x\}$
  IN
  $\land\ \mathit{DeltaVecAdd}(F[x],\ \mathit{DeltaVecFunSubsetSum}(F,\ S)) = \mathit{DeltaVecFunSubsetSum}(F,\ T)$
  $\land\ \mathit{DeltaVecAdd}(\mathit{DeltaVecFunSubsetSum}(F,\ S),\ F[x]) = \mathit{DeltaVecFunSubsetSum}(F,\ T)$
  $\land\ T \subseteq D$
  $\land\ \mathit{IsFiniteSet}(T)$

THEOREM $\mathit{DeltaVecFunSubsetSumNewElem} \triangleq$
  ASSUME NEW F, NEW S, NEW x, $\mathit{DeltaVecFunSubsetSumNewElem\_Hypothesis}(F,\ S,\ x)$
  PROVE $\mathit{DeltaVecFunSubsetSumNewElem\_Conclusion}(F,\ S,\ x)$
PROOF
  $\langle 1 \rangle$ DEFINE $D \triangleq$ DOMAIN F

$\langle 1 \rangle$ DEFINE $T \triangleq S \cup \{x\}$
$\langle 1 \rangle$ DEFINE $I \triangleq \mathit{ExactSeqFor}(S)$
$\langle 1 \rangle$ DEFINE $J \triangleq \mathit{ExactSeqFor}(T)$
$\langle 1 \rangle$ HIDE DEF $D,\ T,\ I,\ J$

$\langle 1 \rangle 1.$ $F \in [D \to \mathit{DeltaVecType}]$ BY DEF $\mathit{DeltaVecFunSubsetSumNewElem\_Hypothesis},\ D$

$\langle 1 \rangle 2.$ $\mathit{IsFiniteSet}(\{x\})$ BY $\mathit{FiniteSetSingleton}$

$\langle 1 \rangle 3.$ $\mathit{IsFiniteSet}(S)$ BY DEF $\mathit{DeltaVecFunSubsetSumNewElem\_Hypothesis}$
$\langle 1 \rangle 4.$ $\mathit{IsFiniteSet}(T)$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\mathit{FiniteSetUnion}$ DEF T

$\langle 1 \rangle 5.$ $x \in D$ BY DEF $\mathit{DeltaVecFunSubsetSumNewElem\_Hypothesis},\ D$
$\langle 1 \rangle 6.$ $x \notin S$ BY DEF $\mathit{DeltaVecFunSubsetSumNewElem\_Hypothesis}$
$\langle 1 \rangle 7.$ $x \in T$ BY DEF T
$\langle 1 \rangle 8.$ $T \setminus \{x\} = S$ BY $\langle 1 \rangle 6$ DEF T

$\langle 1 \rangle 9.$ $S \subseteq D$ BY DEF $\mathit{DeltaVecFunSubsetSumNewElem\_Hypothesis},\ D$
$\langle 1 \rangle 10.$ $T \subseteq D$ BY $\langle 1 \rangle 5$, $\langle 1 \rangle 9$ DEF T

$\langle 1 \rangle 11.$ $\mathit{DeltaVecFunSubsetSum}(F,\ S) = \mathit{DeltaVecFunIndexSum}(F,\ I)$
BY $\langle 1 \rangle 3$ DEF $\mathit{DeltaVecFunSubsetSum},\ I$

$\langle 1 \rangle 12.$ $\mathit{DeltaVecFunSubsetSum}(F,\ T) = \mathit{DeltaVecFunIndexSum}(F,\ J)$
BY $\langle 1 \rangle 4$ DEF $\mathit{DeltaVecFunSubsetSum},\ J$

$\langle 1 \rangle 13.$ $\mathit{IsExactSeqFor}(I,\ S)$ BY $\langle 1 \rangle 3$, $\mathit{ExactSeqForProperties}$ DEF I
$\langle 1 \rangle 14.$ $\mathit{IsExactSeqFor}(J,\ T)$ BY $\langle 1 \rangle 4$, $\mathit{ExactSeqForProperties}$ DEF J

$\langle 1 \rangle 15.$ $\mathit{ExactSeq\_Each}(J,\ T)$ BY $\langle 1 \rangle 14$ DEF $\mathit{IsExactSeqFor}$

$\langle 1 \rangle 16.$ $J \in \mathit{Seq}(T)$ BY $\langle 1 \rangle 14$ DEF $\mathit{IsExactSeqFor}$
$\langle 1 \rangle 17.$ $J \in \mathit{Seq}(D)$ BY $\langle 1 \rangle 10$, $\langle 1 \rangle 16$, $\mathit{SeqSupset}$

$\langle 1 \rangle 18.$ PICK $j : j \in 1 \,..\, \mathit{Len}(J) \land J[j] = x$ BY $\langle 1 \rangle 7$, $\langle 1 \rangle 15$ DEF $\mathit{ExactSeq\_Each}$

$\langle 1 \rangle$ DEFINE $K \triangleq \mathit{RemoveAt}(J,\ j)$
$\langle 1 \rangle$ HIDE DEF K
$\langle 1 \rangle 19.$ $\mathit{IsExactSeqFor}(K,\ S)$ BY $\langle 1 \rangle 8$, $\langle 1 \rangle 14$, $\langle 1 \rangle 18$, $\mathit{ExactSeqRemoveAt}$ DEF K
$\langle 1 \rangle 20.$ $K \in \mathit{Seq}(S)$ BY $\langle 1 \rangle 19$ DEF $\mathit{IsExactSeqFor}$
$\langle 1 \rangle 21.$ $K \in \mathit{Seq}(D)$ BY $\langle 1 \rangle 9$, $\langle 1 \rangle 20$, $\mathit{SeqSupset}$
$\langle 1 \rangle 22.$ $\mathit{DeltaVecFunIndexSum}(F,\ K) \in \mathit{DeltaVecType}$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 21$, $\mathit{DeltaVecFunIndexSumType}$

$\langle 1 \rangle 23.$ $\mathit{DeltaVecFunIndexSum}(F,\ J) = \mathit{DeltaVecAdd}(F[x],\ \mathit{DeltaVecFunIndexSum}(F,\ K))$
$\langle 2 \rangle$ USE $\langle 1 \rangle 1$, $\langle 1 \rangle 17$, $\langle 1 \rangle 18$
$\langle 2 \rangle$ USE $\mathit{DeltaVecFunIndexSumRemoveAt}$
$\langle 2 \rangle$ USE DEF K
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle 24.$ $\mathit{DeltaVecFunIndexSum}(F,\ I) = \mathit{DeltaVecFunIndexSum}(F,\ K)$

$\langle 2\rangle$ $DeltaVecFunIndexSumAnyExactSeq\_Conclusion(F, S, I, K)$
$\langle 3\rangle$ $DeltaVecFunIndexSumAnyExactSeq\_Hypothesis(F, S, I, K)$
$\langle 4\rangle$ USE $\langle 1\rangle 1$, $\langle 1\rangle 9$, $\langle 1\rangle 13$, $\langle 1\rangle 19$
$\langle 4\rangle$ QED BY DEF $DeltaVecFunIndexSumAnyExactSeq\_Hypothesis$
$\langle 3\rangle$ QED BY $DeltaVecFunIndexSumAnyExactSeq$
$\langle 2\rangle$ QED BY DEF $DeltaVecFunIndexSumAnyExactSeq\_Conclusion$

$\langle 1\rangle 25.$ $DeltaVecAdd(F[x], DeltaVecFunIndexSum(F, I)) = DeltaVecFunIndexSum(F, J)$
BY $\langle 1\rangle 23$, $\langle 1\rangle 24$

$\langle 1\rangle 26.$ $DeltaVecAdd(F[x], DeltaVecFunSubsetSum(F, S)) = DeltaVecFunSubsetSum(F, T)$
BY $\langle 1\rangle 25$ DEF $DeltaVecFunSubsetSum$, I, J

$\langle 1\rangle 27.$ $DeltaVecAdd(DeltaVecFunSubsetSum(F, S), F[x]) = DeltaVecFunSubsetSum(F, T)$
$\langle 2\rangle 1.$ $F[x] \in DeltaVecType$ BY $\langle 1\rangle 1$, $\langle 1\rangle 5$
$\langle 2\rangle 2.$ $DeltaVecFunSubsetSum(F, S) \in DeltaVecType$
BY $\langle 1\rangle 1$, $\langle 1\rangle 3$, $\langle 1\rangle 9$, $DeltaVecFunSubsetSumType$ DEF D
$\langle 2\rangle$ QED BY $\langle 1\rangle 26$, $\langle 2\rangle 1$, $\langle 2\rangle 2$, $DeltaVecAddCommutative$

$\langle 1\rangle$ QED
$\langle 2\rangle$ USE $\langle 1\rangle 4$, $\langle 1\rangle 10$, $\langle 1\rangle 26$, $\langle 1\rangle 27$
$\langle 2\rangle$ QED BY DEF $DeltaVecFunSubsetSumNewElem\_Conclusion$, D, T

Adding element x to subset S does not change the subset sum when either already $x \in S$ or $F[x] = Zero$.

We define explicit operators to capture the hypothesis and conclusion. Otherwise the provers seem to have difficulty figuring out how to apply this theorem.

$DeltaVecFunSubsetSumElemNoChange\_Hypothesis(F, S, x) \triangleq$
LET
$D \triangleq$ DOMAIN F
IN
$\land F \in [D \rightarrow DeltaVecType]$
$\land S \subseteq D$
$\land IsFiniteSet(S)$
$\land x \in D$
$\land x \in S \lor F[x] = DeltaVecZero$

$DeltaVecFunSubsetSumElemNoChange\_Conclusion(F, S, x) \triangleq$
LET
$D \triangleq$ DOMAIN F
$T \triangleq S \cup \{x\}$
IN

$\wedge$ $DeltaVecFunSubsetSum(F, S) = DeltaVecFunSubsetSum(F, T)$
$\wedge$ $T \subseteq D$
$\wedge$ $IsFiniteSet(T)$

THEOREM $DeltaVecFunSubsetSumElemNoChange \triangleq$
ASSUME NEW F, NEW S, NEW x, $DeltaVecFunSubsetSumElemNoChange\_Hypothesis(F, S, x)$
PROVE $DeltaVecFunSubsetSumElemNoChange\_Conclusion(F, S, x)$
PROOF
⟨1⟩ DEFINE $D \triangleq$ DOMAIN F
⟨1⟩ DEFINE $T \triangleq S \cup \{x\}$
⟨1⟩ HIDE DEF D, T

⟨1⟩1. $F \in [D \rightarrow DeltaVecType]$ BY DEF $DeltaVecFunSubsetSumElemNoChange\_Hypothesis$, D

⟨1⟩2. $x \in D$ BY DEF $DeltaVecFunSubsetSumElemNoChange\_Hypothesis$, D
⟨1⟩3. $x \in S \vee F[x] = DeltaVecZero$ BY DEF $DeltaVecFunSubsetSumElemNoChange\_Hypothesis$

⟨1⟩4. $S \subseteq D$ BY DEF $DeltaVecFunSubsetSumElemNoChange\_Hypothesis$, D
⟨1⟩5. $T \subseteq D$ BY ⟨1⟩2, ⟨1⟩4 DEF T

⟨1⟩6. $IsFiniteSet(\{x\})$ BY $FiniteSetSingleton$
⟨1⟩7. $IsFiniteSet(S)$ BY DEF $DeltaVecFunSubsetSumElemNoChange\_Hypothesis$
⟨1⟩8. $IsFiniteSet(T)$ BY ⟨1⟩6, ⟨1⟩7, $FiniteSetUnion$ DEF T

⟨1⟩9. $DeltaVecFunSubsetSum(F, S) = DeltaVecFunSubsetSum(F, T)$
⟨2⟩1. CASE $x \in S$ BY ⟨2⟩1 DEF T
⟨2⟩2. CASE $x \notin S$
⟨3⟩1. $DeltaVecAdd(F[x], DeltaVecFunSubsetSum(F, S)) = DeltaVecFunSubsetSum(F, T)$
⟨4⟩ $DeltaVecFunSubsetSumNewElem\_Conclusion(F, S, x)$
⟨5⟩ $DeltaVecFunSubsetSumNewElem\_Hypothesis(F, S, x)$
⟨6⟩ USE ⟨1⟩1, ⟨1⟩2, ⟨1⟩4, ⟨1⟩7, ⟨2⟩2
⟨6⟩ QED BY DEF $DeltaVecFunSubsetSumNewElem\_Hypothesis$, D
⟨5⟩ QED BY $DeltaVecFunSubsetSumNewElem$
⟨4⟩ QED BY DEF $DeltaVecFunSubsetSumNewElem\_Conclusion$, T
⟨3⟩2. $F[x] = DeltaVecZero$ BY ⟨1⟩3, ⟨2⟩2
⟨3⟩3. $DeltaVecFunSubsetSum(F, S) \in DeltaVecType$
⟨4⟩ USE ⟨1⟩1, ⟨1⟩4, ⟨1⟩7
⟨4⟩ QED BY $DeltaVecFunSubsetSumType$ DEF D
⟨3⟩ QED BY ⟨3⟩1, ⟨3⟩2, ⟨3⟩3, $DeltaVecZeroType$, $DeltaVecAddZero$
⟨2⟩ QED BY ⟨2⟩1, ⟨2⟩2

⟨1⟩ QED
⟨2⟩ USE ⟨1⟩5, ⟨1⟩8, ⟨1⟩9
⟨2⟩ QED BY DEF $DeltaVecFunSubsetSumElemNoChange\_Conclusion$, D, T

If two functions F and G have the same value on every $s \in S$, then their subset sums on S are the same.

We define explicit operators to capture the hypothesis and conclusion. Otherwise the provers seem to have difficulty figuring out how to apply this theorem.

$DeltaVecFunSubsetSumSameSubset\_Hypothesis(F, G, S) \triangleq$
LET
  $DF \triangleq$ DOMAIN F
  $DG \triangleq$ DOMAIN G
IN
  $\land F \in [DF \to DeltaVecType]$
  $\land G \in [DG \to DeltaVecType]$
  $\land S \subseteq DF$
  $\land S \subseteq DG$
  $\land IsFiniteSet(S)$
  $\land \forall s \in S : F[s] = G[s]$

$DeltaVecFunSubsetSumSameSubset\_Conclusion(F, G, S) \triangleq$
  $\land DeltaVecFunSubsetSum(F, S) = DeltaVecFunSubsetSum(G, S)$

THEOREM $DeltaVecFunSubsetSumSameSubset \triangleq$
  ASSUME NEW F, NEW G, NEW S, $DeltaVecFunSubsetSumSameSubset\_Hypothesis(F, G, S)$
  PROVE $DeltaVecFunSubsetSumSameSubset\_Conclusion(F, G, S)$
PROOF
  $\langle 1 \rangle$ DEFINE $DF \triangleq$ DOMAIN F
  $\langle 1 \rangle$ DEFINE $DG \triangleq$ DOMAIN G
  $\langle 1 \rangle$ DEFINE $I \triangleq ExactSeqFor(S)$
  $\langle 1 \rangle$ DEFINE $QF \triangleq [i \in 1 \,..\, Len(I) \mapsto F[I[i]]]$
  $\langle 1 \rangle$ DEFINE $QG \triangleq [i \in 1 \,..\, Len(I) \mapsto G[I[i]]]$
  $\langle 1 \rangle$ HIDE DEF DF, DG, I, QF, QG

  $\langle 1 \rangle 1.$ $F \in [DF \to DeltaVecType]$ BY DEF $DeltaVecFunSubsetSumSameSubset\_Hypothesis, DF$
  $\langle 1 \rangle 2.$ $G \in [DG \to DeltaVecType]$ BY DEF $DeltaVecFunSubsetSumSameSubset\_Hypothesis, DG$

  $\langle 1 \rangle 3.$ $S \subseteq DF$ BY DEF $DeltaVecFunSubsetSumSameSubset\_Hypothesis, DF$
  $\langle 1 \rangle 4.$ $S \subseteq DG$ BY DEF $DeltaVecFunSubsetSumSameSubset\_Hypothesis, DG$

  $\langle 1 \rangle 5.$ $IsFiniteSet(S)$ BY DEF $DeltaVecFunSubsetSumSameSubset\_Hypothesis$
  $\langle 1 \rangle 6.$ $\forall s \in S : F[s] = G[s]$ BY DEF $DeltaVecFunSubsetSumSameSubset\_Hypothesis$

  $\langle 1 \rangle 7.$ $DeltaVecFunSubsetSum(F, S) = DeltaVecSeqSum(QF)$
    $\langle 2 \rangle 1.$ $DeltaVecFunSubsetSum(F, S) = DeltaVecFunIndexSum(F, I)$ BY DEF $DeltaVecFunSubsetSum, I$
    $\langle 2 \rangle 2.$ $DeltaVecFunIndexSum(F, I) = DeltaVecSeqSum(QF)$ BY DEF $DeltaVecFunIndexSum, QF$
    $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1, \langle 2 \rangle 2$

$\langle 1 \rangle 8.$ $DeltaVecFunSubsetSum(G, S) = DeltaVecSeqSum(QG)$
$\langle 2 \rangle 1.$ $DeltaVecFunSubsetSum(G, S) = DeltaVecFunIndexSum(G, I)$ BY DEF $DeltaVecFunSubsetSum, I$
$\langle 2 \rangle 2.$ $DeltaVecFunIndexSum(G, I) = DeltaVecSeqSum(QG)$ BY DEF $DeltaVecFunIndexSum, QG$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1, \langle 2 \rangle 2$

$\langle 1 \rangle 9.$ $QF = QG$
$\langle 2 \rangle 1.$ $IsExactSeqFor(I, S)$ BY $\langle 1 \rangle 5$, $ExactSeqForProperties$ DEF I
$\langle 2 \rangle 2.$ $I \in Seq(S)$ BY $\langle 2 \rangle 1$ DEF $IsExactSeqFor$
$\langle 2 \rangle 3.$ $\forall i \in 1 \,..\, Len(I) : I[i] \in S$ BY $\langle 2 \rangle 2$, $ElementOfSeq$
$\langle 2 \rangle 4.$ $QF \in [1 \,..\, Len(I) \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 1, \langle 1 \rangle 3, \langle 2 \rangle 3$ DEF QF
$\langle 2 \rangle 5.$ $QG \in [1 \,..\, Len(I) \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 2, \langle 1 \rangle 4, \langle 2 \rangle 3$ DEF QG
$\langle 2 \rangle 6.$ $\forall i \in 1 \,..\, Len(I) : QF[i] = QG[i]$ BY $\langle 1 \rangle 6, \langle 2 \rangle 3$ DEF QF, QG
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 4, \langle 2 \rangle 5, \langle 2 \rangle 6$, Isa

$\langle 1 \rangle 10.$ $DeltaVecFunSubsetSum(F, S) = DeltaVecFunSubsetSum(G, S)$ BY $\langle 1 \rangle 7, \langle 1 \rangle 8, \langle 1 \rangle 9$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 10$ DEF $DeltaVecFunSubsetSumSameSubset\_Conclusion$

Adding a delta vector at a point in a function preserves the condition that the function has a range of delta vectors.

THEOREM $DeltaVecFunAddAtPreservesType \triangleq$
ASSUME
NEW D,
NEW $F \in [D \rightarrow DeltaVecType]$,
NEW $d \in D$,
NEW $v \in DeltaVecType$
PROVE
$DeltaVecFunAddAt(F, d, v) \in [D \rightarrow DeltaVecType]$
PROOF
$\langle 1 \rangle$ DEFINE $E \triangleq DeltaVecFunAddAt(F, d, v)$
$\langle 1 \rangle$ HIDE DEF E
$\langle 1 \rangle$ SUFFICES $E \in [D \rightarrow DeltaVecType]$ BY DEF E
$\langle 1 \rangle 1.$ $E = [F$ EXCEPT $![d] = DeltaVecAdd(F[d], v)]$ BY DEF $DeltaVecFunAddAt, E$
$\langle 1 \rangle$ SUFFICES ASSUME NEW $k \in D$ PROVE $E[k] \in DeltaVecType$ BY $\langle 1 \rangle 1$
$\langle 1 \rangle 2.$ CASE $k \neq d$
$\langle 2 \rangle 1.$ $E[k] = F[k]$ BY $\langle 1 \rangle 1, \langle 1 \rangle 2$
$\langle 2 \rangle 2.$ $F[k] \in DeltaVecType$ OBVIOUS
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1, \langle 2 \rangle 2$
$\langle 1 \rangle 3.$ CASE $k = d$
$\langle 2 \rangle 1.$ $E[k] = DeltaVecAdd(F[k], v)$ BY $\langle 1 \rangle 1, \langle 1 \rangle 3$
$\langle 2 \rangle 2.$ $F[k] \in DeltaVecType$ OBVIOUS
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1, \langle 2 \rangle 2$, $DeltaVecAddType$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$

Adding a delta vector at a point in a function preserves the condition that the function has a finite non-zero range.

THEOREM $DeltaVecFunAddAtPreservesFiniteNonZeroRange \triangleq$
  ASSUME
    NEW D,
    NEW $F \in [D \rightarrow DeltaVecType]$, $DeltaVecFunHasFiniteNonZeroRange(F)$,
    NEW $d \in D$,
    NEW $v \in DeltaVecType$
  PROVE
  $DeltaVecFunHasFiniteNonZeroRange(DeltaVecFunAddAt(F, d, v))$
PROOF
  $\langle 1 \rangle$ DEFINE $E \triangleq DeltaVecFunAddAt(F, d, v)$
  $\langle 1 \rangle$ DEFINE $SF \triangleq \{k \in D : F[k] \neq DeltaVecZero\}$
  $\langle 1 \rangle$ DEFINE $SE \triangleq \{k \in D : E[k] \neq DeltaVecZero\}$
  $\langle 1 \rangle$ HIDE DEF E, SF, SE
  $\langle 1 \rangle$ SUFFICES $DeltaVecFunHasFiniteNonZeroRange(E)$ BY DEF E
  $\langle 1 \rangle 1.$ $E \in [D \rightarrow DeltaVecType]$ BY $DeltaVecFunAddAtPreservesType$ DEF E
  $\langle 1 \rangle$ SUFFICES $IsFiniteSet(SE)$ BY $\langle 1 \rangle 1$ DEF $DeltaVecFunHasFiniteNonZeroRange$, SE
  $\langle 1 \rangle 2.$ $E = [F$ EXCEPT $![d] = DeltaVecAdd(F[d], v)]$ BY DEF $DeltaVecFunAddAt$, E
  $\langle 1 \rangle 3.$ $IsFiniteSet(SF)$ BY DEF $DeltaVecFunHasFiniteNonZeroRange$, SF
  $\langle 1 \rangle 4.$ $IsFiniteSet(\{d\})$ BY $FiniteSetSingleton$
  $\langle 1 \rangle 5.$ $IsFiniteSet(SF \cup \{d\})$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $FiniteSetUnion$
  $\langle 1 \rangle$ SUFFICES $SE \subseteq (SF \cup \{d\})$ BY $\langle 1 \rangle 5$, $FiniteSetSubset$
  $\langle 1 \rangle 6.$ SUFFICES ASSUME NEW $k \in SE$, $k \notin SF$, $k \neq d$ PROVE FALSE OBVIOUS
  $\langle 1 \rangle 7.$ $k \in D$ BY $\langle 1 \rangle 6$ DEF SE
  $\langle 1 \rangle 8.$ $E[k] \neq DeltaVecZero$ BY $\langle 1 \rangle 6$ DEF SE
  $\langle 1 \rangle 9.$ $F[k] = DeltaVecZero$ BY $\langle 1 \rangle 6$, $\langle 1 \rangle 7$ DEF SF
  $\langle 1 \rangle 10.$ $E[k] = F[k]$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $\langle 1 \rangle 6$, $\langle 1 \rangle 7$
  $\langle 1 \rangle$ QED BY $\langle 1 \rangle 8$, $\langle 1 \rangle 9$, $\langle 1 \rangle 10$

Adding delta vector v to $F[x]$ increases the sum of the function by v.

We define explicit operators to capture the hypothesis and conclusion. Otherwise the provers seem to have difficulty figuring out how to apply this theorem.

The strategy of the proof is as follows. We define G as the function produced by adding v to component x in function F. Clearly, F and G have the same domain and if F has a finite non-zero range of delta vectors, so will G.

We define S as the subset of the domain of F that maps to non-zero values, and likewise T as the subset of the domain of G that maps to non-zero values. Note that since F and G are identical everywhere except possibly at x, S and T will be the same except that possibly one or the other will include x.

To get rid of this "possibly", we expand S to $S1$ by adding the element x if it was not already included. Note that $SubsetSum(F, S) = SubsetSum(F, S1)$ because S already contained all domain elements that map to non-zero values under F.

We do the same thing with T, and now we have sets $S1$ and $T1$ that are identical and include x. From these sets we construct $S2$ and $T2$ respectively by taking x out. Since F and G are identical everywhere except possibly at x, we can conclude that $S2 = T2$ and consequently that $SubsetSum(F, S2) = SubsetSum(G, T2)$.

We use the $DeltaVecFunSubsetSumNewElem$ lemma to relate the $S1$ and $T1$ subset sums to the respective $S2$ and $T2$ subset sums. Putting everything together with a little commutativity and associativity gives us the conclusion of the theorem.

$DeltaVecFunSumAddAt\_Hypothesis(F, x, v) \triangleq$
LET
$D \triangleq$ DOMAIN F
IN
$\land F \in [D \rightarrow DeltaVecType]$
$\land DeltaVecFunHasFiniteNonZeroRange(F)$
$\land x \in D$
$\land v \in DeltaVecType$

$DeltaVecFunSumAddAt\_Conclusion(F, x, v) \triangleq$
LET
$D \triangleq$ DOMAIN F
$G \triangleq DeltaVecFunAddAt(F, x, v)$
IN
$\land G \in [D \rightarrow DeltaVecType]$
$\land DeltaVecFunHasFiniteNonZeroRange(G)$
$\land DeltaVecFunSum(G) \in DeltaVecType$
$\land DeltaVecFunSum(G) = DeltaVecAdd(v, DeltaVecFunSum(F))$
$\land DeltaVecFunSum(G) = DeltaVecAdd(DeltaVecFunSum(F), v)$

THEOREM $DeltaVecFunSumAddAt \triangleq$
ASSUME NEW F, NEW x, NEW v, $DeltaVecFunSumAddAt\_Hypothesis(F, x, v)$
PROVE $DeltaVecFunSumAddAt\_Conclusion(F, x, v)$
PROOF

Define some convenient abbreviations for this proof.

$\langle 1 \rangle$ DEFINE $Type \triangleq DeltaVecType$
$\langle 1 \rangle$ DEFINE $Zero \triangleq DeltaVecZero$
$\langle 1 \rangle$ DEFINE $Add(\_a, \_b) \triangleq DeltaVecAdd(\_a, \_b)$
$\langle 1 \rangle$ DEFINE $SubsetSum(\_F, \_S) \triangleq DeltaVecFunSubsetSum(\_F, \_S)$
$\langle 1 \rangle$ DEFINE $Sum(\_F) \triangleq DeltaVecFunSum(\_F)$
$\langle 1 \rangle$ DEFINE $AddAt(\_F, \_x, \_v) \triangleq DeltaVecFunAddAt(\_F, \_x, \_v)$

$\langle 1 \rangle$ DEFINE $D \triangleq$ DOMAIN F

$\langle 1 \rangle$ DEFINE $G \triangleq AddAt(F, x, v)$
$\langle 1 \rangle$ DEFINE $S \triangleq \{k \in D : F[k] \neq Zero\}$
$\langle 1 \rangle$ DEFINE $T \triangleq \{k \in D : G[k] \neq Zero\}$
$\langle 1 \rangle$ DEFINE $S1 \triangleq S \cup \{x\}$
$\langle 1 \rangle$ DEFINE $T1 \triangleq T \cup \{x\}$
$\langle 1 \rangle$ DEFINE $S2 \triangleq S1 \setminus \{x\}$
$\langle 1 \rangle$ DEFINE $T2 \triangleq T1 \setminus \{x\}$
$\langle 1 \rangle$ HIDE DEF $D, G, S, T, S1, T1, S2, T2$

Some easy initial facts.

$\langle 1 \rangle 1.$ $x \in D$ BY DEF $DeltaVecFunSumAddAt\_Hypothesis, D$
$\langle 1 \rangle 2.$ $v \in Type$ BY DEF $DeltaVecFunSumAddAt\_Hypothesis$

$\langle 1 \rangle 3.$ $F \in [D \rightarrow Type]$ BY DEF $DeltaVecFunSumAddAt\_Hypothesis, D$
$\langle 1 \rangle 4.$ $G \in [D \rightarrow Type]$
  $\langle 2 \rangle$ USE $\langle 1 \rangle 1, \langle 1 \rangle 2, \langle 1 \rangle 3$
  $\langle 2 \rangle$ QED BY $DeltaVecFunAddAtPreservesType$ DEF G

$\langle 1 \rangle 5.$ $DeltaVecFunHasFiniteNonZeroRange(F)$ BY DEF $DeltaVecFunSumAddAt\_Hypothesis$
$\langle 1 \rangle 6.$ $DeltaVecFunHasFiniteNonZeroRange(G)$
  $\langle 2 \rangle$ USE $\langle 1 \rangle 1, \langle 1 \rangle 2, \langle 1 \rangle 3, \langle 1 \rangle 5$
  $\langle 2 \rangle$ QED BY $DeltaVecFunAddAtPreservesFiniteNonZeroRange$ DEF G

$\langle 1 \rangle 7.$ $Sum(G) \in Type$ BY $\langle 1 \rangle 4, \langle 1 \rangle 6, DeltaVecFunSumType$

$\langle 1 \rangle 8.$ $S \subseteq D$ BY DEF S
$\langle 1 \rangle 9.$ $T \subseteq D$ BY DEF T
$\langle 1 \rangle 10.$ $S1 \subseteq D$ BY $\langle 1 \rangle 1, \langle 1 \rangle 8$ DEF $S1$
$\langle 1 \rangle 11.$ $T1 \subseteq D$ BY $\langle 1 \rangle 1, \langle 1 \rangle 9$ DEF $T1$
$\langle 1 \rangle 12.$ $S2 \subseteq D$ BY $\langle 1 \rangle 10$ DEF $S2$
$\langle 1 \rangle 13.$ $T2 \subseteq D$ BY $\langle 1 \rangle 11$ DEF $T2$

$\langle 1 \rangle 14.$ $IsFiniteSet(\{x\})$ BY $FiniteSetSingleton$
$\langle 1 \rangle 15.$ $IsFiniteSet(S)$ BY $\langle 1 \rangle 3, \langle 1 \rangle 5$ DEF $DeltaVecFunHasFiniteNonZeroRange, S$
$\langle 1 \rangle 16.$ $IsFiniteSet(T)$ BY $\langle 1 \rangle 4, \langle 1 \rangle 6$ DEF $DeltaVecFunHasFiniteNonZeroRange, T$
$\langle 1 \rangle 17.$ $IsFiniteSet(S1)$ BY $\langle 1 \rangle 14, \langle 1 \rangle 15, FiniteSetUnion$ DEF $S1$
$\langle 1 \rangle 18.$ $IsFiniteSet(T1)$ BY $\langle 1 \rangle 14, \langle 1 \rangle 16, FiniteSetUnion$ DEF $T1$
$\langle 1 \rangle 19.$ $IsFiniteSet(S2)$ BY $\langle 1 \rangle 17, FiniteSetSubset$ DEF $S2$
$\langle 1 \rangle 20.$ $IsFiniteSet(T2)$ BY $\langle 1 \rangle 18, FiniteSetSubset$ DEF $T2$

$Sum(F)$ and $Sum(G)$ are the same as the respective subset sums over $S1$ and $T1$.

$\langle 1 \rangle 21.$ $Sum(F) = SubsetSum(F, S1)$
  $\langle 2 \rangle 1.$ $Sum(F) = SubsetSum(F, S)$ BY $\langle 1 \rangle 3$ DEF $DeltaVecFunSum, S$
  $\langle 2 \rangle 2.$ $SubsetSum(F, S) = SubsetSum(F, S1)$
    $\langle 3 \rangle$ $DeltaVecFunSubsetSumElemNoChange\_Conclusion(F, S, x)$
      $\langle 4 \rangle$ $DeltaVecFunSubsetSumElemNoChange\_Hypothesis(F, S, x)$
        $\langle 5 \rangle$ $x \in S \lor F[x] = DeltaVecZero$ BY $\langle 1 \rangle 1$ DEF S

$\langle 5 \rangle$ USE $\langle 1 \rangle 1,\ \langle 1 \rangle 3,\ \langle 1 \rangle 8,\ \langle 1 \rangle 15$
$\langle 5 \rangle$ QED BY DEF $\mathit{DeltaVecFunSubsetSumElemNoChange\_Hypothesis}$
$\langle 4 \rangle$ QED BY $\mathit{DeltaVecFunSubsetSumElemNoChange}$
$\langle 3 \rangle$ QED BY DEF $\mathit{DeltaVecFunSubsetSumElemNoChange\_Conclusion},\ S1$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 2$

$\langle 1 \rangle 22.\ \mathit{Sum}(G) = \mathit{SubsetSum}(G,\ T1)$
$\langle 2 \rangle 1.\ \mathit{Sum}(G) = \mathit{SubsetSum}(G,\ T)$ BY $\langle 1 \rangle 4$ DEF $\mathit{DeltaVecFunSum},\ T$
$\langle 2 \rangle 2.\ \mathit{SubsetSum}(G,\ T) = \mathit{SubsetSum}(G,\ T1)$
$\langle 3 \rangle\ \mathit{DeltaVecFunSubsetSumElemNoChange\_Conclusion}(G,\ T,\ x)$
$\langle 4 \rangle\ \mathit{DeltaVecFunSubsetSumElemNoChange\_Hypothesis}(G,\ T,\ x)$
$\langle 5 \rangle\ x \in T \lor G[x] = \mathit{Zero}$ BY $\langle 1 \rangle 1$ DEF T
$\langle 5 \rangle$ USE $\langle 1 \rangle 1,\ \langle 1 \rangle 4,\ \langle 1 \rangle 9,\ \langle 1 \rangle 16$
$\langle 5 \rangle$ QED BY DEF $\mathit{DeltaVecFunSubsetSumElemNoChange\_Hypothesis}$
$\langle 4 \rangle$ QED BY $\mathit{DeltaVecFunSubsetSumElemNoChange}$
$\langle 3 \rangle$ QED BY DEF $\mathit{DeltaVecFunSubsetSumElemNoChange\_Conclusion},\ T1$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 2$

Relate the $S1$ and $T1$ subset sums to the respective sums over the smaller subsets $S2$ and $T2$.

$\langle 1 \rangle 23.\ \mathit{SubsetSum}(F,\ S1) = \mathit{Add}(F[x],\ \mathit{SubsetSum}(F,\ S2))$
$\langle 2 \rangle 1.\ \mathit{SubsetSum}(F,\ S2 \cup \{x\}) = \mathit{Add}(F[x],\ \mathit{SubsetSum}(F,\ S2))$
$\langle 3 \rangle\ \mathit{DeltaVecFunSubsetSumNewElem\_Conclusion}(F,\ S2,\ x)$
$\langle 4 \rangle\ \mathit{DeltaVecFunSubsetSumNewElem\_Hypothesis}(F,\ S2,\ x)$
$\langle 5 \rangle\ x \notin S2$ BY DEF $S2$
$\langle 5 \rangle$ USE $\langle 1 \rangle 1,\ \langle 1 \rangle 3,\ \langle 1 \rangle 12,\ \langle 1 \rangle 19$
$\langle 5 \rangle$ QED BY DEF $\mathit{DeltaVecFunSubsetSumNewElem\_Hypothesis}$
$\langle 4 \rangle$ QED BY $\mathit{DeltaVecFunSubsetSumNewElem}$
$\langle 3 \rangle$ QED BY DEF $\mathit{DeltaVecFunSubsetSumNewElem\_Conclusion}$
$\langle 2 \rangle 2.\ S1 = S2 \cup \{x\}$
$\langle 3 \rangle\ x \in S1$ BY DEF $S1$
$\langle 3 \rangle$ QED BY DEF $S2$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 2$

$\langle 1 \rangle 24.\ \mathit{SubsetSum}(G,\ T1) = \mathit{Add}(G[x],\ \mathit{SubsetSum}(G,\ T2))$
$\langle 2 \rangle 1.\ \mathit{SubsetSum}(G,\ T2 \cup \{x\}) = \mathit{Add}(G[x],\ \mathit{SubsetSum}(G,\ T2))$
$\langle 3 \rangle\ \mathit{DeltaVecFunSubsetSumNewElem\_Conclusion}(G,\ T2,\ x)$
$\langle 4 \rangle\ \mathit{DeltaVecFunSubsetSumNewElem\_Hypothesis}(G,\ T2,\ x)$
$\langle 5 \rangle\ x \notin T2$ BY DEF $T2$
$\langle 5 \rangle$ USE $\langle 1 \rangle 1,\ \langle 1 \rangle 4,\ \langle 1 \rangle 13,\ \langle 1 \rangle 20$
$\langle 5 \rangle$ QED BY DEF $\mathit{DeltaVecFunSubsetSumNewElem\_Hypothesis}$
$\langle 4 \rangle$ QED BY $\mathit{DeltaVecFunSubsetSumNewElem}$
$\langle 3 \rangle$ QED BY DEF $\mathit{DeltaVecFunSubsetSumNewElem\_Conclusion}$
$\langle 2 \rangle 2.\ T1 = T2 \cup \{x\}$
$\langle 3 \rangle\ x \in T1$ BY DEF $T1$
$\langle 3 \rangle$ QED BY DEF $T2$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 2$

The consequences of $G = AddAt(F, x, v)$

$\langle 1 \rangle 25.$ $G = [F$ EXCEPT $![x] = Add(F[x], v)]$ BY DEF $DeltaVecFunAddAt, G$

$\langle 1 \rangle 26.$ $G[x] = Add(v, F[x])$
  $\langle 2 \rangle$ $G[x] = Add(F[x], v)$ BY $\langle 1 \rangle 1, \langle 1 \rangle 3, \langle 1 \rangle 25$
  $\langle 2 \rangle$ $F[x] \in Type$ BY $\langle 1 \rangle 1, \langle 1 \rangle 3$
  $\langle 2 \rangle$ $v \in Type$ BY $\langle 1 \rangle 2$
  $\langle 2 \rangle$ QED BY $DeltaVecAddCommutative$

$\langle 1 \rangle 27.$ $S2 = T2$
  $\langle 2 \rangle$ $S1 = T1$
    $\langle 3 \rangle$ $\forall k \in D : k \neq x \Rightarrow F[k] = G[k]$ BY $\langle 1 \rangle 3, \langle 1 \rangle 4, \langle 1 \rangle 25$
    $\langle 3 \rangle$ QED BY DEF $S, T, S1, T1$
  $\langle 2 \rangle$ QED BY DEF $S2, T2$

$\langle 1 \rangle 28.$ $SubsetSum(F, S2) = SubsetSum(G, T2)$
  $\langle 2 \rangle$ SUFFICES $SubsetSum(F, S2) = SubsetSum(G, S2)$ BY $\langle 1 \rangle 27$
  $\langle 2 \rangle$ $DeltaVecFunSubsetSumSameSubset\_Conclusion(F, G, S2)$
    $\langle 3 \rangle$ $DeltaVecFunSubsetSumSameSubset\_Hypothesis(F, G, S2)$
      $\langle 4 \rangle$ USE $\langle 1 \rangle 3, \langle 1 \rangle 4, \langle 1 \rangle 12, \langle 1 \rangle 19, \langle 1 \rangle 25$
      $\langle 4 \rangle$ $x \notin S2$ BY DEF $S2$
      $\langle 4 \rangle$ $\forall s \in S2 : F[s] = G[s]$ OBVIOUS
      $\langle 4 \rangle$ QED BY DEF $DeltaVecFunSubsetSumSameSubset\_Hypothesis$
    $\langle 3 \rangle$ QED BY $DeltaVecFunSubsetSumSameSubset$
  $\langle 2 \rangle$ QED BY DEF $DeltaVecFunSubsetSumSameSubset\_Conclusion$

$\langle 1 \rangle 29.$ $SubsetSum(G, T1) = Add(v, SubsetSum(F, S1))$
  $\langle 2 \rangle$ $SubsetSum(G, T1) = Add(v, Add(F[x], SubsetSum(F, S2)))$
    $\langle 3 \rangle$ $SubsetSum(G, T1) = Add(Add(v, F[x]), SubsetSum(F, S2))$ BY $\langle 1 \rangle 24, \langle 1 \rangle 26, \langle 1 \rangle 28$
    $\langle 3 \rangle$ $F[x] \in Type$ BY $\langle 1 \rangle 1, \langle 1 \rangle 3$
    $\langle 3 \rangle$ $v \in Type$ BY $\langle 1 \rangle 2$
    $\langle 3 \rangle$ $SubsetSum(F, S2) \in Type$ BY $\langle 1 \rangle 3, \langle 1 \rangle 12, \langle 1 \rangle 19, DeltaVecFunSubsetSumType$
    $\langle 3 \rangle$ QED BY $DeltaVecAddAssociative$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 23$

$\langle 1 \rangle 30.$ $Sum(G) = Add(v, Sum(F))$ BY $\langle 1 \rangle 21, \langle 1 \rangle 22, \langle 1 \rangle 29$

$\langle 1 \rangle 31.$ $Sum(G) = Add(Sum(F), v)$
  $\langle 2 \rangle 1.$ $v \in Type$ BY $\langle 1 \rangle 2$
  $\langle 2 \rangle 2.$ $Sum(F) \in Type$BY $\langle 1 \rangle 3, \langle 1 \rangle 5, DeltaVecFunSumType$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 30, \langle 2 \rangle 1, \langle 2 \rangle 2, DeltaVecAddCommutative$

$\langle 1 \rangle$ QED
  $\langle 2 \rangle$ USE $\langle 1 \rangle 4, \langle 1 \rangle 6, \langle 1 \rangle 7, \langle 1 \rangle 30, \langle 1 \rangle 31$
  $\langle 2 \rangle$ QED BY DEF $DeltaVecFunSumAddAt\_Conclusion, G, D$

C.11 Facts about upright delta vectors

MODULE $NaiadClockProofDeltaVecUpright$

EXTENDS $NaiadClockProofDeltaVecFuns$

Facts about upright delta vectors.

If v is an upright delta vector then for every positive point t there is some negative point $s \prec t$ such that v is nonpos up thru s. We call point s a support in v for point t.

THEOREM $DeltaVecUpright\_ExistsSupport \triangleq$
ASSUME
NEW $leq \in PointRelationType$,
NEW $v \in DeltaVecType$,
NEW $t \in Point$,
$IsPartialOrder(leq)$,
$IsDeltaVecUpright(leq, v)$,
$v[t] > 0$
PROVE
LET
$a \preceq b \triangleq leq[a][b]$
$a \prec b \triangleq a \preceq b \land a \neq b$
IN
$\exists s \in Point :$
$\land\ s \prec t$
$\land\ v[s] < 0$
$\land\ IsDeltaVecNonposUpto(leq, v, s)$
PROOF
$\langle 1 \rangle 1.$ $IsDeltaVecSupported(leq, v, t)$ BY DEF $IsDeltaVecUpright$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$ DEF $IsDeltaVecSupported$

DeltaVecZero is an upright delta vector.

THEOREM $DeltaVecUpright\_Zero \triangleq$

ASSUME
NEW $leq \in PointRelationType$,
$IsPartialOrder(leq)$
PROVE
$IsDeltaVecUpright(leq,\ DeltaVecZero)$
PROOF
$\langle 1 \rangle$ QED BY DEF $DeltaVecZero,\ IsDeltaVecUpright,\ IsDeltaVecSupported,\ Isa$

The sum of two upright delta vectors is an upright delta vector.

THEOREM $DeltaVecUpright\_Add\ \triangleq$
ASSUME
NEW $leq \in PointRelationType$,
NEW $v1 \in DeltaVecType$,
NEW $v2 \in DeltaVecType$,
$IsPartialOrder(leq)$,
$IsDeltaVecUpright(leq,\ v1)$,
$IsDeltaVecUpright(leq,\ v2)$
PROVE
$IsDeltaVecUpright(leq,\ DeltaVecAdd(v1,\ v2))$
PROOF
$\langle 1 \rangle 1.$ PICK $v0 \in DeltaVecType : v0 = DeltaVecAdd(v1,\ v2)$ BY $DeltaVecAddType$

$\langle 1 \rangle$ DEFINE $a \preceq b\ \triangleq\ leq[a][b]$
$\langle 1 \rangle$ DEFINE $a \prec b\ \triangleq\ a \preceq b \land a \neq b$

Assume that point t is positive in $v0$. It suffices to find a support for t. A support is a point lower than t that is negative in $v0$ and no yet lower point is positive in $v0$.

$\langle 1 \rangle 2.$ SUFFICES ASSUME
NEW $t \in Point$,
$v0[t] \ > 0$
PROVE
$\exists\, s \ \in Point :$
$\land\ s \prec t$
$\land\ v0[s] < 0$
$\land\ \neg \exists\, u \ \in Point : u \preceq s \land v0[u] > 0$
BY $\langle 1 \rangle 1$ DEF $IsDeltaVecUpright,\ IsDeltaVecSupported,\ IsDeltaVecNonposUpto$

$\langle 1 \rangle 3.$ $v1[t] > 0 \lor v2[t] > 0$ BY $\langle 1 \rangle 1,\ \langle 1 \rangle 2,\ SMTT(10)$ DEF $DeltaVecAdd,\ DeltaVecType$

Without loss of generality, pick va as whichever of $v1$ or $v2$ is positive at point t, and pick vb as the other.

$\langle 1 \rangle$ DEFINE $vaisv1 \triangleq v1[t] > 0$
$\langle 1 \rangle$ HIDE DEF $vaisv1$

$\langle 1 \rangle 4.$ PICK $va \in DeltaVecType : va =$ IF $vaisv1$ THEN $v1$ ELSE $v2$ OBVIOUS
$\langle 1 \rangle 5.$ PICK $vb \in DeltaVecType : vb =$ IF $vaisv1$ THEN $v2$ ELSE $v1$ OBVIOUS
$\langle 1 \rangle 6.$ $IsDeltaVecUpright(leq, va)$ BY $\langle 1 \rangle 4$
$\langle 1 \rangle 7.$ $IsDeltaVecUpright(leq, vb)$ BY $\langle 1 \rangle 5$
$\langle 1 \rangle 8.$ $v0 = DeltaVecAdd(va, vb)$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 4$, $\langle 1 \rangle 5$, $DeltaVecAddCommutative$
$\langle 1 \rangle 9.$ $va[t] > 0$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 4$ DEF $vaisv1$

Since va is upright and $va[t] > 0$, we can pick a lower point x that is negative in va and no yet lower point is positive in va.

$\langle 1 \rangle 10.$ PICK $x \in Point$:
$\wedge\ x \prec t$
$\wedge\ va[x] < 0$
$\wedge\ IsDeltaVecNonposUpto(leq, va, x)$
BY $\langle 1 \rangle 6$, $\langle 1 \rangle 9$, $DeltaVecUpright\_ExistsSupport$

State what we know about x as separate facts.

$\langle 1 \rangle 11.$ $x \prec t$ BY $\langle 1 \rangle 10$
$\langle 1 \rangle 12.$ $va[x] < 0$ BY $\langle 1 \rangle 10$
$\langle 1 \rangle 13.$ $IsDeltaVecNonposUpto(leq, va, x)$ BY $\langle 1 \rangle 10$

$\langle 1 \rangle 14.$ CASE $\neg \exists\, s \in Point : s \preceq x \wedge vb[s] > 0$

No point up thru x is positive in vb. In this case, point x is a support in $v0$ for point t, since in $v0$ point x must be negative and no lower point can be positive.

$\langle 2 \rangle 1.$ $\neg(vb[x] > 0)$ BY $\langle 1 \rangle 14$, $PartialOrderReflexive$
$\langle 2 \rangle 2.$ $v0[x] < 0$ BY $\langle 2 \rangle 1$, $\langle 1 \rangle 8$, $\langle 1 \rangle 12$, $SMTT(10)$ DEF $DeltaVecAdd$, $DeltaVecType$
$\langle 2 \rangle 3.$ ASSUME NEW $u \in Point$, $u \preceq x$, $v0[u] > 0$ PROVE FALSE
$\langle 3 \rangle 1.$ $\neg(va[u] > 0)$ BY $\langle 2 \rangle 3$, $\langle 1 \rangle 13$ DEF $IsDeltaVecNonposUpto$
$\langle 3 \rangle 2.$ $\neg(vb[u] > 0)$ BY $\langle 2 \rangle 3$, $\langle 1 \rangle 14$
$\langle 3 \rangle 3.$ $\neg(v0[u] > 0)$ BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 1 \rangle 8$, $SMTT(10)$ DEF $DeltaVecAdd$, $DeltaVecType$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 3$, $\langle 2 \rangle 3$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 11$, $\langle 2 \rangle 2$, $\langle 2 \rangle 3$

$\langle 1 \rangle 15.$ CASE $\exists\, s \in Point : s \preceq x \wedge vb[s] > 0$

Some point at or lower than x is positive in vb. We pick one.

$\langle 2 \rangle 1.$ PICK $s \in Point : s \preceq x \wedge vb[s] > 0$ BY $\langle 1 \rangle 15$

State what we know about s as separate facts.

$\langle 2 \rangle 2.$ $s \preceq x$ BY $\langle 2 \rangle 1$
$\langle 2 \rangle 3.$ $vb[s] > 0$ BY $\langle 2 \rangle 1$

Since vb is upright and $vb[s] > 0$, we can pick a lower point y that is negative in vb and no yet lower point is positive in vb.

Since point $y \prec s \preceq x$, no point at or lower than y can be positive in va. Therefore, in $v0$ point y must be negative and no yet lower point can be positive. Hence point y is a support in $v0$ for point t.

$\langle 2 \rangle 4.$ PICK $y \in Point$:
$\wedge\ y \prec s$
$\wedge\ vb[y] < 0$

$\wedge$ $IsDeltaVecNonposUpto(leq,\ vb,\ y)$
BY $\langle 1\rangle 7,\ \langle 2\rangle 3,\ DeltaVecUpright\_ExistsSupport$

State what we know about y as separate facts.

$\langle 2\rangle 5.\ y \prec s$ BY $\langle 2\rangle 4$
$\langle 2\rangle 6.\ vb[y] < 0$ BY $\langle 2\rangle 4$
$\langle 2\rangle 7.\ IsDeltaVecNonposUpto(leq,\ vb,\ y)$ BY $\langle 2\rangle 4$

$\langle 2\rangle 8.\ y \preceq x$ BY $\langle 2\rangle 2,\ \langle 2\rangle 5,\ PartialOrderTransitive$
$\langle 2\rangle 9.\ \neg(va[y] > 0)$ BY $\langle 1\rangle 13,\ \langle 2\rangle 8$ DEF $IsDeltaVecNonposUpto$
$\langle 2\rangle 10.\ v0[y] < 0$ BY $\langle 2\rangle 6,\ \langle 2\rangle 9,\ \langle 1\rangle 8,\ SMTT(10)$ DEF $DeltaVecAdd,\ DeltaVecType$
$\langle 2\rangle 11.$ ASSUME NEW $u \in Point,\ u \preceq y,\ v0[u] > 0$ PROVE FALSE
$\langle 3\rangle 1.\ u \preceq x$ BY $\langle 2\rangle 8,\ \langle 2\rangle 11,\ PartialOrderTransitive$
$\langle 3\rangle 2.\ \neg(va[u] > 0)$ BY $\langle 3\rangle 1,\ \langle 1\rangle 13$ DEF $IsDeltaVecNonposUpto$
$\langle 3\rangle 3.\ \neg(vb[u] > 0)$ BY $\langle 2\rangle 7,\ \langle 2\rangle 11$ DEF $IsDeltaVecNonposUpto$
$\langle 3\rangle 4.\ \neg(v0[u] > 0)$ BY $\langle 3\rangle 2,\ \langle 3\rangle 3,\ \langle 1\rangle 8,\ SMTT(10)$ DEF $DeltaVecAdd,\ DeltaVecType$
$\langle 3\rangle$ QED BY $\langle 3\rangle 4,\ \langle 2\rangle 11$
$\langle 2\rangle 12.\ y \prec t$ BY $\langle 2\rangle 8,\ \langle 1\rangle 11,\ PartialOrderStrictlyTransitive$
$\langle 2\rangle$ QED BY $\langle 2\rangle 10,\ \langle 2\rangle 11,\ \langle 2\rangle 12$

$\langle 1\rangle$ QED BY $\langle 1\rangle 14,\ \langle 1\rangle 15$

The skip k sum of a sequence of upright delta vectors is an upright delta vector.

COROLLARY $DeltaVecUpright\_SeqSkipSum \triangleq$
ASSUME
NEW $leq \in PointRelationType$,
NEW $Q \in Seq(DeltaVecType)$,
NEW $k \in Nat$,
$IsPartialOrder(leq)$,
$\forall\, i \in Nat : k < i \wedge i \leq Len(Q) \Rightarrow IsDeltaVecUpright(leq,\ Q[i])$
PROVE
$IsDeltaVecUpright(leq,\ DeltaVecSeqSkipSum(k,\ Q))$
PROOF
$\langle 1\rangle$ DEFINE $Prop(a) \triangleq IsDeltaVecUpright(leq,\ a)$
$\langle 1\rangle$ USE $DeltaVecUpright\_Zero$
$\langle 1\rangle$ USE $DeltaVecUpright\_Add$

$\langle 1\rangle\ DeltaVecSeqSkipSumProp\_Conclusion(Prop,\ Q,\ k)$
$\langle 2\rangle\ DeltaVecSeqSkipSumProp\_Hypothesis(Prop,\ Q,\ k)$
$\langle 3\rangle$ QED BY DEF $DeltaVecSeqSkipSumProp\_Hypothesis$
$\langle 2\rangle$ QED BY $DeltaVecSeqSkipSumProp$

$\langle 1 \rangle$ QED BY DEF $DeltaVecSeqSkipSumProp\_Conclusion,\ Prop$

The sum of a sequence of upright delta vectors is an upright delta vector.

COROLLARY $DeltaVecUpright\_SeqSum \triangleq$
ASSUME
NEW $leq \in PointRelationType$,
NEW $Q \in Seq(DeltaVecType)$,
NEW $k \in Nat$,
$IsPartialOrder(leq)$,
$\forall\, i \in 1 \,..\, Len(Q) : IsDeltaVecUpright(leq,\ Q[i])$
PROVE
$IsDeltaVecUpright(leq,\ DeltaVecSeqSum(Q))$
PROOF
$\langle 1 \rangle$ DEFINE $Prop(a) \triangleq IsDeltaVecUpright(leq,\ a)$
$\langle 1 \rangle$ USE $DeltaVecUpright\_Zero$
$\langle 1 \rangle$ USE $DeltaVecUpright\_Add$

$\langle 1 \rangle$ $DeltaVecSeqSumProp\_Conclusion(Prop,\ Q)$
$\langle 2 \rangle$ $DeltaVecSeqSumProp\_Hypothesis(Prop,\ Q)$
$\langle 3 \rangle$ QED BY DEF $DeltaVecSeqSumProp\_Hypothesis$
$\langle 2 \rangle$ QED BY $DeltaVecSeqSumProp$
$\langle 1 \rangle$ QED BY DEF $DeltaVecSeqSumProp\_Conclusion,\ Prop$

The sum of the delta vectors in the range of a function, all of which are upright, is an upright delta vector.

COROLLARY $DeltaVecUpright\_FunSum \triangleq$
ASSUME
NEW $leq \in PointRelationType$,
NEW D,
NEW $F \in [D \rightarrow DeltaVecType]$,
$IsPartialOrder(leq)$,
$DeltaVecFunHasFiniteNonZeroRange(F)$,
$\forall\, d \in D : IsDeltaVecUpright(leq,\ F[d])$
PROVE
$IsDeltaVecUpright(leq,\ DeltaVecFunSum(F))$

PROOF
$\langle 1 \rangle$ DEFINE $Prop(a) \triangleq IsDeltaVecUpright(leq, a)$
$\langle 1 \rangle$ USE $DeltaVecUpright\_Zero$
$\langle 1 \rangle$ USE $DeltaVecUpright\_Add$

$\langle 1 \rangle$ $DeltaVecFunSumProp\_Conclusion(Prop, F)$
$\langle 2 \rangle$ $DeltaVecFunSumProp\_Hypothesis(Prop, F)$
$\langle 3 \rangle$ QED BY DEF $DeltaVecFunSumProp\_Hypothesis$
$\langle 2 \rangle$ QED BY $DeltaVecFunSumProp$
$\langle 1 \rangle$ QED BY DEF $DeltaVecFunSumProp\_Conclusion, Prop$

C.12 Facts about beta-upright delta vectors

MODULE $NaiadClockProofDeltaVecBetaUpright$

EXTENDS $NaiadClockProofDeltaVecUpright$

Facts about beta-upright delta vectors.

Given a vb-upright delta vector va, then for every positive point t in va there is in va or vb some negative point $s \prec t$ such that va is nonpos up thru s. We call point s a vb-foundation for point t in va.

THEOREM $DeltaVecBetaUpright\_ExistsFoundation \triangleq$
  ASSUME
    NEW $leq \in PointRelationType$,
    NEW $va \in DeltaVecType$,
    NEW $vb \in DeltaVecType$,
    $IsPartialOrder(leq)$,
    $IsDeltaVecBetaUpright(leq, va, vb)$,
    NEW $t \in Point$, $va[t] > 0$
  PROVE
  LET
    $a \preceq b \triangleq leq[a][b]$
    $a \prec b \triangleq a \preceq b \land a \neq b$
  IN
  $\exists s \in Point :$
  $\land\ s \prec t$
  $\land\ va[s] < 0 \lor vb[s] < 0$
  $\land\ IsDeltaVecNonposUpto(leq, va, s)$
PROOF
  $\langle 1 \rangle$ QED BY DEF $IsDeltaVecBetaUpright$

A zero delta vector is vb-upright for any vb.

THEOREM $DeltaVecBetaUpright\_Zero \triangleq$
  ASSUME

NEW $leq \in PointRelationType$,
NEW $vb \in DeltaVecType$,
$IsPartialOrder(leq)$
PROVE
$IsDeltaVecBetaUpright(leq, DeltaVecZero, vb)$
PROOF
⟨1⟩ QED BY DEF $DeltaVecZero$, $IsDeltaVecBetaUpright$, Isa

If delta vector va is vb-upright, and vb and vc are upright delta vectors, then va is $(vb + vc)$-upright.

THEOREM $DeltaVecBetaUpright\_Add \triangleq$
ASSUME
NEW $leq \in PointRelationType$,
NEW $va \in DeltaVecType$,
NEW $vb \in DeltaVecType$,
NEW $vc \in DeltaVecType$,
$IsPartialOrder(leq)$,
$IsDeltaVecBetaUpright(leq, va, vb)$,
$IsDeltaVecUpright(leq, vb)$,
$IsDeltaVecUpright(leq, vc)$
PROVE
$IsDeltaVecBetaUpright(leq, va, DeltaVecAdd(vb, vc))$
PROOF
⟨1⟩ DEFINE $a \preceq b \triangleq leq[a][b]$
⟨1⟩ DEFINE $a \prec b \triangleq a \preceq b \land a \neq b$

Assume that t is a positive point in va and that there is no $(vb + vc)$-foundation for t in va. It suffices to show a contradiction.

⟨1⟩1. SUFFICES ASSUME
NEW $t \in Point$,
$va[t] > 0$,
$\neg\exists\, s \in Point :$
$\land\ s \prec t$
$\land\ va[s] < 0 \lor (vb[s] + vc[s] < 0)$
$\land\ \neg\exists\, u \in Point : u \preceq s \land va[u] > 0$
PROVE FALSE
⟨2⟩ USE DEF $DeltaVecAdd$
⟨2⟩ USE DEF $DeltaVecType$
⟨2⟩ USE DEF $IsDeltaVecBetaUpright$
⟨2⟩ USE DEF $IsDeltaVecNonposUpto$
⟨2⟩ QED BY $SMTT(10)$

Since va is vb-upright, we can pick x as a vb-foundation for t.

$\langle 1 \rangle 2.$ PICK $x \in Point$:
$\quad \wedge \; x \prec t$
$\quad \wedge \; va[x] < 0 \vee vb[x] < 0$
$\quad \wedge \; IsDeltaVecNonposUpto(leq, va, x)$
BY $\langle 1 \rangle 1$, $DeltaVecBetaUpright\_ExistsFoundation$

State what we know about x as separate facts.

$\langle 1 \rangle 3.$ $x \prec t$ BY $\langle 1 \rangle 2$
$\langle 1 \rangle 4.$ $va[x] < 0 \vee vb[x] < 0$ BY $\langle 1 \rangle 2$
$\langle 1 \rangle 5.$ $IsDeltaVecNonposUpto(leq, va, x)$ BY $\langle 1 \rangle 2$

If $va[x] < 0$ then x is a $(vb + vc)$-foundation for t in va. So this must not be. We deduce that $vb[x] < 0$.

$\langle 1 \rangle 6.$ $\neg(va[x] < 0)$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 3$, $\langle 1 \rangle 5$ DEF $IsDeltaVecNonposUpto$
$\langle 1 \rangle 7.$ $vb[x] < 0$ BY $\langle 1 \rangle 4$, $\langle 1 \rangle 6$

If $vb[x] + vc[x] < 0$ then x is a $(vb + vc)$-foundation for t in va. So this must not be. We deduce by arithmetic that $vc[x] > 0$.

$\langle 1 \rangle 8.$ $\neg(vb[x] + vc[x] < 0)$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 3$, $\langle 1 \rangle 5$ DEF $IsDeltaVecNonposUpto$
$\langle 1 \rangle 9.$ $vc[x] > 0$ BY $\langle 1 \rangle 7$, $\langle 1 \rangle 8$, $SMTT(10)$ DEF $DeltaVecAdd$, $DeltaVecType$

Since vc is upright, we can pick y as a support in vc for x.

$\langle 1 \rangle 10.$ PICK $y \in Point$:
$\quad \wedge \; y \prec x$
$\quad \wedge \; vc[y] < 0$
$\quad \wedge \; IsDeltaVecNonposUpto(leq, vc, y)$
BY $\langle 1 \rangle 9$, $DeltaVecUpright\_ExistsSupport$

State what we know about y as separate facts.

$\langle 1 \rangle 11.$ $y \prec x$ BY $\langle 1 \rangle 10$
$\langle 1 \rangle 12.$ $vc[y] < 0$ BY $\langle 1 \rangle 10$
$\langle 1 \rangle 13.$ $IsDeltaVecNonposUpto(leq, vc, y)$ BY $\langle 1 \rangle 10$

By transitivity, we almost have y as $(vb + vc)$-support for t in va.

$\langle 1 \rangle 14.$ $y \prec t$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 11$, $PartialOrderStrictlyTransitive$
$\langle 1 \rangle 15.$ ASSUME NEW $u \in Point$, $u \preceq y$, $va[u] > 0$ PROVE FALSE
$\quad \langle 2 \rangle$ $u \preceq x$ BY $\langle 1 \rangle 11$, $\langle 1 \rangle 15$, $PartialOrderTransitive$
$\quad \langle 2 \rangle$ QED BY $\langle 1 \rangle 5$, $\langle 1 \rangle 15$ DEF $IsDeltaVecNonposUpto$

If $vb[y] + vc[y] < 0$ then y is a $(vb + vc)$-support for t in va. So this must not be. We deduce by arithmetic that $vb[y] > 0$.

$\langle 1 \rangle 16.$ $\neg(vb[y] + vc[y] < 0)$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 14$, $\langle 1 \rangle 15$
$\langle 1 \rangle 17.$ $vb[y] > 0$ BY $\langle 1 \rangle 12$, $\langle 1 \rangle 16$, $SMTT(10)$ DEF $DeltaVecAdd$, $DeltaVecType$

Since vb is upright, we can pick z as a support in vb for y.

$\langle 1 \rangle 18.$ PICK $z \in Point$:
$\quad \wedge \; z \prec y$
$\quad \wedge \; vb[z] < 0$
$\quad \wedge \; IsDeltaVecNonposUpto(leq, vb, z)$
BY $\langle 1 \rangle 17$, $DeltaVecUpright\_ExistsSupport$

State what we know about z as separate facts.

$\langle 1 \rangle 19.\ z \prec y$ BY $\langle 1 \rangle 18$
$\langle 1 \rangle 20.\ vb[z] < 0$ BY $\langle 1 \rangle 18$
$\langle 1 \rangle 21.\ IsDeltaVecNonposUpto(leq,\ vb,\ z)$ BY $\langle 1 \rangle 18$

By transitivity, we almost have z as $(vb + vc)$-support for t in va.

$\langle 1 \rangle 22.\ z \prec t$ BY $\langle 1 \rangle 14,\ \langle 1 \rangle 19,\ PartialOrderStrictlyTransitive$
$\langle 1 \rangle 23.$ ASSUME NEW $u \in Point,\ u \preceq z,\ va[u] > 0$ PROVE FALSE
  $\langle 2 \rangle\ u \preceq y$ BY $\langle 1 \rangle 19,\ \langle 1 \rangle 23,\ PartialOrderTransitive$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 15,\ \langle 1 \rangle 23$

If $vb[z] + vc[z] < 0$ then z is a $(vb + vc)$-support for t in va. So this must not be. We deduce by arithmetic that $vc[z] > 0$.

$\langle 1 \rangle 24.\ \neg(vb[z] + vc[z] < 0)$ BY $\langle 1 \rangle 1,\ \langle 1 \rangle 22,\ \langle 1 \rangle 23$
$\langle 1 \rangle 25.\ vc[z] > 0$ BY $\langle 1 \rangle 20,\ \langle 1 \rangle 24,\ SMTT(10)$ DEF $DeltaVecAdd,\ DeltaVecType$

But $z \prec y$ and y is a support in vc, so we cannot have $vc[z] > 0$. This completes the proof.

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 13,\ \langle 1 \rangle 19,\ \langle 1 \rangle 25$ DEF $IsDeltaVecNonposUpto$

Given F mapping to upright delta vectors with a finite non-zero range, if delta vector va is $F[d]$-upright for some d, then va is also $Sum(F)$-upright.

$DeltaVecBetaUpright\_FunSum\_Hypothesis(leq,\ F,\ va,\ d0) \triangleq$
  LET
    $D \triangleq$ DOMAIN F
  IN
  $\land\ leq \in PointRelationType$
  $\land\ IsPartialOrder(leq)$
  $\land\ F \in [D \rightarrow DeltaVecType]$
  $\land\ va \in DeltaVecType$
  $\land\ DeltaVecFunHasFiniteNonZeroRange(F)$
  $\land\ \forall\, d \in D : IsDeltaVecUpright(leq,\ F[d])$
  $\land\ d0 \in D$
  $\land\ IsDeltaVecBetaUpright(leq,\ va,\ F[d0])$

THEOREM $DeltaVecBetaUpright\_FunSum \triangleq$
  ASSUME
    NEW leq,
    NEW F,
    NEW va,
    NEW $d0$,
    $DeltaVecBetaUpright\_FunSum\_Hypothesis(leq,\ F,\ va,\ d0)$

PROVE
$IsDeltaVecBetaUpright(leq, va, DeltaVecFunSum(F))$
PROOF
$\langle 1 \rangle$ USE DEF $DeltaVecBetaUpright\_FunSum\_Hypothesis$
$\langle 1 \rangle$ DEFINE $D \triangleq$ DOMAIN F
$\langle 1 \rangle$ $leq \in PointRelationType$ OBVIOUS
$\langle 1 \rangle$ $IsPartialOrder(leq)$ OBVIOUS
$\langle 1 \rangle$ $F \in [D \rightarrow DeltaVecType]$ OBVIOUS
$\langle 1 \rangle$ $va \in DeltaVecType$ OBVIOUS
$\langle 1 \rangle$ $DeltaVecFunHasFiniteNonZeroRange(F)$ OBVIOUS
$\langle 1 \rangle$ $\forall d \in D : IsDeltaVecUpright(leq, F[d])$ OBVIOUS
$\langle 1 \rangle$ $d0 \in D$ OBVIOUS
$\langle 1 \rangle 1.$ $IsDeltaVecBetaUpright(leq, va, F[d0])$ OBVIOUS
$\langle 1 \rangle$ HIDE DEF $DeltaVecBetaUpright\_FunSum\_Hypothesis$

$\langle 1 \rangle$ DEFINE $G \triangleq [F$ EXCEPT $![d0] = DeltaVecZero]$
$\langle 1 \rangle$ DEFINE $SumF \triangleq DeltaVecFunSum(F)$
$\langle 1 \rangle$ DEFINE $SumG \triangleq DeltaVecFunSum(G)$

$\langle 1 \rangle 2.$ $G \in [D \rightarrow DeltaVecType]$ BY $DeltaVecZeroType$
$\langle 1 \rangle 3.$ $DeltaVecFunHasFiniteNonZeroRange(G)$
$\langle 2 \rangle$ DEFINE $Fnz \triangleq \{d \in D : F[d] \neq DeltaVecZero\}$
$\langle 2 \rangle$ DEFINE $Gnz \triangleq \{d \in D : G[d] \neq DeltaVecZero\}$
$\langle 2 \rangle 1.$ $IsFiniteSet(Fnz)$ BY DEF $DeltaVecFunHasFiniteNonZeroRange$
$\langle 2 \rangle 2.$ $Gnz \subseteq Fnz$ OBVIOUS
$\langle 2 \rangle 3.$ $IsFiniteSet(Gnz)$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $FiniteSetSubset$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 3$ DEF $DeltaVecFunHasFiniteNonZeroRange$

$\langle 1 \rangle 4.$ $SumG \in DeltaVecType$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $DeltaVecFunSumType$
$\langle 1 \rangle 5.$ $\forall d \in D : IsDeltaVecUpright(leq, G[d])$ BY $DeltaVecUpright\_Zero$
$\langle 1 \rangle 6.$ $IsDeltaVecUpright(leq, SumG)$
BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\langle 1 \rangle 5$, $DeltaVecUpright\_FunSum$

$\langle 1 \rangle 7.$ $F = DeltaVecFunAddAt(G, d0, F[d0])$
$\langle 2 \rangle 1.$ $F[d0] = DeltaVecAdd(DeltaVecZero, F[d0])$ BY $DeltaVecAddZero$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$ DEF $DeltaVecFunAddAt$

$\langle 1 \rangle 8.$ $SumF = DeltaVecAdd(F[d0], SumG)$
$\langle 2 \rangle$ HIDE DEF G
$\langle 2 \rangle 1.$ $DeltaVecFunSumAddAt\_Hypothesis(G, d0, F[d0])$
BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$ DEF $DeltaVecFunSumAddAt\_Hypothesis$
$\langle 2 \rangle 2.$ $DeltaVecFunSumAddAt\_Conclusion(G, d0, F[d0])$
BY $\langle 2 \rangle 1$, $DeltaVecFunSumAddAt$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 1 \rangle 7$ DEF $DeltaVecFunSumAddAt\_Conclusion$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 4$, $\langle 1 \rangle 6$, $\langle 1 \rangle 8$, $DeltaVecBetaUpright\_Add$

If we have delta vectors va and vb such that $va + vb$ is upright and va positive implies $va + vb$, then va is vb-upright.

THEOREM $DeltaVecBetaUpright\_PositiveImplies \triangleq$
ASSUME
NEW $leq \in PointRelationType$,
NEW $va \in DeltaVecType$,
NEW $vb \in DeltaVecType$,
$IsPartialOrder(leq)$,
$IsDeltaVecUpright(leq, DeltaVecAdd(va, vb))$,
$IsDeltaVecPositiveImplies(va, DeltaVecAdd(va, vb))$
PROVE
$IsDeltaVecBetaUpright(leq, va, vb)$
PROOF
$\langle 1 \rangle$ DEFINE $a \preceq b \triangleq leq[a][b]$
$\langle 1 \rangle$ DEFINE $a \prec b \triangleq a \preceq b \land a \neq b$

$\langle 1 \rangle 1.$ PICK $v0 \in DeltaVecType : v0 = DeltaVecAdd(va, vb)$ BY $DeltaVecAddType$

Assume that t is a positive point in va. It suffices to show that there is a vb-foundation for t in va.

$\langle 1 \rangle 2.$ SUFFICES ASSUME
NEW $t \in Point$,
$va[t] \quad > 0$
PROVE
$\exists\, s \in Point :$
$\land\ s \prec t$
$\land\ va[s] < 0 \lor vb[s] < 0$
$\land\ IsDeltaVecNonposUpto(leq, va, s)$
BY DEF $IsDeltaVecBetaUpright$

$\langle 1 \rangle 3.$ $v0[t] > 0$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$ DEF $IsDeltaVecPositiveImplies$

Since $v0$ is upright, we can pick x as a support in $v0$ for t.

$\langle 1 \rangle 4.$ PICK $x \in Point :$
$\land\ \ x \prec t$
$\land\ \ v0[x] < 0$
$\land\ \ IsDeltaVecNonposUpto(leq, v0, x)$
BY $\langle 1 \rangle 1$, $\langle 1 \rangle 3$, $DeltaVecUpright\_ExistsSupport$

State what we know about x as separate facts.

$\langle 1 \rangle 5.$ $x \prec t$ BY $\langle 1 \rangle 4$
$\langle 1 \rangle 6.$ $v0[x] < 0$ BY $\langle 1 \rangle 4$
$\langle 1 \rangle 7.$ $IsDeltaVecNonposUpto(leq, v0, x)$ BY $\langle 1 \rangle 4$

Deduce that x is a vb-foundation for t in va.

$\langle 1 \rangle 8.$ $va[x] < 0 \lor vb[x] < 0$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 6$, $SMTT(10)$ DEF $DeltaVecAdd$, $DeltaVecType$
$\langle 1 \rangle 9.$ ASSUME NEW $u \in Point$, $u \preceq x$, $va[u] > 0$ PROVE FALSE
  $\langle 2 \rangle 1.$ $\neg(v0[u] > 0)$ BY $\langle 1 \rangle 7$, $\langle 1 \rangle 9$ DEF $IsDeltaVecNonposUpto$
  $\langle 2 \rangle 2.$ $v0[u] > 0$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 9$ DEF $IsDeltaVecPositiveImplies$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 5$, $\langle 1 \rangle 8$, $\langle 1 \rangle 9$ DEF $IsDeltaVecNonposUpto$

C.13 Facts about delta vectors vacant up to point t

MODULE *NaiadClockProofDeltaVecVacantUpto*

EXTENDS *NaiadClockProofDeltaVecBetaUpright*

Facts about delta vectors vacant up to point t.

Given delta vectors va, vb, and vc such that $vc = va + vb$, if any two of these delta vectors are vacant up to point t then the third one is also.

THEOREM $DeltaVecVacantUpto\_Add \triangleq$
  ASSUME
    NEW $leq \in PointRelationType$,
    NEW $va \in DeltaVecType$,
    NEW $vb \in DeltaVecType$,
    NEW $t \in Point$
  PROVE
  LET
    $vc \triangleq DeltaVecAdd(va, vb)$
    $VUT(v) \triangleq IsDeltaVecVacantUpto(leq, v, t)$
  IN
  $\land\ VUT(va) \land VUT(vb) \Rightarrow VUT(vc)$
  $\land\ VUT(vb) \land VUT(vc) \Rightarrow VUT(va)$
  $\land\ VUT(vc) \land VUT(va) \Rightarrow VUT(vb)$

PROOF
  $\langle 1 \rangle$ DEFINE $vc \triangleq DeltaVecAdd(va, vb)$
  $\langle 1 \rangle$ DEFINE $VUT(v) \triangleq IsDeltaVecVacantUpto(leq, v, t)$

  $\langle 1 \rangle$ USE DEF $DeltaVecAdd$
  $\langle 1 \rangle$ USE DEF $DeltaVecType$
  $\langle 1 \rangle$ USE DEF $IsDeltaVecVacantUpto$

  $\langle 1 \rangle 1.$ ASSUME $VUT(va), VUT(vb)$ PROVE $VUT(vc)$ BY $\langle 1 \rangle 1, SMTT(10)$
  $\langle 1 \rangle 2.$ ASSUME $VUT(vb), VUT(vc)$ PROVE $VUT(va)$ BY $\langle 1 \rangle 2, SMTT(10)$
  $\langle 1 \rangle 3.$ ASSUME $VUT(vc), VUT(va)$ PROVE $VUT(vb)$ BY $\langle 1 \rangle 3, SMTT(10)$

  $\langle 1 \rangle$ QED BY $\langle 1 \rangle 1, \langle 1 \rangle 2, \langle 1 \rangle 3$

If we have a delta vectors va and vb such that
(1) va is vb-upright,
(2) vb is upright, and
(3) the sum $va + vb$ is vacant up to point t,
then we can conclude that va is vacant up to point t.

THEOREM $DeltaVecVacantUpto\_BetaUpright \triangleq$
ASSUME
NEW $leq \in PointRelationType$,
NEW $va \in DeltaVecType$,
NEW $vb \in DeltaVecType$,
NEW $t \in Point$,
$IsPartialOrder(leq)$,
$IsDeltaVecBetaUpright(leq, va, vb)$,
$IsDeltaVecUpright(leq, vb)$,
$IsDeltaVecVacantUpto(leq, DeltaVecAdd(va, vb), t)$
PROVE
$IsDeltaVecVacantUpto(leq, va, t)$
PROOF
$\langle 1 \rangle$ DEFINE $a \preceq b \triangleq leq[a][b]$
$\langle 1 \rangle$ DEFINE $a \prec b \triangleq a \preceq b \wedge a \neq b$

$\langle 1 \rangle 1.$ $\forall s \in Point :$
$\wedge\ va[s] \in Int$
$\wedge\ vb[s] \in Int$
$\wedge\ s \preceq t \Rightarrow va[s] + vb[s] = 0$
BY DEF $IsDeltaVecVacantUpto$, $DeltaVecAdd$, $DeltaVecType$

LEMMA : If we can find a point $s \preceq t$ where $va[s] > 0$, we can produce a contradiction.

$\langle 1 \rangle 2.$ ASSUME
NEW $s \in Point$,
$s \preceq t$,
$va[s] > 0$
PROVE FALSE

State what we know about s as separate facts.

$\langle 2 \rangle 1.$ $s \preceq t$ BY $\langle 1 \rangle 2$
$\langle 2 \rangle 2.$ $va[s] > 0$ BY $\langle 1 \rangle 2$

Since va is vb-upright, let x be a vb-foundation for s in va.

$\langle 2 \rangle 3.$ PICK $x \in Point :$
$\wedge\ x \prec s$
$\wedge\ va[x] < 0 \vee vb[x] < 0$
$\wedge\ IsDeltaVecNonposUpto(leq, va, x)$
BY $\langle 2 \rangle 2$, $DeltaVecBetaUpright\_ExistsFoundation$

State what we know about x as separate facts.

$\langle 2\rangle 4.\ x \prec s$ BY $\langle 2\rangle 3$
$\langle 2\rangle 5.\ va[x] < 0 \lor vb[x] < 0$ BY $\langle 2\rangle 3$
$\langle 2\rangle 6.\ IsDeltaVecNonposUpto(leq,\ va,\ x)$ BY $\langle 2\rangle 3$

$\langle 2\rangle 7.$ CASE $vb[x] < 0$

We have $va[x] > 0$ since $va + vb$ is vacant up to point t.

$\langle 3\rangle 1.\ x \preceq t$ BY $\langle 2\rangle 1,\ \langle 2\rangle 4,\ PartialOrderTransitive$
$\langle 3\rangle 2.\ va[x] > 0$ BY $\langle 3\rangle 1,\ \langle 2\rangle 7,\ \langle 1\rangle 1,\ SMTT(10)$

But this contradicts the choice of x.

$\langle 3\rangle 3.\ x \preceq x$ BY $PartialOrderReflexive$
$\langle 3\rangle$ QED BY $\langle 3\rangle 2,\ \langle 3\rangle 3,\ \langle 2\rangle 6$ DEF $IsDeltaVecNonposUpto$

$\langle 2\rangle 8.$ CASE $va[x] < 0$

We have $vb[x] > 0$ since $va + vb$ is vacant up to point t.

$\langle 3\rangle 1.\ x \preceq t$ BY $\langle 2\rangle 1,\ \langle 2\rangle 4,\ PartialOrderTransitive$
$\langle 3\rangle 2.\ vb[x] > 0$ BY $\langle 3\rangle 1,\ \langle 2\rangle 8,\ \langle 1\rangle 1,\ SMTT(10)$

Since vb is upright, let y be a support for x in vb.

$\langle 3\rangle 3.$ PICK $y \in Point$:
$\land\ \ y \prec x$
$\land\ \ vb[y] < 0$
$\land\ \ IsDeltaVecNonposUpto(leq,\ vb,\ y)$
BY $\langle 3\rangle 2,\ DeltaVecUpright\_ExistsSupport$

State what we know about y as separate facts.

$\langle 3\rangle 4.\ y \prec x$ BY $\langle 3\rangle 3$
$\langle 3\rangle 5.\ vb[y] < 0$ BY $\langle 3\rangle 3$
$\langle 3\rangle 6.\ IsDeltaVecNonposUpto(leq,\ vb,\ y)$ BY $\langle 3\rangle 3$

We have $va[y] > 0$ since $va + vb$ is vacant up to point t.

$\langle 3\rangle 7.\ y \preceq t$ BY $\langle 3\rangle 1,\ \langle 3\rangle 4,\ PartialOrderTransitive$
$\langle 3\rangle 8.\ va[y] > 0$ BY $\langle 3\rangle 5,\ \langle 3\rangle 7,\ \langle 1\rangle 1,\ SMTT(10)$

But this contradicts the choice of x.

$\langle 3\rangle$ QED BY $\langle 3\rangle 4,\ \langle 3\rangle 8,\ \langle 2\rangle 6$ DEF $IsDeltaVecNonposUpto$

$\langle 2\rangle$ QED BY $\langle 2\rangle 3,\ \langle 2\rangle 7,\ \langle 2\rangle 8$

So let us assume that the conclusion is false and then derive a contradiction. If the conclusion is false, then there must be some point $s \preceq t$ such that $va[s] \neq 0$.

$\langle 1\rangle 3.$ SUFFICES ASSUME
NEW $s \in Point$,
$s \preceq t$,
$va[s] \neq 0$
PROVE FALSE
BY DEF $IsDeltaVecVacantUpto$

State what we know about s as separate facts.

$\langle 1\rangle 4.\ s \preceq t$ BY $\langle 1\rangle 3$

$\langle 1 \rangle 5.\ va[s] \neq 0$ BY $\langle 1 \rangle 3$

So we have two cases: either $va[s] > 0$ or $va[s] < 0$. In either case, we find a point $\preceq t$ where va is positive. This produces a contradiction by our lemma.

$\langle 1 \rangle 6.$ CASE $va[s] > 0$ BY $\langle 1 \rangle 2,\ \langle 1 \rangle 4,\ \langle 1 \rangle 6$

$\langle 1 \rangle 7.$ CASE $va[s] < 0$

We have $vb[s] > 0$ since $va + vb$ is vacant up to point t.

$\langle 2 \rangle 1.\ vb[s] > 0$ BY $\langle 1 \rangle 1,\ \langle 1 \rangle 4,\ \langle 1 \rangle 7,\ SMTT(10)$

Since vb is upright, let x be a support for s in vb.

$\langle 2 \rangle 2.$ PICK $x \in Point$:
$\land\ x \prec s$
$\land\ vb[x] < 0$
$\land\ IsDeltaVecNonposUpto(leq,\ vb,\ x)$
BY $\langle 2 \rangle 1,\ DeltaVecUpright\_ExistsSupport$

State what we know about x as separate facts.

$\langle 2 \rangle 3.\ x \prec s$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 4.\ vb[x] < 0$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 5.\ IsDeltaVecNonposUpto(leq,\ vb,\ x)$ BY $\langle 2 \rangle 2$

We have $va[x] > 0$ since $va + vb$ is vacant up to point t.

$\langle 2 \rangle 6.\ x \preceq t$ BY $\langle 2 \rangle 3,\ \langle 1 \rangle 4,\ PartialOrderTransitive$
$\langle 2 \rangle 7.\ va[x] > 0$ BY $\langle 2 \rangle 4,\ \langle 2 \rangle 6,\ \langle 1 \rangle 1,\ SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 2,\ \langle 2 \rangle 6,\ \langle 2 \rangle 7$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1,\ \langle 1 \rangle 4,\ \langle 1 \rangle 5,\ \langle 1 \rangle 6,\ \langle 1 \rangle 7,\ SMTT(10)$

C.14 Additional invariants needed in the proof

MODULE $NaiadClockProofInvariants$

EXTENDS $NaiadClockProofDeltaVecVacantUpto$

Additional invariants needed in the proof.

For every skip count k, sending processor p, and receiving processor q, $InfoAt(k,\ p,\ q)$ is a delta vector.

$$
\begin{array}{l}
InvInfoAtType \triangleq \\
\quad \forall\, k \in Nat : \\
\quad \forall\, p \in Proc : \\
\quad \forall\, q \in Proc : \\
\quad \text{LET} \\
\qquad M \quad\quad\;\; \triangleq msg[p][q] \\
\qquad LenM \quad \triangleq Len(M) \\
\qquad InRange \triangleq \\
\qquad\quad \lor\, k \in \text{DOMAIN } M \\
\qquad\quad \lor\, k \in 1 \,..\, LenM \\
\qquad\quad \lor\, k \neq 0 \land k \leq LenM \\
\qquad\quad \lor\, 0 < k \land k \leq LenM \\
\quad \text{IN} \\
\quad\; \land\, InfoAt(k,\ p,\ q) \in DeltaVecType \\
\quad\; \land\, k = 0 \Rightarrow InfoAt(k,\ p,\ q) = DeltaVecZero \\
\quad\; \land\, LenM < k \Rightarrow InfoAt(k,\ p,\ q) = DeltaVecZero \\
\quad\; \land\, InRange \Rightarrow InfoAt(k,\ p,\ q) = M[k]
\end{array}
$$

For every skip count k, sending processor p, and receiving processor q, $IncomingInfo(k,\ p,\ q)$ is a delta vector.

$$
\begin{array}{l}
InvIncomingInfoType \triangleq \\
\quad \forall\, k \in Nat : \\
\quad \forall\, p \in Proc : \\
\quad \forall\, q \in Proc : \\
\quad \text{LET} \\
\qquad sum \quad \triangleq IncomingInfo(k,\ p,\ q)! : !sum \\
\quad \text{IN}
\end{array}
$$

$\land\ sum \in DeltaVecType$
$\land\ IncomingInfo(k,\ p,\ q) \in DeltaVecType$

For every skip count k, sending processor p, and receiving processor q, $GlobalIncomingInfo(k,\ p,\ q)$ is a delta vector.

$$
\begin{array}{l}
InvGlobalIncomingInfoType \ \triangleq \\
\quad \forall\, k \in Nat : \\
\quad \forall\, p \in Proc : \\
\quad \forall\, q \in Proc : \\
\quad \text{LET} \\
\qquad F \ \triangleq\ GlobalIncomingInfo\_F(k,\ p,\ q) \\
\quad \text{IN} \\
\quad \land\ F \in [Proc \rightarrow DeltaVecType] \\
\quad \land\ DeltaVecFunHasFiniteNonZeroRange(F) \\
\quad \land\ GlobalIncomingInfo(k,\ p,\ q) \in DeltaVecType
\end{array}
$$

$GlobalIncomingInfo(0,\ p,\ q)$ is the same regardless of p.

$$
\begin{array}{l}
InvGlobalIncomingInfoSkip0 \ \triangleq \\
\quad \forall\, p1 \in Proc : \\
\quad \forall\, p2 \in Proc : \\
\quad \forall\, q \in Proc : \\
\quad GlobalIncomingInfo(0,\ p1,\ q) = GlobalIncomingInfo(0,\ p2,\ q)
\end{array}
$$

For every skip count k, sending processor p, and receiving processor q, the $InfoAt(k,\ p,\ q)$ is $IncomingInfo(k,\ p,\ q)$-upright.

Note that $InfoAt(k,\ p,\ q)$ is an item of incoming information on the message queue from processor p to processor q. $IncomingInfo(k,\ p,\ q)$ is the sum of all subsequent incoming information on that message queue plus all information in $temp[p]$ that has not yet been sent.

$$
\begin{array}{l}
InvInfoAtBetaUpright \ \triangleq \\
\quad \forall\, k \in Nat : \\
\quad \forall\, p \in Proc : \\
\quad \forall\, q \in Proc : \\
\quad IsDeltaVecBetaUpright(lleq,\ InfoAt(k,\ p,\ q),\ IncomingInfo(k,\ p,\ q))
\end{array}
$$

For every skip count k, sending processor p, and receiving processor q, the $InfoAt(k,\ p,\ q)$ is $GlobalIncomingInfo(k,\ p,\ q)$-upright.

Note that $InfoAt(k,\ p,\ q)$ is an item of incoming information on the message queue from processor p to processor q.

$GlobalIncomingInfo(k,\ p,\ q)$ is the sum of all incoming information to processor q except for skipping the first k messages coming from processor p.

$$
\begin{array}{l}
InvGlobalInfoAtBetaUpright \triangleq \\
\quad \forall\, k \in Nat : \\
\quad \forall\, p \in Proc : \\
\quad \forall\, q \in Proc : \\
\quad IsDeltaVecBetaUpright(lleq,\ InfoAt(k,\ p,\ q),\ GlobalIncomingInfo(k,\ p,\ q))
\end{array}
$$

C.15 Deduce various invariants from others

MODULE $NaiadClockProofDeduceInv$

EXTENDS $NaiadClockProofInvariants$

Deduce various invariants from others.

We prove these deductions in both the current state (unprimed) and the next state (primed). It is the exact same proof each way so we prove both ways at once using $goal^{\#}$ to represent both instances. There is no deduction rule that permits you to deduce the primed version from the unprimed version since, in general, such a rule would be unsound.

The invariant $InvInfoAtType$ follows from $InvType$.

THEOREM $DeduceInvInfoAtType \triangleq$
  LET $goal \triangleq$
    $InvType$
    $\Rightarrow$
    $InvInfoAtType$
  IN
  $goal \land goal'$
PROOF
  ⟨1⟩ DEFINE $goal \triangleq DeduceInvInfoAtType\,!\,:\,!\,goal$
  ⟨1⟩ DEFINE $DoPr(primeit,\, x) \triangleq$ IF $primeit$ THEN x' ELSE x

  ⟨1⟩ SUFFICES ASSUME NEW $primeit \in$ BOOLEAN PROVE $DoPr(primeit,\, goal)$ OBVIOUS

  ⟨1⟩ DEFINE $x^{\#} \triangleq DoPr(primeit,\, x)$

  ⟨1⟩1. SUFFICES ASSUME $InvType^{\#}$ PROVE $InvInfoAtType^{\#}$ OBVIOUS

  ⟨1⟩ DEFINE $I(k,\, p,\, q) \triangleq InvInfoAtType\,!(k)\,!(p)\,!(q)$
  ⟨1⟩ HIDE DEF I

  ⟨1⟩ SUFFICES ASSUME NEW $k \in Nat$, NEW $p \in Proc$, NEW $q \in Proc$ PROVE $I(k,\, p,\, q)^{\#}$
    The prover needs help to distribute $DoPr$ through quantifiers.
    ⟨2⟩ $\forall\, k \in Nat : \forall\, p \in Proc : \forall\, q \in Proc : I(k,\, p,\, q)^{\#}$ BY DEF I
    ⟨2⟩ $(\forall\, k \in Nat : \forall\, p \in Proc : \forall\, q \in Proc : I(k,\, p,\, q))^{\#}$ OBVIOUS
    ⟨2⟩ QED BY IsaDEF $InvInfoAtType,\, I$

  ⟨1⟩ DEFINE $M \triangleq msg[p][q]$
  ⟨1⟩ DEFINE $LenM \triangleq Len(M)$
  ⟨1⟩ HIDE DEF $M,\, LenM$

$\langle 1 \rangle 2.\ M^{\#} \in Seq(DeltaVecType)$ BY $\langle 1 \rangle 1$ DEF $InvType,\ M$
$\langle 1 \rangle 4.\ LenM^{\#} \in Nat$ BY $\langle 1 \rangle 2,\ LenInNat$ DEF $LenM$
$\langle 1 \rangle 7.$ DOMAIN $M^{\#} = 1 \,..\, LenM^{\#}$ BY $\langle 1 \rangle 2,\ LenDef$ DEF $LenM$

$\langle 1 \rangle$ DEFINE $InRange \triangleq$
$\quad \lor k \in$ DOMAIN $M^{\#}$
$\quad \lor k \in 1 \,..\, LenM^{\#}$
$\quad \lor k \neq 0 \land k \leq LenM^{\#}$
$\quad \lor 0 < k \land k \leq LenM^{\#}$

$\langle 1 \rangle 8.\ InRange \Rightarrow 0 < k \land k \leq LenM^{\#}$ BY $\langle 1 \rangle 4,\ \langle 1 \rangle 7,\ SMTT(10)$

$\langle 1 \rangle 9.$ CASE $k = 0$
$\langle 2 \rangle 1.\ \neg(0 < k \land k \leq LenM^{\#})$ BY $\langle 1 \rangle 4,\ \langle 1 \rangle 9,\ SMTT(10)$
$\langle 2 \rangle 2.\ InfoAt(k,\ p,\ q)^{\#} = DeltaVecZero$ BY $\langle 2 \rangle 1$ DEF $InfoAt,\ LenM,\ M$
$\langle 2 \rangle 3.\ InfoAt(k,\ p,\ q)^{\#} \in DeltaVecType$ BY $\langle 2 \rangle 2,\ DeltaVecZeroType$
$\langle 2 \rangle 4.\ \neg(LenM^{\#} < k)$ BY $\langle 1 \rangle 4,\ \langle 1 \rangle 9,\ SMTT(10)$
$\langle 2 \rangle 5.\ \neg InRange$BY $\langle 2 \rangle 1,\ \langle 1 \rangle 8$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2,\ \langle 2 \rangle 3,\ \langle 2 \rangle 4,\ \langle 2 \rangle 5,\ \langle 1 \rangle 9$ DEF $M,\ LenM,\ I$

$\langle 1 \rangle 10.$ CASE $0 < k \land k \leq LenM^{\#}$
$\langle 2 \rangle 1.\ InfoAt(k,\ p,\ q)^{\#} = M[k]^{\#}$ BY $\langle 1 \rangle 10$ DEF $InfoAt,\ LenM,\ M$
$\langle 2 \rangle 2.\ k \in 1 \,..\, LenM^{\#}$ BY $\langle 1 \rangle 4,\ \langle 1 \rangle 10,\ SMTT(10)$
$\langle 2 \rangle 3.\ M[k]^{\#} \in DeltaVecType$ BY $\langle 2 \rangle 2,\ \langle 1 \rangle 2,\ ElementOfSeq$ DEF $LenM$
$\langle 2 \rangle 4.\ k \neq 0$BY $\langle 1 \rangle 4,\ \langle 1 \rangle 10,\ SMTT(10)$
$\langle 2 \rangle 5.\ \neg(LenM^{\#} < k)$ BY $\langle 1 \rangle 4,\ \langle 1 \rangle 10,\ SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 3,\ \langle 2 \rangle 4,\ \langle 2 \rangle 5$ DEF $M,\ LenM,\ I$

$\langle 1 \rangle 11.$ CASE $LenM^{\#} < k$
$\langle 2 \rangle 1.\ \neg(0 < k \land k \leq LenM^{\#})$ BY $\langle 1 \rangle 4,\ \langle 1 \rangle 11,\ SMTT(10)$
$\langle 2 \rangle 2.\ InfoAt(k,\ p,\ q)^{\#} = DeltaVecZero$ BY $\langle 2 \rangle 1$ DEF $InfoAt,\ LenM,\ M$
$\langle 2 \rangle 3.\ InfoAt(k,\ p,\ q)^{\#} \in DeltaVecType$ BY $\langle 2 \rangle 2,\ DeltaVecZeroType$
$\langle 2 \rangle 4.\ k \neq 0$ BY $\langle 1 \rangle 4,\ \langle 1 \rangle 11,\ SMTT(10)$
$\langle 2 \rangle 5.\ LenM^{\#} < k$ BY $\langle 1 \rangle 11$
$\langle 2 \rangle 6.\ \neg InRange$BY $\langle 2 \rangle 1,\ \langle 1 \rangle 8$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2,\ \langle 2 \rangle 3,\ \langle 2 \rangle 4,\ \langle 2 \rangle 5,\ \langle 2 \rangle 6$ DEF $M,\ LenM,\ I$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 4,\ \langle 1 \rangle 9,\ \langle 1 \rangle 10,\ \langle 1 \rangle 11,\ SMTT(10)$

The invariant $InvIncomingInfoType$ follows from $InvType$.

THEOREM $DeduceInvIncomingInfoType \triangleq$
LET $goal \triangleq$

$InvType$
$\Rightarrow$
$InvIncomingInfoType$
IN
$goal \land goal'$
PROOF
$\langle 1 \rangle$ DEFINE $goal \triangleq DeduceInvIncomingInfoType!:!goal$
$\langle 1 \rangle$ DEFINE $DoPr(primeit, x) \triangleq$ IF $primeit$ THEN x' ELSE x

$\langle 1 \rangle$ SUFFICES ASSUME NEW $primeit \in$ BOOLEAN PROVE $DoPr(primeit, goal)$ OBVIOUS
$\langle 1 \rangle$ DEFINE $x^{\#} \triangleq DoPr(primeit, x)$

$\langle 1 \rangle 1.$ SUFFICES ASSUME $InvType^{\#}$ PROVE $InvIncomingInfoType^{\#}$ OBVIOUS

$\langle 1 \rangle$ SUFFICES ASSUME
NEW $k \in Nat$,
NEW $p \in Proc$,
NEW $q \in Proc$
PROVE $InvIncomingInfoType!(k)!(p)!(q)^{\#}$

The prover needs help to distribute $DoPr$ through quantifiers.

$\langle 2 \rangle$ DEFINE $I(k, p, q) \triangleq InvIncomingInfoType!(k)!(p)!(q)$
$\langle 2 \rangle$ HIDE DEF I
$\langle 2 \rangle$ $\forall k \in Nat : \forall p \in Proc : \forall q \in Proc : I(k, p, q)^{\#}$ BY DEF I
$\langle 2 \rangle$ $(\forall k \in Nat : \forall p \in Proc : \forall q \in Proc : I(k, p, q))^{\#}$ OBVIOUS
$\langle 2 \rangle$ QED BY Isa DEF $InvIncomingInfoType, I$

$\langle 1 \rangle$ DEFINE $tempp \triangleq temp[p]$
$\langle 1 \rangle$ DEFINE $msgpq \triangleq msg[p][q]$
$\langle 1 \rangle$ DEFINE $sum \triangleq IncomingInfo(k, p, q)!:!sum$

$\langle 1 \rangle 2.$ $tempp^{\#} \in DeltaVecType$ BY $\langle 1 \rangle 1$ DEF $InvType$
$\langle 1 \rangle 3.$ $msgpq^{\#} \in Seq(DeltaVecType)$ BY $\langle 1 \rangle 1$ DEF $InvType$
$\langle 1 \rangle 4.$ $sum^{\#} \in DeltaVecType$ BY $\langle 1 \rangle 3$, $DeltaVecSeqSkipSumType$
$\langle 1 \rangle 5.$ $IncomingInfo(k, p, q)^{\#} \in DeltaVecType$
BY $\langle 1 \rangle 2$, $\langle 1 \rangle 4$, $DeltaVecAddType$ DEF $IncomingInfo$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 4$, $\langle 1 \rangle 5$

The invariant $InvGlobalIncomingInfoType$ follows from $InvType$.

THEOREM $DeduceInvGlobalIncomingInfoType \triangleq$
LET $goal \triangleq$

$InvType$
$\Rightarrow$
$InvGlobalIncomingInfoType$
IN
$goal \wedge goal'$
PROOF
$\langle 1 \rangle$ DEFINE $goal \triangleq DeduceInvGlobalIncomingInfoType\,!\,:\,!\,goal$
$\langle 1 \rangle$ DEFINE $DoPr(primeit,\, x) \triangleq$ IF $primeit$ THEN x' ELSE x

$\langle 1 \rangle$ SUFFICES ASSUME NEW $primeit \in$ BOOLEAN PROVE $DoPr(primeit,\, goal)$ OBVIOUS
$\langle 1 \rangle$ DEFINE $x^{\#} \triangleq DoPr(primeit,\, x)$

$\langle 1 \rangle 1.$ SUFFICES ASSUME $InvType^{\#}$ PROVE $InvGlobalIncomingInfoType^{\#}$ OBVIOUS
$\langle 1 \rangle 2.$ $InvIncomingInfoType^{\#}$ BY $\langle 1 \rangle 1,\, DeduceInvIncomingInfoType$

$\langle 1 \rangle$ SUFFICES ASSUME
NEW $k \in Nat$,
NEW $p \in Proc$,
NEW $q \in Proc$
PROVE $InvGlobalIncomingInfoType\,!(k)\,!(p)\,!(q)^{\#}$

The prover needs help to distribute $DoPr$ through quantifiers.

$\langle 2 \rangle$ DEFINE $I(k,\, p,\, q) \triangleq InvGlobalIncomingInfoType\,!(k)\,!(p)\,!(q)$
$\langle 2 \rangle$ HIDE DEF I
$\langle 2 \rangle$ $\forall\, k \in Nat : \forall\, p \in Proc : \forall\, q \in Proc : I(k,\, p,\, q)^{\#}$ BY DEF I
$\langle 2 \rangle$ $(\forall\, k \in Nat : \forall\, p \in Proc : \forall\, q \in Proc : I(k,\, p,\, q))^{\#}$ OBVIOUS
$\langle 2 \rangle$ QED BY Isa DEF $InvGlobalIncomingInfoType,\, I$

$\langle 1 \rangle$ DEFINE $GII \triangleq GlobalIncomingInfo(k,\, p,\, q)$
$\langle 1 \rangle$ DEFINE $F \triangleq GlobalIncomingInfo(k,\, p,\, q)\,!\,:\,!\,F$

$\langle 1 \rangle 3.$ $F^{\#} \in [Proc \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 2$ DEF $InvIncomingInfoType$

$\langle 1 \rangle 4.$ $DeltaVecFunHasFiniteNonZeroRange(F^{\#})$
$\langle 2 \rangle$ DEFINE $FFP \triangleq \{fp \in Proc : F[fp] \neq DeltaVecZero\}$
$\langle 2 \rangle$ DEFINE $TFP \triangleq \{fp \in Proc : temp[fp] \neq DeltaVecZero\}$
$\langle 2 \rangle$ DEFINE $MFP \triangleq \{fp \in Proc : msg[fp][q] \neq \langle\rangle\}$

$\langle 2 \rangle$ SUFFICES $IsFiniteSet(FFP^{\#})$ BY $\langle 1 \rangle 3$ DEF $DeltaVecFunHasFiniteNonZeroRange$

$\langle 2 \rangle 1.$ $IsFiniteSet(TFP^{\#})$ BY $\langle 1 \rangle 1$ DEF $InvType,\, IsFiniteTempProcs$
$\langle 2 \rangle 2.$ $IsFiniteSet(MFP^{\#})$ BY $\langle 1 \rangle 1$ DEF $InvType,\, IsFiniteMsgSenders$
$\langle 2 \rangle 3.$ $IsFiniteSet(TFP^{\#} \cup MFP^{\#})$ BY $\langle 2 \rangle 2,\, \langle 2 \rangle 1,\, FiniteSetUnion$

$\langle 2 \rangle 4.$ $FFP^{\#} \subseteq (TFP^{\#} \cup MFP^{\#})$
$\langle 3 \rangle 1.$ SUFFICES ASSUME
NEW $fp \in Proc$,
$fp \notin TFP^{\#}$,
$fp \notin MFP^{\#}$

PROVE $fp \notin FFP^{\#}$
OBVIOUS

$\langle 3 \rangle$ SUFFICES $F^{\#}[fp] = DeltaVecZero$ OBVIOUS

$\langle 3 \rangle 2.$ $temp^{\#}[fp] = DeltaVecZero$ BY $\langle 3 \rangle 1$

$\langle 3 \rangle 3.$ ASSUME NEW $fk \in Nat$ PROVE $IncomingInfo(fk,\, fp,\, q)^{\#} = DeltaVecZero$
$\langle 4 \rangle$ DEFINE $sum \triangleq DeltaVecSeqSkipSum(fk,\, msg^{\#}[fp][q])$
$\langle 4 \rangle$ DEFINE $add \triangleq DeltaVecAdd(sum,\, temp^{\#}[fp])$
$\langle 4 \rangle 1.$ $sum = DeltaVecZero$
$\langle 5 \rangle 1.$ $msg^{\#}[fp][q] = \langle\rangle$ BY $\langle 3 \rangle 1$
$\langle 5 \rangle 2.$ $msg^{\#}[fp][q] \in Seq(DeltaVecType)$ BY $\langle 1 \rangle 1$ DEF $InvType$
$\langle 5 \rangle$ QED BY $\langle 5 \rangle 1$, $\langle 5 \rangle 2$, $DeltaVecSeqSkipSumEmpty$
$\langle 4 \rangle 2.$ $add = DeltaVecZero$ BY $\langle 3 \rangle 2$, $\langle 4 \rangle 1$, $DeltaVecAddZero$, $DeltaVecZeroType$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 2$ DEF $IncomingInfo$

$\langle 3 \rangle 4.$ CASE $fp = p$
$\langle 4 \rangle 1.$ $F^{\#}[fp] = IncomingInfo(k,\, fp,\, q)^{\#}$ BY $\langle 3 \rangle 4$
$\langle 4 \rangle 2.$ $k \in Nat$ OBVIOUS
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 3 \rangle 3$

$\langle 3 \rangle 5.$ CASE $fp \neq p$
$\langle 4 \rangle 1.$ $F^{\#}[fp] = IncomingInfo(0,\, fp,\, q)^{\#}$ BY $\langle 3 \rangle 5$
$\langle 4 \rangle 2.$ $0 \in Nat$ OBVIOUS
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 3 \rangle 3$

$\langle 3 \rangle$ QED BY $\langle 3 \rangle 4$, $\langle 3 \rangle 5$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 3$, $\langle 2 \rangle 4$, $FiniteSetSubset$

$\langle 1 \rangle 5.$ $GII^{\#} \in DeltaVecType$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $DeltaVecFunSumType$ DEF $GlobalIncomingInfo$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $\langle 1 \rangle 5$ DEF $GlobalIncomingInfo\_F$

The invariant $InvGlobalIncomingInfoSkip0$ follows from $InvType$.

THEOREM $DeduceInvGlobalIncomingInfoSkip0 \triangleq$
LET $goal \triangleq$
$InvType$
$\Rightarrow$
$InvGlobalIncomingInfoSkip0$
IN
$goal \land goal'$

PROOF
$\langle 1 \rangle$ DEFINE $goal \triangleq DeduceInvGlobalIncomingInfoSkip0!:!goal$
$\langle 1 \rangle$ DEFINE $DoPr(primeit, x) \triangleq$ IF $primeit$ THEN x' ELSE x

$\langle 1 \rangle$ SUFFICES ASSUME NEW $primeit \in$ BOOLEAN PROVE $DoPr(primeit, goal)$ OBVIOUS
$\langle 1 \rangle$ DEFINE $x^{\#} \triangleq DoPr(primeit, x)$

$\langle 1 \rangle$ SUFFICES ASSUME $InvType^{\#}$ PROVE $InvGlobalIncomingInfoSkip0^{\#}$ OBVIOUS

$\langle 1 \rangle$ SUFFICES ASSUME
NEW $p1 \in Proc$,
NEW $p2 \in Proc$,
NEW $q \in Proc$
PROVE $(GlobalIncomingInfo(0, p1, q) = GlobalIncomingInfo(0, p2, q))^{\#}$
BY DEF $InvGlobalIncomingInfoSkip0$

$\langle 1 \rangle$ DEFINE $GII(p) \triangleq GlobalIncomingInfo(0, p, q)$
$\langle 1 \rangle$ DEFINE $F(p) \triangleq GlobalIncomingInfo\_F(0, p, q)$

$\langle 1 \rangle 1.$ $(F(p1) = F(p2))^{\#}$ BY DEF $GlobalIncomingInfo\_F$

$\langle 1 \rangle 2.$ ASSUME NEW $p \in Proc$
PROVE $(GII(p) = DeltaVecFunSum(F(p)))^{\#}$
BY DEF $GlobalIncomingInfo$, $GlobalIncomingInfo\_F$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$

The invariant $InvGlobalIncomingInfoUpright$ follows from subsidiary invariants.

THEOREM $DeduceInvGlobalIncomingInfoUpright \triangleq$
LET $goal \triangleq$
$\land\ InvType$
$\land\ InvIncomingInfoUpright$
$\Rightarrow$
$InvGlobalIncomingInfoUpright$
IN
$goal \land goal'$
PROOF
$\langle 1 \rangle$ DEFINE $goal \triangleq DeduceInvGlobalIncomingInfoUpright!:!goal$
$\langle 1 \rangle$ DEFINE $DoPr(primeit, x) \triangleq$ IF $primeit$ THEN x' ELSE x

$\langle 1 \rangle$ SUFFICES ASSUME NEW $primeit \in$ BOOLEAN PROVE $DoPr(primeit, goal)$ OBVIOUS
$\langle 1 \rangle$ DEFINE $x^{\#} \triangleq DoPr(primeit, x)$

$\langle 1 \rangle$ SUFFICES ASSUME
$InvType^{\#}$,
$InvIncomingInfoUpright^{\#}$
PROVE $InvGlobalIncomingInfoUpright^{\#}$
OBVIOUS

$\langle 1 \rangle$ SUFFICES ASSUME
NEW $k \in Nat$,
NEW $p \in Proc$,
NEW $q \in Proc$
PROVE $IsDeltaVecUpright(lleq, GlobalIncomingInfo(k, p, q))^{\#}$
BY DEF $InvGlobalIncomingInfoUpright$

$\langle 1 \rangle$ $InvIncomingInfoType^{\#}$ BY $DeduceInvIncomingInfoType$
$\langle 1 \rangle$ $InvGlobalIncomingInfoType^{\#}$ BY $DeduceInvGlobalIncomingInfoType$

Pick a value corresponding to either the primed or the unprimed case. This makes things simpler for the prover in subsequent obligations by removing the $DoPr$ clutter.

$\langle 1 \rangle 1.$ PICK $lleqx : lleqx = lleq^{\#}$ OBVIOUS
$\langle 1 \rangle 2.$ PICK $Fx : Fx = GlobalIncomingInfo\_F(k, p, q)^{\#}$ OBVIOUS
$\langle 1 \rangle 3.$ PICK $GIIx : GIIx = GlobalIncomingInfo(k, p, q)^{\#}$ OBVIOUS

$\langle 1 \rangle 4.$ $lleqx \in PointRelationType$ BY $\langle 1 \rangle 1$ DEF $InvType$
$\langle 1 \rangle 5.$ $IsPartialOrder(lleqx)$ BY $\langle 1 \rangle 1$ DEF $InvType$

$\langle 1 \rangle 6.$ $Fx \in [Proc \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 2$ DEF $InvGlobalIncomingInfoType$
$\langle 1 \rangle 7.$ $DeltaVecFunHasFiniteNonZeroRange(Fx)$ BY $\langle 1 \rangle 2$ DEF $InvGlobalIncomingInfoType$
$\langle 1 \rangle 8.$ $GIIx = DeltaVecFunSum(Fx)$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$ DEF $GlobalIncomingInfo\_F$, $GlobalIncomingInfo$

$\langle 1 \rangle 9.$ ASSUME NEW $p1 \in Proc$ PROVE $IsDeltaVecUpright(lleqx, Fx[p1])$
$\langle 2 \rangle 1.$ ASSUME NEW $k1 \in Nat$ PROVE $IsDeltaVecUpright(lleqx, IncomingInfo(k1, p1, q)^{\#})$
BY $\langle 1 \rangle 1$ DEF $InvIncomingInfoUpright$

$\langle 2 \rangle 2.$ CASE $p1 = p$
$\langle 3 \rangle 1.$ $Fx[p1] = IncomingInfo(k, p1, q)^{\#}$ BY $\langle 2 \rangle 2$, $\langle 1 \rangle 2$ DEF $GlobalIncomingInfo\_F$
$\langle 3 \rangle 2.$ $k \in Nat$ OBVIOUS
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 2 \rangle 1$
$\langle 2 \rangle 3.$ CASE $p1 \neq p$
$\langle 3 \rangle 1.$ $Fx[p1] = IncomingInfo(0, p1, q)^{\#}$ BY $\langle 2 \rangle 3$, $\langle 1 \rangle 2$ DEF $GlobalIncomingInfo\_F$
$\langle 3 \rangle 2.$ $0 \in Nat$ OBVIOUS
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 2 \rangle 1$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$

$\langle 1 \rangle 10.$ $IsDeltaVecUpright(lleqx, DeltaVecFunSum(Fx))$
$\langle 2 \rangle$ USE $\langle 1 \rangle 4$, $\langle 1 \rangle 5$, $\langle 1 \rangle 6$, $\langle 1 \rangle 7$, $\langle 1 \rangle 9$
$\langle 2 \rangle$ QED BY $DeltaVecUpright\_FunSum$

$\langle 1 \rangle 11.$ $IsDeltaVecUpright(lleqx, GIIx)$ BY $\langle 1 \rangle 8$, $\langle 1 \rangle 10$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 3$, $\langle 1 \rangle 11$

The invariant $InvGlobVacantUptoImpliesNrec$ follows from subsidiary invariants.

THEOREM $DeduceInvGlobVacantUptoImpliesNrec \triangleq$
  LET $goal \triangleq$
    $\land\ InvType$
    $\land\ InvGlobalIncomingInfoUpright$
    $\land\ InvGlobalRecordCount$
    $\Rightarrow$
    $InvGlobVacantUptoImpliesNrec$
  IN
  $goal \land goal'$
PROOF
  $\langle 1 \rangle$ DEFINE $goal \triangleq DeduceInvGlobVacantUptoImpliesNrec!:!goal$
  $\langle 1 \rangle$ DEFINE $DoPr(primeit, x) \triangleq$ IF $primeit$ THEN x' ELSE x

  $\langle 1 \rangle$ SUFFICES ASSUME NEW $primeit \in$ BOOLEAN PROVE $DoPr(primeit, goal)$ OBVIOUS
  $\langle 1 \rangle$ DEFINE $x^{\#} \triangleq DoPr(primeit, x)$

  $\langle 1 \rangle$ SUFFICES ASSUME
      $InvType^{\#}$,
      $InvGlobalIncomingInfoUpright^{\#}$,
      $InvGlobalRecordCount^{\#}$
    PROVE $InvGlobVacantUptoImpliesNrec^{\#}$
    OBVIOUS

  $\langle 1 \rangle$ $InvGlobalIncomingInfoType^{\#}$ BY $DeduceInvGlobalIncomingInfoType$

  $\langle 1 \rangle$ SUFFICES ASSUME
      NEW $q \in Proc$,
      NEW $t \in Point$,
      $GlobVacantUpto(q, t)^{\#}$
    PROVE $NrecVacantUpto(t)^{\#}$
    BY DEF $InvGlobVacantUptoImpliesNrec$

Pick a value corresponding to either the primed or the unprimed case. This makes things simpler for the prover in subsequent obligations by removing the $DoPr$ clutter.

  $\langle 1 \rangle 1.$ PICK $nrecx : nrecx = nrec^{\#}$ OBVIOUS
  $\langle 1 \rangle 2.$ PICK $GIIx : GIIx = GlobalIncomingInfo(0, q, q)^{\#}$ OBVIOUS
  $\langle 1 \rangle 3.$ PICK $globxq : globxq = glob[q]^{\#}$ OBVIOUS
  $\langle 1 \rangle 4.$ PICK $lleqx : lleqx = lleq^{\#}$ OBVIOUS

$\langle 1\rangle$ DEFINE $a \preceq b \triangleq lleqx[a][b]$
$\langle 1\rangle$ DEFINE $a \prec b \triangleq a \preceq b \land a \neq b$

$\langle 1\rangle 5.\ lleqx \in PointRelationType$ BY $\langle 1\rangle 4$ DEF $InvType$
$\langle 1\rangle 6.\ IsPartialOrder(lleqx)$ BY $\langle 1\rangle 4$ DEF $InvType$
$\langle 1\rangle 7.\ nrecx = DeltaVecAdd(GIIx,\ globxq)$ BY $\langle 1\rangle 1,\ \langle 1\rangle 2,\ \langle 1\rangle 3$ DEF $InvGlobalRecordCount$
$\langle 1\rangle 8.\ nrecx \in CountVecType$ BY $\langle 1\rangle 1$ DEF $InvType$
$\langle 1\rangle 9.\ globxq \in DeltaVecType$ BY $\langle 1\rangle 3$ DEF $InvType$
$\langle 1\rangle 10.\ IsDeltaVecUpright(lleqx,\ GIIx)$ BY $\langle 1\rangle 2,\ \langle 1\rangle 4$ DEF $InvGlobalIncomingInfoUpright$
$\langle 1\rangle 11.\ GIIx \in DeltaVecType$ BY $\langle 1\rangle 2$ DEF $InvGlobalIncomingInfoType$

$\langle 1\rangle 12.$ SUFFICES ASSUME NEW $s \in Point,\ s \preceq t$ PROVE $nrecx[s] = 0$
BY $\langle 1\rangle 1,\ \langle 1\rangle 4$ DEF $NrecVacantUpto,\ IsDeltaVecVacantUpto$

$\langle 1\rangle 13.\ nrecx[s] \in Nat$ BY $\langle 1\rangle 8$ DEF $CountVecType$
$\langle 1\rangle 14.\ globxq[s] \in Int$ BY $\langle 1\rangle 9$ DEF $DeltaVecType$
$\langle 1\rangle 15.\ GIIx[s] \in Int$ BY $\langle 1\rangle 11$ DEF $DeltaVecType$

Since $nrec$ is a count vector, all its points must be non-negative. Hence if there is a point $s\ lleq\ t$ such that $nrec[s] \neq 0$, it must be the case that $nrec[s] > 0$. Assume we have such an s and prove a contradiction.

$\langle 1\rangle 16.$ SUFFICES ASSUME $nrecx[s] > 0$ PROVE FALSE BY $\langle 1\rangle 13,\ SMTT(10)$

Since point $s \preceq t$, we have $globq[s] = 0$. Since $nrec = GII + globq$, we have $GII[s] > 0$.

$\langle 1\rangle 17.\ nrecx[s] = GIIx[s] + globxq[s]$ BY $\langle 1\rangle 7$ DEF $DeltaVecAdd$
$\langle 1\rangle 18.\ globxq[s] = 0$ BY $\langle 1\rangle 3,\ \langle 1\rangle 4,\ \langle 1\rangle 12$ DEF $GlobVacantUpto,\ IsDeltaVecVacantUpto$
$\langle 1\rangle 19.\ GIIx[s] > 0$ BY $\langle 1\rangle 13,\ \langle 1\rangle 15,\ \langle 1\rangle 16,\ \langle 1\rangle 17,\ \langle 1\rangle 18,\ SMTT(10)$

Since GII is an upright delta vector and $GII[s] > 0$, there must be a point $u \preceq s$ such that $GII[u] < 0$.

$\langle 1\rangle 20.$ PICK $u \in Point : u \preceq s \land GIIx[u] < 0$
BY $\langle 1\rangle 5,\ \langle 1\rangle 6,\ \langle 1\rangle 10,\ \langle 1\rangle 11,\ \langle 1\rangle 19,\ DeltaVecUpright\_ExistsSupport$

But then point $u \preceq t$, which means that $globq[u] = 0$. Since $nrec = GII + globq$, we can conclude that $GII[u] < 0$ cannot be true.

$\langle 1\rangle 21.\ u \preceq t$ BY $\langle 1\rangle 5,\ \langle 1\rangle 6,\ \langle 1\rangle 12,\ \langle 1\rangle 20,\ PartialOrderTransitive$

$\langle 1\rangle 22.\ globxq[u] = 0$ BY $\langle 1\rangle 3,\ \langle 1\rangle 4,\ \langle 1\rangle 21$ DEF $GlobVacantUpto,\ IsDeltaVecVacantUpto$
$\langle 1\rangle 23.\ nrecx[u] \in Nat$ BY $\langle 1\rangle 8$ DEF $CountVecType$
$\langle 1\rangle 24.\ GIIx[u] \in Int$ BY $\langle 1\rangle 11$ DEF $DeltaVecType$

$\langle 1\rangle 25.\ nrecx[u] = GIIx[u] + globxq[u]$ BY $\langle 1\rangle 7$ DEF $DeltaVecAdd$
$\langle 1\rangle 26.\ \neg(GIIx[u] < 0)$ BY $\langle 1\rangle 22,\ \langle 1\rangle 23,\ \langle 1\rangle 24,\ \langle 1\rangle 25,\ SMTT(10)$

$\langle 1\rangle$ QED BY $\langle 1\rangle 20,\ \langle 1\rangle 26$

The invariant $InvGlobalInfoAtBetaUpright$ follows from subsidiary invariants.

THEOREM $DeduceInvGlobalInfoAtBetaUpright \triangleq$
  LET $goal \triangleq$
    $\land\ InvType$
    $\land\ InvInfoAtBetaUpright$
    $\land\ InvIncomingInfoUpright$
    $\Rightarrow$
   $InvGlobalInfoAtBetaUpright$
  IN
  $goal \land goal'$
PROOF
  $\langle 1 \rangle$ DEFINE $goal \triangleq DeduceInvGlobalInfoAtBetaUpright\,!:!goal$
  $\langle 1 \rangle$ DEFINE $DoPr(primeit, x) \triangleq$ IF $primeit$ THEN x' ELSE x

  $\langle 1 \rangle$ SUFFICES ASSUME NEW $primeit \in$ BOOLEAN PROVE $DoPr(primeit, goal)$ OBVIOUS
  $\langle 1 \rangle$ DEFINE $x^{\#} \triangleq DoPr(primeit, x)$

  $\langle 1 \rangle$ SUFFICES ASSUME
      $InvType^{\#}$,
      $InvInfoAtBetaUpright^{\#}$,
      $InvIncomingInfoUpright^{\#}$
    PROVE $InvGlobalInfoAtBetaUpright^{\#}$
    OBVIOUS

  $\langle 1 \rangle$ $InvInfoAtType^{\#}$ BY $DeduceInvInfoAtType$
  $\langle 1 \rangle$ $InvIncomingInfoType^{\#}$ BY $DeduceInvIncomingInfoType$
  $\langle 1 \rangle$ $InvGlobalIncomingInfoType^{\#}$ BY $DeduceInvGlobalIncomingInfoType$

The prover chews right through all the $DoPr$ clutter with no problem. It looks like the prover has gotten better since I wrote some of the other proofs in this module.

  $\langle 1 \rangle$ SUFFICES ASSUME
      NEW $k \in Nat$,
      NEW $p \in Proc$,
      NEW $q \in Proc$
    PROVE
    LET
      $IA \triangleq InfoAt(k, p, q)$
      $GII \triangleq GlobalIncomingInfo(k, p, q)$
    IN
    $IsDeltaVecBetaUpright(lleq, IA, GII)^{\#}$
    BY DEF $InvGlobalInfoAtBetaUpright$

  $\langle 1 \rangle$ DEFINE $IA \triangleq InfoAt(k, p, q)$
  $\langle 1 \rangle$ DEFINE $II \triangleq IncomingInfo(k, p, q)$

$\langle 1 \rangle$ DEFINE $GII \triangleq GlobalIncomingInfo(k, p, q)$
$\langle 1 \rangle$ DEFINE $F \triangleq GlobalIncomingInfo\_F(k, p, q)$

$\langle 1 \rangle$ SUFFICES $IsDeltaVecBetaUpright(lleq, IA, GII)^{\#}$ OBVIOUS

$\langle 1 \rangle 6.$ $(lleq \in PointRelationType)^{\#}$ BY DEF $InvType$
$\langle 1 \rangle 7.$ $(IsPartialOrder(lleq))^{\#}$ BY DEF $InvType$
$\langle 1 \rangle 8.$ $(IA \in DeltaVecType)^{\#}$ BY DEF $InvInfoAtType$
$\langle 1 \rangle 9.$ $(IsDeltaVecBetaUpright(lleq, IA, II))^{\#}$ BY DEF $InvInfoAtBetaUpright$
$\langle 1 \rangle 10.$ $(II = F[p])^{\#}$ BY DEF $GlobalIncomingInfo\_F$
$\langle 1 \rangle 11.$ $(GII \in DeltaVecType)^{\#}$ BY DEF $InvGlobalIncomingInfoType$
$\langle 1 \rangle 12.$ $(F \in [Proc \rightarrow DeltaVecType])^{\#}$ BY DEF $InvGlobalIncomingInfoType$
$\langle 1 \rangle 13.$ $(DeltaVecFunHasFiniteNonZeroRange(F))^{\#}$ BY DEF $InvGlobalIncomingInfoType$
$\langle 1 \rangle 14.$ $(GII = DeltaVecFunSum(F))^{\#}$ BY DEF $GlobalIncomingInfo\_F$, $GlobalIncomingInfo$

$\langle 1 \rangle 15.$ ASSUME NEW $p1 \in Proc$ PROVE $IsDeltaVecUpright(lleq, F[p1])^{\#}$
  $\langle 2 \rangle 1.$ ASSUME NEW $k1 \in Nat$ PROVE $IsDeltaVecUpright(lleq, IncomingInfo(k1, p1, q))^{\#}$
    BY DEF $InvIncomingInfoUpright$

  $\langle 2 \rangle 2.$ CASE $p1 = p$
    $\langle 3 \rangle 1.$ $(F[p1] = IncomingInfo(k, p1, q))^{\#}$ BY $\langle 2 \rangle 2$ DEF $GlobalIncomingInfo\_F$
    $\langle 3 \rangle 2.$ $k \in Nat$ OBVIOUS
    $\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 2 \rangle 1$
  $\langle 2 \rangle 3.$ CASE $p1 \neq p$
    $\langle 3 \rangle 1.$ $(F[p1] = IncomingInfo(0, p1, q))^{\#}$ BY $\langle 2 \rangle 3$ DEF $GlobalIncomingInfo\_F$
    $\langle 3 \rangle 2.$ $0 \in Nat$ OBVIOUS
    $\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 2 \rangle 1$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$

$\langle 1 \rangle 16.$ $DeltaVecBetaUpright\_FunSum\_Hypothesis(lleq, F, IA, p)^{\#}$
    BY $\langle 1 \rangle 6$, $\langle 1 \rangle 7$, $\langle 1 \rangle 8$, $\langle 1 \rangle 9$, $\langle 1 \rangle 10$, $\langle 1 \rangle 12$, $\langle 1 \rangle 13$, $\langle 1 \rangle 15$
    DEF $DeltaVecBetaUpright\_FunSum\_Hypothesis$

$\langle 1 \rangle 17.$ $IsDeltaVecBetaUpright(lleq, IA, DeltaVecFunSum(F))^{\#}$
    BY $\langle 1 \rangle 16$, $DeltaVecBetaUpright\_FunSum$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 14$, $\langle 1 \rangle 17$

C.16 How the actions affect the state variables

MODULE $NaiadClockProofAffectState$

EXTENDS $NaiadClockProofDeduceInv$

How the actions affect the state variables

Here we establish a lot of facts about how each action affects the various state variables. Later, we repeatedly appeal to these facts when proving facts about how each action affects a state operator or when proving that an action preserves an invariant.

How $NextCommon$ updates the state variables.

$NextCommon\_State\_Conclusion \triangleq$
$\wedge$ UNCHANGED $lleq$
$\wedge\ nrecvut' = [xt \in Point \mapsto NrecVacantUpto(xt)]$
$\wedge\ globvut' = [xp \in Proc \mapsto [xt \in Point \mapsto GlobVacantUpto(xp, xt)]]$

$\wedge\ nrecvut' \in [Point \rightarrow \text{BOOLEAN}]$
$\wedge\ globvut' \in [Proc \rightarrow [Point \rightarrow \text{BOOLEAN}]]$
$\wedge\ IsPartialOrder(lleq')$

THEOREM $NextCommon\_State \triangleq$
ASSUME
$InvType$,
$NextCommon$
PROVE
$NextCommon\_State\_Conclusion$
PROOF

Type and value of $lleq'$

$\langle 1 \rangle 1.$ UNCHANGED $lleq$ BY DEF $NextCommon$
$\langle 1 \rangle 2.$ $IsPartialOrder(lleq')$ BY $\langle 1 \rangle 1$ DEF $InvType$

Type of $nrecvut'$

$\langle 1 \rangle 3.$ $nrecvut' = [xt \in Point \mapsto NrecVacantUpto(xt)]$
BY DEF $NextCommon$

$\langle 1 \rangle 4.\ nrecvut' \in [Point \rightarrow \text{BOOLEAN}]$
  $\langle 2 \rangle$ USE DEF $NrecVacantUpto$
  $\langle 2 \rangle$ USE DEF $IsDeltaVecVacantUpto$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 3$

Type of $globvut'$

$\langle 1 \rangle 5.\ globvut' = [xp \in Proc \mapsto [xt \in Point \mapsto GlobVacantUpto(xp, xt)]]$
    BY DEF $NextCommon$

$\langle 1 \rangle 6.\ globvut' \in [Proc \rightarrow [Point \rightarrow \text{BOOLEAN}]]$
  $\langle 2 \rangle$ USE DEF $GlobVacantUpto$
  $\langle 2 \rangle$ USE DEF $IsDeltaVecVacantUpto$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 5$

$\langle 1 \rangle$ USE DEF $NextCommon\_State\_Conclusion$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $\langle 1 \rangle 5$, $\langle 1 \rangle 6$

How the $NextPerformOperation(p, c, r)$ action updates the state variables.

$NextPerformOperation\_State\_Conclusion(p, c, r) \triangleq$
  LET
    $delta \triangleq NextPerformOperation\_Delta(p, c, r)$
  IN
  $\land\ c \in [Point \rightarrow Nat]$
  $\land\ r \in [Point \rightarrow Nat]$
  $\land\ delta \in DeltaVecType$
  $\land\ \forall\, xt \in Point : c[xt] \leq nrec[xt]$
  $\land\ IsDeltaVecUpright(lleq, delta)$

  $\land\ nrec$::
    $\land\ nrec' = DeltaVecAdd(nrec, delta)$
    $\land\ nrec' = DeltaVecAdd(delta, nrec)$

  $\land\ temp$::
    $\forall\, fp \in Proc :$
      IF $fp = p$
      THEN
        $\land\ temp'[fp] = DeltaVecAdd(temp[fp], delta)$
        $\land\ temp'[fp] = DeltaVecAdd(delta, temp[fp])$
      ELSE UNCHANGED $temp[fp]$

$\land$ UNCHANGED $glob$
$\land$ UNCHANGED msg
$\land$ $NextCommon\_State\_Conclusion!$:

$\land$ $InvType'$

THEOREM $NextPerformOperation\_State \triangleq$
ASSUME
NEW $p \in Proc$,
NEW $c \in PointToNat$,
NEW $r \in PointToNat$,
$InvType$,
$NextPerformOperation\_WithPCR(p, c, r)$
PROVE
$NextPerformOperation\_State\_Conclusion(p, c, r)$
PROOF
$\langle 1 \rangle$ USE DEF $NextPerformOperation\_Delta$

$\langle 1 \rangle$ DEFINE $delta \triangleq NextPerformOperation\_Delta(p, c, r)$
$\langle 1 \rangle$ HIDE DEF $delta$

Type and value of c and r

$\langle 1 \rangle 1.$ $c \in [Point \rightarrow Nat]$ BY $AssumePointToNat$
$\langle 1 \rangle 2.$ $r \in [Point \rightarrow Nat]$ BY $AssumePointToNat$

$\langle 1 \rangle 3.$ $\forall xt \in Point : c[xt] \in Nat$ BY $\langle 1 \rangle 1$
$\langle 1 \rangle 4.$ $\forall xt \in Point : r[xt] \in Nat$ BY $\langle 1 \rangle 2$

Type and value of delta

$\langle 1 \rangle 5.$ $\forall xt \in Point : delta[xt] = r[xt] - c[xt]$ BY DEF $NextPerformOperation\_WithPCR$, $delta$
$\langle 1 \rangle 6.$ $\forall xt \in Point : delta[xt] \in Int$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $\langle 1 \rangle 5$, $SMTT(10)$

$\langle 1 \rangle 7.$ $\exists S : delta \in [Point \rightarrow S]$ BY Isa DEF $NextPerformOperation\_WithPCR$, $delta$
$\langle 1 \rangle 8.$ $delta \in DeltaVecType$ BY $\langle 1 \rangle 6$, $\langle 1 \rangle 7$ DEF $DeltaVecType$

$\langle 1 \rangle 9.$ $IsDeltaVecUpright(lleq, delta)$ BY DEF $NextPerformOperation\_WithPCR$, $delta$

Type and value of $nrec'$

$\langle 1 \rangle 10.$ $nrec \in CountVecType$ BY DEF $InvType$
$\langle 1 \rangle 11.$ $nrec \in DeltaVecType$ BY $\langle 1 \rangle 10$, $SMTT(10)$ DEF $CountVecType$, $DeltaVecType$

$\langle 1 \rangle 12.$ $\forall xt \in Point : nrec[xt] \in Nat$ BY $\langle 1 \rangle 10$ DEF $CountVecType$
$\langle 1 \rangle 13.$ $\forall xt \in Point : c[xt] \leq nrec[xt]$ BY DEF $NextPerformOperation\_WithPCR$
$\langle 1 \rangle 14.$ $\forall xt \in Point : nrec[xt] + (r[xt] - c[xt]) \in Nat$ BY $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $\langle 1 \rangle 12$, $\langle 1 \rangle 13$, $SMTT(10)$
$\langle 1 \rangle 15.$ $\forall xt \in Point : nrec[xt] + delta[xt] \in Nat$ BY $\langle 1 \rangle 5$, $\langle 1 \rangle 14$

$\langle 1 \rangle 16.$ $nrec' = DeltaVecAdd(nrec, delta)$ BY DEF $NextPerformOperation\_WithPCR$, $delta$

$\langle 1\rangle 17.\ nrec' = DeltaVecAdd(delta,\ nrec)$ BY $\langle 1\rangle 8,\ \langle 1\rangle 11,\ \langle 1\rangle 16,\ DeltaVecAddCommutative$

$\langle 1\rangle 18.\ \forall\, xt \in Point : nrec'[xt] = nrec[xt] + delta[xt]$ BY $\langle 1\rangle 16$ DEF $DeltaVecAdd$
$\langle 1\rangle 19.\ \forall\, xt \in Point : nrec'[xt] \in Nat$ BY $\langle 1\rangle 18,\ \langle 1\rangle 15$

$\langle 1\rangle 20.\ nrec' \in DeltaVecType$ BY $\langle 1\rangle 8,\ \langle 1\rangle 11,\ \langle 1\rangle 16,\ DeltaVecAddType$
$\langle 1\rangle 21.\ nrec' \in CountVecType$ BY $\langle 1\rangle 19,\ \langle 1\rangle 20$ DEF $DeltaVecType,\ CountVecType$

$\langle 1\rangle 22.\ NextPerformOperation\_State\_Conclusion(p,\ c,\ r)!nrec$
BY $\langle 1\rangle 16,\ \langle 1\rangle 17$ DEF $delta$

Type and value of $temp'$

$\langle 1\rangle 23.\ temp \in [Proc \to DeltaVecType]$ BY DEF $InvType$

$\langle 1\rangle 24.\ DeltaVecAdd(temp[p],\ delta) \in DeltaVecType$ BY $\langle 1\rangle 8,\ \langle 1\rangle 23,\ DeltaVecAddType$

$\langle 1\rangle 25.\ temp' = [temp \text{ EXCEPT } ![p] = DeltaVecAdd(temp[p],\ delta)]$
BY $\langle 1\rangle 23$ DEF $NextPerformOperation\_WithPCR,\ delta$

$\langle 1\rangle 26.\ temp' \in [Proc \to DeltaVecType]$ BY $\langle 1\rangle 23,\ \langle 1\rangle 24,\ \langle 1\rangle 25$

$\langle 1\rangle 27.\ temp'[p] = DeltaVecAdd(temp[p],\ delta)$ BY $\langle 1\rangle 25,\ \langle 1\rangle 26$

$\langle 1\rangle 28.\ temp'[p] = DeltaVecAdd(delta,\ temp[p])$ BY $\langle 1\rangle 8,\ \langle 1\rangle 23,\ \langle 1\rangle 27,\ DeltaVecAddCommutative$

$\langle 1\rangle 29.$ ASSUME NEW $fp \in Proc,\ fp \neq p$
PROVE UNCHANGED $temp[fp]$
BY $\langle 1\rangle 23,\ \langle 1\rangle 25,\ \langle 1\rangle 29$

$\langle 1\rangle 30.\ NextPerformOperation\_State\_Conclusion(p,\ c,\ r)!temp$
BY $\langle 1\rangle 27,\ \langle 1\rangle 28,\ \langle 1\rangle 29$ DEF $delta$

Type and value of $glob'$

$\langle 1\rangle 31.$ UNCHANGED $glob$ BY DEF $NextPerformOperation\_WithPCR$
$\langle 1\rangle 32.\ glob' \in [Proc \to DeltaVecType]$ BY $\langle 1\rangle 31$ DEF $InvType$

Type and value of msg'

$\langle 1\rangle 33.$ UNCHANGED msg BY DEF $NextPerformOperation\_WithPCR$
$\langle 1\rangle 34.\ msg' \in [Proc \to [Proc \to Seq(DeltaVecType)]]$ BY $\langle 1\rangle 33$ DEF $InvType$

$\langle 1\rangle 35.\ NextCommon\_State\_Conclusion! :$
$\langle 2\rangle$ USE DEF $NextPerformOperation\_WithPCR$
$\langle 2\rangle$ USE DEF $NextCommon\_State\_Conclusion$
$\langle 2\rangle$ QED BY $NextCommon\_State$

$IsFiniteTempProcs'$

$\langle 1\rangle 36.\ IsFiniteTempProcs'$
$\langle 2\rangle$ DEFINE $FP \triangleq \{fp \in Proc : temp[fp] \neq DeltaVecZero\}$
$\langle 2\rangle 1.\ IsFiniteSet(FP)$ BY DEF $InvType,\ IsFiniteTempProcs$

$\langle 2\rangle 2.$ $IsFiniteSet(\{p\})$ BY $FiniteSetSingleton$
$\langle 2\rangle 3.$ $IsFiniteSet(FP \cup \{p\})$ BY $\langle 2\rangle 1$, $\langle 2\rangle 2$, $FiniteSetUnion$
$\langle 2\rangle 4.$ $FP' \subseteq (FP \cup \{p\})$ BY $\langle 1\rangle 29$
$\langle 2\rangle 5.$ $IsFiniteSet(FP')$ BY $\langle 2\rangle 3$, $\langle 2\rangle 4$, $FiniteSetSubset$
$\langle 2\rangle$ QED BY $\langle 2\rangle 5$ DEF $IsFiniteTempProcs$

$IsFiniteMsgSenders'$

$\langle 1\rangle 37.$ $IsFiniteMsgSenders'$
$\langle 2\rangle 1.$ $IsFiniteMsgSenders$ BY DEF $InvType$
$\langle 2\rangle$ QED BY $\langle 2\rangle 1$, $\langle 1\rangle 33$ DEF $IsFiniteMsgSenders$

$\langle 1\rangle 38.$ $InvType'$ BY $\langle 1\rangle 21$, $\langle 1\rangle 26$, $\langle 1\rangle 32$, $\langle 1\rangle 34$, $\langle 1\rangle 35$, $\langle 1\rangle 36$, $\langle 1\rangle 37$ DEF $InvType$

$\langle 1\rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$
$\langle 1\rangle$ USE DEF $delta$
$\langle 1\rangle$ QED BY $\langle 1\rangle 1$, $\langle 1\rangle 2$, $\langle 1\rangle 8$, $\langle 1\rangle 9$, $\langle 1\rangle 13$, $\langle 1\rangle 22$, $\langle 1\rangle 30$, $\langle 1\rangle 31$, $\langle 1\rangle 33$, $\langle 1\rangle 35$, $\langle 1\rangle 38$

How the $NextSendUpdate(p,\ t)$ action updates the state variables.

$NextSendUpdate\_State\_Conclusion(p,\ tt) \triangleq$
LET
$gamma \triangleq NextSendUpdate\_Gamma(p,\ tt)$
IN
$\land\ gamma \in DeltaVecType$
$\land\ IsDeltaVecPositiveImplies(gamma,\ temp[p])$
$\land\ IsDeltaVecUpright(lleq,\ temp[p]) \Rightarrow IsDeltaVecUpright(lleq,\ temp[p])'$

$\land\ temp::$
$\forall fp \in Proc :$
IF $fp = p$
THEN
$\land\ temp[fp] = DeltaVecAdd(temp'[fp],\ gamma)$
$\land\ temp[fp] = DeltaVecAdd(gamma,\ temp'[fp])$
ELSE UNCHANGED $temp[fp]$

$\land\ msg::$
$\forall fp \in Proc :$
$\forall fq \in Proc :$
IF $fp = p$
THEN $msg'[fp][fq] = Append(msg[fp][fq],\ gamma)$
ELSE UNCHANGED $msg[fp][fq]$

$\land$ UNCHANGED $glob$
$\land$ UNCHANGED $nrec$
$\land$ $NextCommon\_State\_Conclusion\,!\,:$

$\land$ $InvType'$

THEOREM $NextSendUpdate\_State \triangleq$
ASSUME
NEW $p \in Proc$,
NEW $tt \in$ SUBSET $Point$,
$InvType$,
$NextSendUpdate\_WithPTT(p, tt)$
PROVE
$NextSendUpdate\_State\_Conclusion(p, tt)$
PROOF
$\langle 1 \rangle$ USE DEF $NextSendUpdate\_Gamma$

$\langle 1 \rangle$ DEFINE $gamma \triangleq NextSendUpdate\_Gamma(p, tt)$
$\langle 1 \rangle$ DEFINE $newtempp \triangleq NextSendUpdate\,!(p)!(tt)!newtempp$
$\langle 1 \rangle$ HIDE DEF $gamma$, $newtempp$

$\langle 1 \rangle 1.$ $temp \in [Proc \to DeltaVecType]$ BY DEF $InvType$
$\langle 1 \rangle 2.$ $msg \in [Proc \to [Proc \to Seq(DeltaVecType)]]$ BY DEF $InvType$

$\langle 1 \rangle 3.$ $gamma \in DeltaVecType$
$\langle 2 \rangle$ USE DEF $DeltaVecType$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 1$, $SMTT(10)$ DEF $gamma$

$\langle 1 \rangle 4.$ $newtempp \in DeltaVecType$
$\langle 2 \rangle 1.$ $0 \in Int$ BY $SMTT(10)$
$\langle 2 \rangle$ USE DEF $DeltaVecType$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 1 \rangle 1$ DEF $newtempp$

$\langle 1 \rangle 5.$ $temp' = [temp$ EXCEPT $![p] = newtempp]$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 4$ DEF $NextSendUpdate\_WithPTT$, $newtempp$

Type and value of $temp'$ in relation to gamma

$\langle 1 \rangle 6.$ $temp' \in [Proc \to DeltaVecType]$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 4$, $\langle 1 \rangle 5$

$\langle 1 \rangle 7.$ ASSUME NEW $t \in Point$
PROVE $gamma[t] =$ IF $t \in tt$ THEN $temp[p][t]$ ELSE 0
BY DEF $gamma$, $NextSendUpdate\_WithPTT$

$\langle 1 \rangle 8.$ ASSUME NEW $t \in Point$
PROVE $temp'[p][t] =$ IF $t \in tt$ THEN 0 ELSE $temp[p][t]$
BY $\langle 1 \rangle 1$ DEF $NextSendUpdate\_WithPTT$

$\langle 1 \rangle 9.$ $temp[p] = DeltaVecAdd(gamma,\ temp'[p])$
$\langle 2 \rangle$ SUFFICES ASSUME NEW $t \in Point$
PROVE $temp[p][t] = gamma[t] + temp'[p][t]$
BY $\langle 1 \rangle 1$ DEF $DeltaVecAdd,\ DeltaVecType$
$\langle 2 \rangle 1.$ $gamma[t] \in Int$ BY $\langle 1 \rangle 3$ DEF $DeltaVecType$
$\langle 2 \rangle 2.$ $temp[p][t] \in Int$ BY $\langle 1 \rangle 1$ DEF $DeltaVecType$
$\langle 2 \rangle 3.$ $temp'[p][t] \in Int$ BY $\langle 1 \rangle 6$ DEF $DeltaVecType$
$\langle 2 \rangle 4.$ CASE $t \in tt$
$\langle 3 \rangle 1.$ $gamma[t] = temp[p][t]$ BY $\langle 2 \rangle 4,\ \langle 1 \rangle 7$
$\langle 3 \rangle 2.$ $temp'[p][t] = 0$ BY $\langle 2 \rangle 4,\ \langle 1 \rangle 8$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 3 \rangle 2,\ \langle 2 \rangle 1,\ \langle 2 \rangle 2,\ \langle 2 \rangle 3,\ SMTT(10)$
$\langle 2 \rangle 5.$ CASE $t \notin tt$
$\langle 3 \rangle 1.$ $gamma[t] = 0$ BY $\langle 2 \rangle 5,\ \langle 1 \rangle 7$
$\langle 3 \rangle 2.$ $temp'[p][t] = temp[p][t]$ BY $\langle 2 \rangle 5,\ \langle 1 \rangle 8$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 3 \rangle 2,\ \langle 2 \rangle 1,\ \langle 2 \rangle 2,\ \langle 2 \rangle 3,\ SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 4,\ \langle 2 \rangle 5$

$\langle 1 \rangle 10.$ $temp[p] = DeltaVecAdd(temp'[p],\ gamma)$
BY $\langle 1 \rangle 3,\ \langle 1 \rangle 6,\ \langle 1 \rangle 9,\ DeltaVecAddCommutative$

$\langle 1 \rangle 11.$ ASSUME NEW $fp \in Proc,\ fp \neq p$ PROVE UNCHANGED $temp[fp]$
$\langle 2 \rangle$ $temp' = [temp$ EXCEPT $![p] = temp'[p]]$ BY $\langle 1 \rangle 1,\ \langle 1 \rangle 6$ DEF $NextSendUpdate\_WithPTT$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 1,\ \langle 1 \rangle 11$

$\langle 1 \rangle 12.$ $IsDeltaVecPositiveImplies(gamma,\ temp[p])$
$\langle 2 \rangle 1.$ SUFFICES ASSUME NEW $t \in Point,\ gamma[t] > 0$
PROVE $temp[p][t] > 0$
BY $\langle 1 \rangle 1,\ \langle 1 \rangle 3$ DEF $IsDeltaVecPositiveImplies$
$\langle 2 \rangle 2.$ $gamma[t] \in Int$ BY $\langle 1 \rangle 3$ DEF $DeltaVecType$
$\langle 2 \rangle 3.$ $temp[p][t] \in Int$ BY $\langle 1 \rangle 1$ DEF $DeltaVecType$
$\langle 2 \rangle 4.$ $temp'[p][t] \in Int$ BY $\langle 1 \rangle 6$ DEF $DeltaVecType$
$\langle 2 \rangle 5.$ $temp[p][t] = gamma[t] + temp'[p][t]$ BY $\langle 1 \rangle 9$ DEF $DeltaVecAdd$
$\langle 2 \rangle 6.$ CASE $t \in tt$
$\langle 3 \rangle 1.$ $gamma[t] = temp[p][t]$ BY $\langle 2 \rangle 6,\ \langle 1 \rangle 7$
$\langle 3 \rangle 2.$ $temp'[p][t] = 0$ BY $\langle 2 \rangle 6,\ \langle 1 \rangle 8$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 3 \rangle 2,\ \langle 2 \rangle 1,\ \langle 2 \rangle 2,\ \langle 2 \rangle 3,\ \langle 2 \rangle 4,\ \langle 2 \rangle 5,\ SMTT(10)$
$\langle 2 \rangle 7.$ CASE $t \notin tt$
$\langle 3 \rangle 1.$ $gamma[t] = 0$ BY $\langle 2 \rangle 7,\ \langle 1 \rangle 7$
$\langle 3 \rangle 2.$ $temp'[p][t] = temp[p][t]$ BY $\langle 2 \rangle 7,\ \langle 1 \rangle 8$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 3 \rangle 2,\ \langle 2 \rangle 1,\ \langle 2 \rangle 2,\ \langle 2 \rangle 3,\ \langle 2 \rangle 4,\ \langle 2 \rangle 5,\ SMTT(10)$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 6,\ \langle 2 \rangle 7$

$\langle 1 \rangle 13.$ $NextSendUpdate\_State\_Conclusion(p,\ tt)!temp$
BY $\langle 1 \rangle 9,\ \langle 1 \rangle 10,\ \langle 1 \rangle 11$ DEF $gamma$

Type and value of msg' in relation to gamma

$\langle 1 \rangle$14. ASSUME NEW $q \in Proc$ PROVE $Append(msg[p][q], gamma) \in Seq(DeltaVecType)$
$\langle 2 \rangle$ DEFINE $msgpq \triangleq msg[p][q]$
$\langle 2 \rangle$ HIDE DEF $msgpq$
$\langle 2 \rangle$ SUFFICES $Append(msgpq, gamma) \in Seq(DeltaVecType)$ BY DEF $msgpq$
$\langle 2 \rangle$1. $msgpq \in Seq(DeltaVecType)$ BY $\langle 1 \rangle 2$ DEF $msgpq$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 1 \rangle 3$, $AppendProperties$

$\langle 1 \rangle$15. $msg' = [msg$ EXCEPT $![p] = [q \in Proc \mapsto Append(msg[p][q], gamma)]]$
BY DEF $NextSendUpdate\_WithPTT$, $gamma$

$\langle 1 \rangle$16. $msg' \in [Proc \rightarrow [Proc \rightarrow Seq(DeltaVecType)]]$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 14$, $\langle 1 \rangle 15$

$\langle 1 \rangle$17. ASSUME NEW $fp \in Proc$, NEW $fq \in Proc$, $fp = p$
PROVE $msg'[fp][fq] = Append(msg[fp][fq], gamma)$
BY $\langle 1 \rangle 2$, $\langle 1 \rangle 15$, $\langle 1 \rangle 16$, $\langle 1 \rangle 17$

$\langle 1 \rangle$18. ASSUME NEW $fp \in Proc$, NEW $fq \in Proc$, $fp \neq p$
PROVE UNCHANGED $msg[fp][fq]$
BY $\langle 1 \rangle 2$, $\langle 1 \rangle 15$, $\langle 1 \rangle 16$, $\langle 1 \rangle 18$

$\langle 1 \rangle$19. $NextSendUpdate\_State\_Conclusion(p, tt)!msg$
BY $\langle 1 \rangle 17$, $\langle 1 \rangle 18$ DEF $gamma$

Type and value of $glob'$

$\langle 1 \rangle$20. UNCHANGED $glob$ BY DEF $NextSendUpdate\_WithPTT$
$\langle 1 \rangle$21. $glob' \in [Proc \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 20$ DEF $InvType$

Type and value of $nrec'$

$\langle 1 \rangle$22. UNCHANGED $nrec$ BY DEF $NextSendUpdate\_WithPTT$
$\langle 1 \rangle$23. $nrec' \in CountVecType$ BY $\langle 1 \rangle 22$ DEF $InvType$

$\langle 1 \rangle$24. $NextCommon\_State\_Conclusion!$:
$\langle 2 \rangle$ USE DEF $NextSendUpdate\_WithPTT$
$\langle 2 \rangle$ USE DEF $NextCommon\_State\_Conclusion$
$\langle 2 \rangle$ QED BY $NextCommon\_State$

$IsFiniteTempProcs'$

$\langle 1 \rangle$25. $IsFiniteTempProcs'$
$\langle 2 \rangle$ DEFINE $FP \triangleq \{fp \in Proc : temp[fp] \neq DeltaVecZero\}$
$\langle 2 \rangle$1. $IsFiniteSet(FP)$ BY DEF $InvType$, $IsFiniteTempProcs$
$\langle 2 \rangle$2. $IsFiniteSet(\{p\})$ BY $FiniteSetSingleton$
$\langle 2 \rangle$3. $IsFiniteSet(FP \cup \{p\})$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $FiniteSetUnion$
$\langle 2 \rangle$4. $FP' \subseteq (FP \cup \{p\})$ BY $\langle 1 \rangle 11$
$\langle 2 \rangle$5. $IsFiniteSet(FP')$ BY $\langle 2 \rangle 3$, $\langle 2 \rangle 4$, $FiniteSetSubset$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 5$ DEF $IsFiniteTempProcs$

$IsFiniteMsgSenders'$

$\langle 1 \rangle 26.$ $IsFiniteMsgSenders'$
$\langle 2 \rangle$ SUFFICES ASSUME NEW $fq \in Proc$
PROVE $IsFiniteSet(\{fp \in Proc : msg'[fp][fq] \neq \langle\rangle\})$
BY DEF $IsFiniteMsgSenders$

$\langle 2 \rangle$ DEFINE $FP \triangleq \{fp \in Proc : msg[fp][fq] \neq \langle\rangle\}$

$\langle 2 \rangle 1.$ $IsFiniteMsgSenders$ BY DEF $InvType$
$\langle 2 \rangle 2.$ $IsFiniteSet(FP)$ BY $\langle 2 \rangle 1$ DEF $IsFiniteMsgSenders$
$\langle 2 \rangle 3.$ $IsFiniteSet(\{p\})$ BY $FiniteSetSingleton$
$\langle 2 \rangle 4.$ $IsFiniteSet(FP \cup \{p\})$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $FiniteSetUnion$
$\langle 2 \rangle 5.$ $FP' \subseteq FP \cup \{p\}$
$\langle 3 \rangle 1.$ SUFFICES ASSUME NEW $fp \in FP'$ PROVE $fp \in FP \lor fp = p$ OBVIOUS
$\langle 3 \rangle 2.$ CASE $fp = p$ BY $\langle 3 \rangle 2$
$\langle 3 \rangle 3.$ CASE $fp \neq p$
$\langle 4 \rangle 1.$ $fp \in Proc$ BY $\langle 3 \rangle 1$
$\langle 4 \rangle 2.$ UNCHANGED $msg[fp][fq]$ BY $\langle 4 \rangle 1$, $\langle 3 \rangle 3$, $\langle 1 \rangle 18$
$\langle 4 \rangle 3.$ $msg'[fp][fq] \neq \langle\rangle$ BY $\langle 3 \rangle 1$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 4 \rangle 3$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$, $\langle 3 \rangle 3$
$\langle 2 \rangle 6.$ $IsFiniteSet(FP')$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 5$, $FiniteSetSubset$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 6$

$\langle 1 \rangle 27.$ $InvType'$ BY $\langle 1 \rangle 6$, $\langle 1 \rangle 16$, $\langle 1 \rangle 21$, $\langle 1 \rangle 23$, $\langle 1 \rangle 24$, $\langle 1 \rangle 25$, $\langle 1 \rangle 26$ DEF $InvType$

Preservation of upright $temp[p]$

$\langle 1 \rangle 28.$ ASSUME $IsDeltaVecUpright(lleq, temp[p])$ PROVE $IsDeltaVecUpright(lleq, temp[p])'$

$\langle 2 \rangle$ DEFINE $tempp \triangleq temp[p]$
$\langle 2 \rangle$ HIDE DEF $tempp$

$\langle 2 \rangle$ SUFFICES $IsDeltaVecUpright(lleq, tempp')$ BY $\langle 1 \rangle 24$ DEF $tempp$

$\langle 2 \rangle 1.$ $tempp' = newtempp$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 5$ DEF $tempp$
$\langle 2 \rangle 2.$ $IsDeltaVecUpright(lleq, newtempp)$ BY DEF $NextSendUpdate\_WithPTT$, $newtempp$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$

$\langle 1 \rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$
$\langle 1 \rangle$ USE DEF $gamma$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 3$, $\langle 1 \rangle 12$, $\langle 1 \rangle 13$, $\langle 1 \rangle 19$, $\langle 1 \rangle 20$, $\langle 1 \rangle 22$, $\langle 1 \rangle 24$, $\langle 1 \rangle 27$, $\langle 1 \rangle 28$

How the $NextReceiveUpdate(p, q)$ action updates the state variables.

$NextReceiveUpdate\_State\_Conclusion(p,\ q) \triangleq$
  LET
    $kappa \triangleq NextReceiveUpdate\_Kappa(p,\ q)$
    $M \triangleq msg[p][q]$
  IN
  $\land\ M \neq \langle\rangle$

  $\land\ Len(M) \in Nat$
  $\land\ Len(M) \neq 0$
  $\land\ Len(M) > 0$

  $\land\ kappa = Head(M)$
  $\land\ kappa = M[1]$
  $\land\ kappa \in DeltaVecType$

  $\land\ glob::$
    $\forall\, fq \in Proc :$
      IF $fq = q$
      THEN
        $\land\ glob'[fq] = DeltaVecAdd(glob[fq],\ kappa)$
        $\land\ glob'[fq] = DeltaVecAdd(kappa,\ glob[fq])$
      ELSE UNCHANGED $glob[fq]$

  $\land\ msg::$
    $\forall\, fp \in Proc :$
    $\forall\, fq \in Proc :$
      IF $fp = p \land fq = q$
      THEN $msg'[fp][fq] = Tail(msg[fp][fq])$
      ELSE UNCHANGED $msg[fp][fq]$

  $\land$ UNCHANGED $temp$
  $\land$ UNCHANGED $nrec$
  $\land\ NextCommon\_State\_Conclusion\,!\,:$

  $\land\ InvType'$

THEOREM $NextReceiveUpdate\_State \triangleq$
  ASSUME
    NEW $p \in Proc$,
    NEW $q \in Proc$,
    $InvType$,
    $NextReceiveUpdate\_WithPQ(p,\ q)$
  PROVE
  $NextReceiveUpdate\_State\_Conclusion(p,\ q)$

PROOF

$\langle 1 \rangle$ USE DEF $NextReceiveUpdate\_Kappa$

$\langle 1 \rangle$ DEFINE $kappa \triangleq NextReceiveUpdate\_Kappa(p, q)$
$\langle 1 \rangle$ HIDE DEF $kappa$

$\langle 1 \rangle$ DEFINE $M \triangleq msg[p][q]$
$\langle 1 \rangle$ HIDE DEF M

$\langle 1 \rangle$ DEFINE $LenM \triangleq Len(M)$
$\langle 1 \rangle$ HIDE DEF $LenM$

Type and value of msg' in relation to kappa

$\langle 1 \rangle 1.$ $msg \in [Proc \rightarrow [Proc \rightarrow Seq(DeltaVecType)]]$ BY DEF $InvType$

$\langle 1 \rangle 2.$ $M \in Seq(DeltaVecType)$ BY $\langle 1 \rangle 1$ DEF M
$\langle 1 \rangle 3.$ $M \neq \langle \rangle$ BY DEF $NextReceiveUpdate\_WithPQ, M$

$\langle 1 \rangle 4.$ $LenM \in Nat$ BY $\langle 1 \rangle 2$, $LenInNat$ DEF $LenM$
$\langle 1 \rangle 5.$ $LenM \neq 0$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $EmptySeq$ DEF $LenM$
$\langle 1 \rangle 6.$ $LenM > 0$ BY $\langle 1 \rangle 4$, $\langle 1 \rangle 5$, $SMTT(10)$
$\langle 1 \rangle 7.$ $1 \in 1 \,..\, LenM$ BY $\langle 1 \rangle 4$, $\langle 1 \rangle 5$, $SMTT(10)$
$\langle 1 \rangle 8.$ $M \in [1 \,..\, LenM \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 2$, $LenAxiom$ DEF $LenM$
$\langle 1 \rangle 9.$ $M[1] \in DeltaVecType$ BY $\langle 1 \rangle 7$, $\langle 1 \rangle 8$
$\langle 1 \rangle 10.$ $Head(M) = M[1]$ BY $\langle 1 \rangle 2$, $HeadDef$
$\langle 1 \rangle 11.$ $Head(M) \in DeltaVecType$ BY $\langle 1 \rangle 9$, $\langle 1 \rangle 10$

$\langle 1 \rangle 12.$ $kappa = Head(M)$ BY DEF $kappa, M$
$\langle 1 \rangle 13.$ $kappa = M[1]$ BY $\langle 1 \rangle 10$, $\langle 1 \rangle 12$
$\langle 1 \rangle 14.$ $kappa \in DeltaVecType$ BY $\langle 1 \rangle 11$, $\langle 1 \rangle 12$

$\langle 1 \rangle 15.$ $Tail(M) \in Seq(DeltaVecType)$ BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $TailProp$

$\langle 1 \rangle 16.$ $msg' = [msg \text{ EXCEPT } ![p][q] = Tail(M)]$ BY DEF $NextReceiveUpdate\_WithPQ, M$

$\langle 1 \rangle 17.$ $msg' \in [Proc \rightarrow [Proc \rightarrow Seq(DeltaVecType)]]$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 15$, $\langle 1 \rangle 16$

$\langle 1 \rangle 18.$ $msg'[p][q] = Tail(M)$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 16$

$\langle 1 \rangle 19.$ $NextReceiveUpdate\_State\_Conclusion(p, q)!msg$
BY $\langle 1 \rangle 1$, $\langle 1 \rangle 16$ DEF M

Type and value of $glob'$ in relation to kappa

$\langle 1 \rangle 20.$ $glob \in [Proc \rightarrow DeltaVecType]$ BY DEF $InvType$

$\langle 1 \rangle 21.$ $DeltaVecAdd(glob[q], kappa) \in DeltaVecType$
BY $\langle 1 \rangle 14$, $\langle 1 \rangle 20$, $DeltaVecAddType$

$\langle 1 \rangle 22.$ $glob' = [glob \text{ EXCEPT } ![q] = DeltaVecAdd(glob[q], kappa)]$
BY DEF $NextReceiveUpdate\_WithPQ, kappa$

$\langle 1 \rangle 23.$ $glob' \in [Proc \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 20$, $\langle 1 \rangle 21$, $\langle 1 \rangle 22$

$\langle 1 \rangle 24.$ $glob'[q] = DeltaVecAdd(glob[q],\ kappa)$ BY $\langle 1 \rangle 20$, $\langle 1 \rangle 22$

$\langle 1 \rangle 25.$ $glob'[q] = DeltaVecAdd(kappa,\ glob[q])$
BY $\langle 1 \rangle 14$, $\langle 1 \rangle 20$, $\langle 1 \rangle 24$, $DeltaVecAddCommutative$

$\langle 1 \rangle 26.$ $NextReceiveUpdate\_State\_Conclusion(p,\ q)!glob$
BY $\langle 1 \rangle 20$, $\langle 1 \rangle 22$, $\langle 1 \rangle 24$, $\langle 1 \rangle 25$ DEF $kappa$

Type and value of $temp'$

$\langle 1 \rangle 27.$ UNCHANGED $temp$ BY DEF $NextReceiveUpdate\_WithPQ$
$\langle 1 \rangle 28.$ $temp' \in [Proc \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 27$ DEF $InvType$

Type and value of $nrec'$

$\langle 1 \rangle 29.$ UNCHANGED $nrec$ BY DEF $NextReceiveUpdate\_WithPQ$
$\langle 1 \rangle 30.$ $nrec' \in CountVecType$ BY $\langle 1 \rangle 29$ DEF $InvType$

$\langle 1 \rangle 31.$ $NextCommon\_State\_Conclusion!:$
$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_WithPQ$
$\langle 2 \rangle$ USE DEF $NextCommon\_State\_Conclusion$
$\langle 2 \rangle$ QED BY $NextCommon\_State$

$IsFiniteTempProcs'$

$\langle 1 \rangle 32.$ $IsFiniteTempProcs'$
$\langle 2 \rangle 1.$ $IsFiniteTempProcs$ BY DEF $InvType$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 1 \rangle 27$ DEF $IsFiniteTempProcs$

$IsFiniteMsgSenders'$

$\langle 1 \rangle 33.$ $IsFiniteMsgSenders'$
$\langle 2 \rangle$ SUFFICES ASSUME NEW $fq \in Proc$
PROVE $IsFiniteSet(\{fp \in Proc : msg'[fp][fq] \neq \langle\rangle\})$
BY DEF $IsFiniteMsgSenders$

$\langle 2 \rangle$ DEFINE $FP \triangleq \{fp \in Proc : msg[fp][fq] \neq \langle\rangle\}$

$\langle 2 \rangle 1.$ $IsFiniteMsgSenders$ BY DEF $InvType$
$\langle 2 \rangle 2.$ $IsFiniteSet(FP)$ BY $\langle 2 \rangle 1$ DEF $IsFiniteMsgSenders$
$\langle 2 \rangle 3.$ $IsFiniteSet(\{p\})$ BY $FiniteSetSingleton$
$\langle 2 \rangle 4.$ $IsFiniteSet(FP \cup \{p\})$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $FiniteSetUnion$
$\langle 2 \rangle 5.$ $FP' \subseteq FP \cup \{p\}$
$\langle 3 \rangle 1.$ SUFFICES ASSUME NEW $fp \in FP'$ PROVE $fp \in FP \lor fp = p$ OBVIOUS
$\langle 3 \rangle 2.$ CASE $fp = p$ BY $\langle 3 \rangle 2$
$\langle 3 \rangle 3.$ CASE $fp \neq p$
$\langle 4 \rangle 1.$ $fp \in Proc$ BY $\langle 3 \rangle 1$
$\langle 4 \rangle 2.$ UNCHANGED $msg[fp][fq]$ BY $\langle 4 \rangle 1$, $\langle 3 \rangle 3$, $\langle 1 \rangle 19$
$\langle 4 \rangle 3.$ $msg'[fp][fq] \neq \langle\rangle$ BY $\langle 3 \rangle 1$

$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 4 \rangle 3$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$, $\langle 3 \rangle 3$
$\langle 2 \rangle 6$. $IsFiniteSet(FP')$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 5$, $FiniteSetSubset$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 6$

$\langle 1 \rangle 34$. $InvType'$ BY $\langle 1 \rangle 17$, $\langle 1 \rangle 23$, $\langle 1 \rangle 28$, $\langle 1 \rangle 30$, $\langle 1 \rangle 31$, $\langle 1 \rangle 32$, $\langle 1 \rangle 33$ DEF $InvType$

$\langle 1 \rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$
$\langle 1 \rangle$ USE DEF $kappa$, M, $LenM$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $\langle 1 \rangle 5$, $\langle 1 \rangle 6$, $\langle 1 \rangle 12$, $\langle 1 \rangle 13$, $\langle 1 \rangle 14$, $\langle 1 \rangle 19$, $\langle 1 \rangle 26$, $\langle 1 \rangle 27$, $\langle 1 \rangle 29$, $\langle 1 \rangle 31$, $\langle 1 \rangle 34$

C.17 How the actions affect InfoAt

MODULE $NaiadClockProofAffectInfoAt$

EXTENDS $NaiadClockProofAffectState$

How the actions affect InfoAt.

The initial state for $InfoAt(fk, fp, fq)$.

$Init\_InfoAt\_Conclusion(fk, fp, fq) \triangleq$
  $InfoAt(fk, fp, fq) = DeltaVecZero$

THEOREM $Init\_InfoAt \triangleq$
  ASSUME
    NEW $fk \in Nat$,
    NEW $fp \in Proc$,
    NEW $fq \in Proc$,
    $InvType$,
    $Init$
  PROVE
  $Init\_InfoAt\_Conclusion(fk, fp, fq)$
PROOF
  $\langle 1 \rangle 1.$ $msg[fp][fq] \in Seq(DeltaVecType)$ BY DEF $InvType$
  $\langle 1 \rangle 2.$ $msg[fp][fq] = \langle\rangle$ BY DEF $Init$
  $\langle 1 \rangle 3.$ $Len(msg[fp][fq]) = 0$ BY $\langle 1 \rangle 2$, $EmptySeq$

  $\langle 1 \rangle$ DEFINE $LenM \triangleq Len(msg[fp][fq])$

  $\langle 1 \rangle 6.$ $LenM = 0$ BY $\langle 1 \rangle 3$

  $\langle 1 \rangle 7.$ $\neg(0 < fk \land fk \leq LenM)$
    $\langle 2 \rangle$ HIDE DEF $LenM$
    $\langle 2 \rangle$ QED BY $\langle 1 \rangle 6$, $SMTT(10)$

  $\langle 1 \rangle 9.$ $InfoAt(fk, fp, fq) = DeltaVecZero$ BY $\langle 1 \rangle 7$ DEF $InfoAt$

  $\langle 1 \rangle$ QED BY $\langle 1 \rangle 9$ DEF $Init\_InfoAt\_Conclusion$

What the $NextPerformOperation(p, c, r)$ action does to $InfoAt(fk, fp, fq)$.

$NextPerformOperation\_InfoAt\_Conclusion(fk, fp, fq, p, c, r) \triangleq$
  UNCHANGED $InfoAt(fk, fp, fq)$

THEOREM $NextPerformOperation\_InfoAt \triangleq$
  ASSUME
    NEW $fk \in Nat$,
    NEW $fp \in Proc$,
    NEW $fq \in Proc$,
    NEW $p \in Proc$,
    NEW $c \in PointToNat$,
    NEW $r \in PointToNat$,
    $InvType$,
    $NextPerformOperation\_WithPCR(p, c, r)$
  PROVE
  $NextPerformOperation\_InfoAt\_Conclusion(fk, fp, fq, p, c, r)$
PROOF
  $\langle 1 \rangle 1.$ $NextPerformOperation\_State\_Conclusion(p, c, r)$ BY $NextPerformOperation\_State$
  $\langle 1 \rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$

  $\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$, Isa DEF $InfoAt$, $NextPerformOperation\_InfoAt\_Conclusion$

What the $NextSendUpdate(p, tt)$ action does to $InfoAt(fk, fp, fq)$.

$NextSendUpdate\_InfoAt\_Conclusion(fk, fp, fq, p, tt) \triangleq$
  LET
    $gamma \triangleq NextSendUpdate\_Gamma(p, tt)$
    $len \triangleq Len(msg[fp][fq])$
  IN
  $\land fp \neq p \Rightarrow$ UNCHANGED $InfoAt(fk, fp, fq)$
  $\land fp = p \Rightarrow$
    $\land fk = len + 1 \Rightarrow InfoAt(fk, fp, fq)' = gamma$
    $\land fk \neq len + 1 \Rightarrow$ UNCHANGED $InfoAt(fk, fp, fq)$

THEOREM $NextSendUpdate\_InfoAt \triangleq$
  ASSUME
    NEW $fk \in Nat$,
    NEW $fp \in Proc$,

NEW $fq \in Proc$,
NEW $p \in Proc$,
NEW $tt \in$ SUBSET $Point$,
$InvType$,
$NextSendUpdate\_WithPTT(p, tt)$
PROVE
$NextSendUpdate\_InfoAt\_Conclusion(fk, fp, fq, p, tt)$
PROOF
$\langle 1 \rangle$ $InvInfoAtType$ BY $DeduceInvInfoAtType$

$\langle 1 \rangle 1.$ $NextSendUpdate\_State\_Conclusion(p, tt)$ BY $NextSendUpdate\_State$
$\langle 1 \rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$

$\langle 1 \rangle 2.$ CASE $fp \neq p$
$\langle 2 \rangle 1.$ UNCHANGED $msg[fp][fq]$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$
$\langle 2 \rangle 2.$ UNCHANGED $InfoAt(fk, fp, fq)$ BY $\langle 2 \rangle 1$ DEF $InfoAt$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $\langle 1 \rangle 2$ DEF $NextSendUpdate\_InfoAt\_Conclusion$

$\langle 1 \rangle 3.$ CASE $fp = p$
$\langle 2 \rangle$ USE DEF $NextSendUpdate\_Gamma$
$\langle 2 \rangle$ DEFINE $gamma \triangleq NextSendUpdate\_Gamma(p, tt)$
$\langle 2 \rangle$ HIDE DEF $gamma$

$\langle 2 \rangle 1.$ $gamma \in DeltaVecType$ BY $\langle 1 \rangle 1$ DEF $gamma$

$\langle 2 \rangle$ DEFINE $M \triangleq msg[fp][fq]$
$\langle 2 \rangle$ DEFINE $LenM \triangleq Len(M)$
$\langle 2 \rangle$ HIDE DEF M, $LenM$

$\langle 2 \rangle 2.$ $msg \in [Proc \rightarrow [Proc \rightarrow Seq(DeltaVecType)]]$ BY DEF $InvType$
$\langle 2 \rangle 3.$ $M \in Seq(DeltaVecType)$ BY $\langle 2 \rangle 2$ DEF M
$\langle 2 \rangle 4.$ $LenM \in Nat$ BY $\langle 2 \rangle 3$, $LenInNat$ DEF $LenM$
$\langle 2 \rangle 5.$ $M' = Append(M, gamma)$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 3$ DEF M, $gamma$
$\langle 2 \rangle 6.$ $M' \in Seq(DeltaVecType)$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 3$, $\langle 2 \rangle 5$, $AppendProperties$
$\langle 2 \rangle 7.$ $LenM' \in Nat$ BY $\langle 2 \rangle 6$, $LenInNat$ DEF $LenM$
$\langle 2 \rangle 8.$ $LenM' = LenM + 1$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 3$, $\langle 2 \rangle 5$, $AppendProperties$ DEF $LenM$

fk is outside the next state message queue.

$\langle 2 \rangle 9.$ CASE $fk = 0 \lor LenM + 1 < fk$
$\langle 3 \rangle 1.$ $fk \neq LenM + 1$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 9$, $SMTT(10)$
$\langle 3 \rangle 2.$ $\neg(0 < fk \land fk \leq LenM)$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 8$, $\langle 2 \rangle 9$, $SMTT(10)$
$\langle 3 \rangle 3.$ $\neg(0 < fk \land fk \leq LenM')$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 8$, $\langle 2 \rangle 9$, $SMTT(10)$
$\langle 3 \rangle 4.$ $InfoAt(fk, fp, fq) = DeltaVecZero$ BY $\langle 3 \rangle 2$ DEF $InfoAt$, $LenM$, M
$\langle 3 \rangle 5.$ $InfoAt(fk, fp, fq)' = DeltaVecZero$ BY $\langle 3 \rangle 3$ DEF $InfoAt$, $LenM$, M
$\langle 3 \rangle 6.$ UNCHANGED $InfoAt(fk, fp, fq)$ BY $\langle 3 \rangle 4$, $\langle 3 \rangle 5$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 6$, $\langle 1 \rangle 3$ DEF $NextSendUpdate\_InfoAt\_Conclusion$, $LenM$, M

fk is inside the previously existing elements on the next state message queue.

$\langle 2 \rangle 10.$ CASE $0 < fk \land fk < LenM + 1$
$\langle 3 \rangle 1.$ $InfoAt(fk, fp, fq) = M[fk]$
$\langle 4 \rangle 1.$ $0 < fk \land fk \leq LenM$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 10$, $SMTT(10)$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$ DEF $InvInfoAtType$, M, $LenM$
$\langle 3 \rangle 2.$ $InfoAt(fk, fp, fq)' = M[fk]'$
$\langle 4 \rangle 1.$ $0 < fk \land fk \leq LenM'$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 8$, $\langle 2 \rangle 10$, $SMTT(10)$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$ DEF $InfoAt$, M, $LenM$
$\langle 3 \rangle 3.$ $M[fk]' = M[fk]$
$\langle 4 \rangle 1.$ $fk \in 1 \,..\, LenM$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 10$, $SMTT(10)$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 2 \rangle 1$, $\langle 2 \rangle 3$, $\langle 2 \rangle 5$, $AppendPropertiesOldElems$ DEF $LenM$
$\langle 3 \rangle 4.$ UNCHANGED $InfoAt(fk, fp, fq)$ BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 3 \rangle 3$
$\langle 3 \rangle 5.$ $fk \neq LenM + 1$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 10$, $SMTT(10)$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 4$, $\langle 3 \rangle 5$, $\langle 1 \rangle 3$ DEF $NextSendUpdate\_InfoAt\_Conclusion$, $LenM$, M

fk is the appended element on the next state message queue.

$\langle 2 \rangle 11.$ CASE $fk = LenM + 1$
$\langle 3 \rangle 1.$ $M[fk]' = gamma$
$\langle 4 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 3$, $\langle 2 \rangle 5$, $\langle 2 \rangle 11$, $AppendPropertiesNewElem$ DEF $LenM$
$\langle 3 \rangle 2.$ $InfoAt(fk, fp, fq)' = M[fk]'$
$\langle 4 \rangle 1.$ $0 < fk \land fk \leq LenM'$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 8$, $\langle 2 \rangle 11$, $SMTT(10)$
$\langle 4 \rangle 2.$ $InfoAt(fk, fp, fq)' = M[fk]'$ BY $\langle 4 \rangle 1$ DEF $InfoAt$, M, $LenM$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 2$
$\langle 3 \rangle 3.$ $InfoAt(fk, fp, fq)' = gamma$ BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 3$, $\langle 2 \rangle 11$, $\langle 1 \rangle 3$ DEF $NextSendUpdate\_InfoAt\_Conclusion$, $LenM$, M, $gamma$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 4$, $\langle 2 \rangle 9$, $\langle 2 \rangle 10$, $\langle 2 \rangle 11$, $\langle 1 \rangle 3$, $SMTT(10)$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$

What the $NextReceiveUpdate(p, q)$ action does to $InfoAt(fk, fp, fq)$.

$NextReceiveUpdate\_InfoAt\_Conclusion(fk, fp, fq, p, q) \triangleq$
IF $fp = p \land fq = q \land fk > 0$
THEN
$InfoAt(fk, fp, fq)' = InfoAt(fk + 1, fp, fq)$
ELSE
UNCHANGED $InfoAt(fk, fp, fq)$

THEOREM $NextReceiveUpdate\_InfoAt \triangleq$
ASSUME
NEW $fk \in Nat$,

NEW $fp \in Proc$,
NEW $fq \in Proc$,
NEW $p \in Proc$,
NEW $q \in Proc$,
$InvType$,
$NextReceiveUpdate\_WithPQ(p, q)$
PROVE
$NextReceiveUpdate\_InfoAt\_Conclusion(fk, fp, fq, p, q)$
PROOF
$\langle 1 \rangle 1.$ $NextReceiveUpdate\_State\_Conclusion(p, q)$ BY $NextReceiveUpdate\_State$
$\langle 1 \rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$

$\langle 1 \rangle 2.$ ASSUME $fk = 0$ PROVE UNCHANGED $InfoAt(fk, fp, fq)$
$\langle 2 \rangle$ $\neg(0 < fk)$ BY $\langle 1 \rangle 2$, $SMTT(10)$
$\langle 2 \rangle$ QED BY DEF $InfoAt$

$\langle 1 \rangle 3.$ ASSUME $fp \neq p \lor fq \neq q$ PROVE UNCHANGED $InfoAt(fk, fp, fq)$
$\langle 2 \rangle 1.$ UNCHANGED $msg[fp][fq]$
$\langle 3 \rangle 1.$ USE $\langle 1 \rangle 1$, $\langle 1 \rangle 3$
$\langle 3 \rangle$ QED BY $ZenonT(20)$ sometimes zenon needs more time
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$ DEF $InfoAt$

$\langle 1 \rangle 4.$ ASSUME $fp = p, fq = q, fk > 0$ PROVE $InfoAt(fk, fp, fq)' = InfoAt(fk + 1, fp, fq)$
$\langle 2 \rangle$ DEFINE $M \triangleq msg[p][q]$
$\langle 2 \rangle$ DEFINE $LenM \triangleq Len(M)$
$\langle 2 \rangle$ HIDE DEF M, $LenM$

$\langle 2 \rangle 1.$ $msg \in [Proc \to [Proc \to Seq(DeltaVecType)]]$ BY DEF $InvType$
$\langle 2 \rangle 2.$ $M \neq \langle \rangle$ BY $\langle 1 \rangle 1$ DEF M
$\langle 2 \rangle 3.$ $M \in Seq(DeltaVecType)$ BY $\langle 2 \rangle 1$ DEF M
$\langle 2 \rangle 4.$ $LenM \in Nat$ BY $\langle 2 \rangle 3$, $LenInNat$ DEF $LenM$
$\langle 2 \rangle 5.$ $M' = Tail(M)$ BY $\langle 1 \rangle 1$ DEF M
$\langle 2 \rangle 6.$ $M' \in Seq(DeltaVecType)$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 2 \rangle 5$, $TailProp$
$\langle 2 \rangle 7.$ $LenM' \in Nat$ BY $\langle 2 \rangle 6$, $LenInNat$ DEF $LenM$
$\langle 2 \rangle 8.$ $LenM' = LenM - 1$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 2 \rangle 5$, $TailProp$ DEF $LenM$
$\langle 2 \rangle 9.$ $LenM = LenM' + 1$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 7$, $\langle 2 \rangle 8$, $SMTT(10)$

Within the sequence —– a simple consequence of $TailProp$.

$\langle 2 \rangle 17.$ CASE $fk \leq LenM'$
$\langle 3 \rangle 1.$ $fk \in 1 \,..\, LenM'$ BY $\langle 2 \rangle 7$, $\langle 2 \rangle 17$, $\langle 1 \rangle 4$, $SMTT(10)$
$\langle 3 \rangle 2.$ $M[fk]' = M[fk + 1]$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $\langle 2 \rangle 5$, $\langle 3 \rangle 1$, $TailProp$ DEF $LenM$
$\langle 3 \rangle 3.$ $fk + 1 \in 1 \,..\, LenM$ BY $\langle 3 \rangle 1$, $\langle 2 \rangle 7$, $\langle 2 \rangle 9$, $SMTT(10)$
$\langle 3 \rangle 4.$ $InfoAt(fk, fp, fq)' = M[fk]'$
$\langle 4 \rangle 1.$ $0 < fk \land fk \leq LenM'$ BY $\langle 3 \rangle 1$, $\langle 2 \rangle 7$, $SMTT(10)$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 1 \rangle 4$ DEF $InfoAt$, M, $LenM$
$\langle 3 \rangle 5.$ $InfoAt(fk + 1, fp, fq) = M[fk + 1]$
$\langle 4 \rangle 1.$ $0 < fk + 1 \land fk + 1 \leq LenM$ BY $\langle 3 \rangle 3$, $\langle 2 \rangle 4$, $SMTT(10)$

$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 1 \rangle 4$ DEF $InfoAt$, M, $LenM$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$, $\langle 3 \rangle 4$, $\langle 3 \rangle 5$

Off the end of the sequence —– both $InfoAt$ are 0.

$\langle 2 \rangle 18.$ CASE $fk > LenM'$
$\langle 3 \rangle 1.$ $InfoAt(fk, fp, fq)' = DeltaVecZero$
$\langle 4 \rangle 1.$ $\neg(0 < fk \land fk \leq LenM')$ BY $\langle 2 \rangle 7$, $\langle 2 \rangle 18$, $\langle 1 \rangle 4$, $SMTT(10)$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 1 \rangle 4$ DEF $InfoAt$, M, $LenM$
$\langle 3 \rangle 2.$ $InfoAt(fk + 1, fp, fq) = DeltaVecZero$
$\langle 4 \rangle 1.$ $fk + 1 \in Nat$ BY $SMTT(10)$
$\langle 4 \rangle 2.$ $\neg(0 < fk + 1 \land fk + 1 \leq LenM)$ BY $\langle 2 \rangle 7$, $\langle 2 \rangle 9$, $\langle 2 \rangle 18$, $SMTT(10)$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 1 \rangle 4$ DEF $InfoAt$, M, $LenM$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 7$, $\langle 2 \rangle 17$, $\langle 2 \rangle 18$, $\langle 1 \rangle 4$, $SMTT(10)$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $SMTT(10)$ DEF $NextReceiveUpdate\_InfoAt\_Conclusion$

What the $NextReceiveUpdate(p, q)$ action means about $InfoAt(1, p, q)$.

THEOREM $NextReceiveUpdate\_InfoAt1 \triangleq$
ASSUME
NEW $p \in Proc$,
NEW $q \in Proc$,
$InvType$,
$NextReceiveUpdate\_WithPQ(p, q)$
PROVE
$InfoAt(1, p, q) = NextReceiveUpdate\_Kappa(p, q)$
PROOF
$\langle 1 \rangle$ DEFINE $kappa \triangleq NextReceiveUpdate\_Kappa(p, q)$

$\langle 1 \rangle 1.$ $NextReceiveUpdate\_State\_Conclusion(p, q)$ BY $NextReceiveUpdate\_State$
$\langle 1 \rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$

$\langle 1 \rangle$ DEFINE $M \triangleq msg[p][q]$
$\langle 1 \rangle$ DEFINE $LenM \triangleq Len(M)$

$\langle 1 \rangle 2.$ $kappa = M[1]$ BY $\langle 1 \rangle 1$
$\langle 1 \rangle 3.$ $LenM \in Nat$ BY $\langle 1 \rangle 1$
$\langle 1 \rangle 4.$ $LenM > 0$ BY $\langle 1 \rangle 1$

$\langle 1 \rangle 5.$ $0 < 1 \land 1 \leq LenM$
$\langle 2 \rangle$ HIDE DEF $LenM$

$\langle 2 \rangle$ QED BY $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $SMTT(10)$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2$, $\langle 1 \rangle 5$ DEF $InfoAt$

C.18 How the actions affect IncomingInfo

MODULE $NaiadClockProofAffectIncomingInfo$

EXTENDS $NaiadClockProofAffectInfoAt$

How the actions affect IncomingInfo.

The initial state for $IncomingInfo(fk, fp, fq)$.

$Init\_IncomingInfo\_Conclusion(fk, fp, fq) \triangleq$
$\quad IncomingInfo(fk, fp, fq) = DeltaVecZero$

THEOREM $Init\_IncomingInfo \triangleq$
ASSUME
NEW $fk \in Nat$,
NEW $fp \in Proc$,
NEW $fq \in Proc$,
$InvType$,
$Init$
PROVE
$Init\_IncomingInfo\_Conclusion(fk, fp, fq)$
PROOF
$\langle 1 \rangle 1.$ $msg[fp][fq] \in Seq(DeltaVecType)$ BY DEF $InvType$
$\langle 1 \rangle 2.$ $msg[fp][fq] = \langle\rangle$ BY DEF $Init$
$\langle 1 \rangle$ DEFINE $sum \triangleq IncomingInfo(fk, fp, fq)!:!sum$
$\langle 1 \rangle 5.$ $sum = DeltaVecZero$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $DeltaVecSeqSkipSumEmpty$
$\langle 1 \rangle 6.$ $temp[fp] = DeltaVecZero$ BY DEF $Init$, $DeltaVecAddZero$
$\langle 1 \rangle 7.$ $DeltaVecAdd(sum, temp[fp]) = DeltaVecZero$
BY $\langle 1 \rangle 5$, $\langle 1 \rangle 6$, $DeltaVecZeroType$, $DeltaVecAddZero$
$\langle 1 \rangle 8.$ $IncomingInfo(fk, fp, fq) = DeltaVecZero$ BY $\langle 1 \rangle 7$ DEF $IncomingInfo$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 8$ DEF $Init\_IncomingInfo\_Conclusion$

What the $NextPerformOperation(p, c, r)$ action does to $IncomingInfo(fk, fp, fq)$.

$NextPerformOperation\_IncomingInfo\_Conclusion(fk, fp, fq, p, c, r) \triangleq$
LET
$delta \triangleq NextPerformOperation\_Delta(p, c, r)$
$II \triangleq IncomingInfo(fk, fp, fq)$
IN
IF $fp = p$ THEN $II' = DeltaVecAdd(II, delta)$ ELSE UNCHANGED II

THEOREM $NextPerformOperation\_IncomingInfo \triangleq$
ASSUME
NEW $fk \in Nat$,
NEW $fp \in Proc$,
NEW $fq \in Proc$,
NEW $p \in Proc$,
NEW $c \in PointToNat$,
NEW $r \in PointToNat$,
$InvType$,
$NextPerformOperation\_WithPCR(p, c, r)$
PROVE
$NextPerformOperation\_IncomingInfo\_Conclusion(fk, fp, fq, p, c, r)$
PROOF
$\langle 1 \rangle$ $InvIncomingInfoType$ BY $DeduceInvIncomingInfoType$
$\langle 1 \rangle 1.$ $NextPerformOperation\_State\_Conclusion(p, c, r)$ BY $NextPerformOperation\_State$
$\langle 1 \rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$

Not affected.

$\langle 1 \rangle 3.$ CASE $fp \neq p$
$\langle 2 \rangle 1.$ UNCHANGED $temp[fp]$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 3$
$\langle 2 \rangle 2.$ UNCHANGED msg BY $\langle 1 \rangle 1$
$\langle 2 \rangle$ USE DEF $IncomingInfo$
$\langle 2 \rangle$ USE DEF $NextPerformOperation\_IncomingInfo\_Conclusion$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $\langle 1 \rangle 3$

Affected.

$\langle 1 \rangle 4.$ CASE $fp = p$
$\langle 2 \rangle 1.$ $msg \in [Proc \rightarrow [Proc \rightarrow Seq(DeltaVecType)]]$ BY DEF $InvType$
$\langle 2 \rangle 2.$ $msg' = msg$ BY $\langle 1 \rangle 1$

The sum of delta vectors is associative. We have

```
incoming' =  summsg +      tempp'
          =  summsg + (tempp  + delta)
          = (summsg +  tempp) + delta
          =     incoming      + delta
```

$\langle 2 \rangle$ DEFINE $tempp \triangleq temp[p]$
$\langle 2 \rangle$ DEFINE $delta \triangleq NextPerformOperation\_Delta(p, c, r)$
$\langle 2 \rangle 3.$ $delta \in DeltaVecType$ BY $\langle 1 \rangle 1$
$\langle 2 \rangle 4.$ $tempp \in DeltaVecType$ BY DEF $InvType$

$\langle 2 \rangle 5.\ tempp' = DeltaVecAdd(tempp,\ delta)$ BY $\langle 1 \rangle 1,\ \langle 1 \rangle 4$
$\langle 2 \rangle$ DEFINE $summsg \triangleq IncomingInfo(fk,\ p,\ fq)!:!sum$
$\langle 2 \rangle 6.\ summsg \in DeltaVecType$ BY DEF $InvIncomingInfoType$
$\langle 2 \rangle 7.$ UNCHANGED $summsg$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle$ DEFINE $incoming \triangleq DeltaVecAdd(summsg,\ tempp)$
$\langle 2 \rangle 8.\ incoming \in DeltaVecType$ BY $\langle 2 \rangle 6,\ \langle 2 \rangle 4,\ DeltaVecAddType$
$\langle 2 \rangle 9.\ incoming' = DeltaVecAdd(summsg,\ tempp')$ BY $\langle 2 \rangle 7$
$\langle 2 \rangle 10.\ incoming' = DeltaVecAdd(summsg,\ DeltaVecAdd(tempp,\ delta))$ BY $\langle 2 \rangle 5,\ \langle 2 \rangle 9$
$\langle 2 \rangle 11.\ incoming' = DeltaVecAdd(DeltaVecAdd(summsg,\ tempp),\ delta)$
$\langle 3 \rangle$ HIDE DEF $incoming,\ summsg,\ tempp,\ delta$
$\langle 3 \rangle$ QED BY $\langle 2 \rangle 3,\ \langle 2 \rangle 4,\ \langle 2 \rangle 6,\ \langle 2 \rangle 10,\ DeltaVecAddAssociative$
$\langle 2 \rangle 12.\ incoming' = DeltaVecAdd(incoming,\ delta)$ BY $\langle 2 \rangle 11$
$\langle 2 \rangle$ USE DEF $IncomingInfo$
$\langle 2 \rangle$ USE DEF $NextPerformOperation\_IncomingInfo\_Conclusion$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 12,\ \langle 1 \rangle 4$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 3,\ \langle 1 \rangle 4$

What the $NextSendUpdate(p,\ tt)$ action does to $IncomingInfo(fk,\ fp,\ fq)$.

$NextSendUpdate\_IncomingInfo\_Conclusion(fk,\ fp,\ fq,\ p,\ tt) \triangleq$
LET
$II \triangleq IncomingInfo(fk,\ fp,\ fq)$
$len \triangleq Len(msg[fp][fq])$
IN
IF $fp = p \land fk > len$ THEN $II' = temp[fp]'$ ELSE UNCHANGED II

THEOREM $NextSendUpdate\_IncomingInfo \triangleq$
ASSUME
NEW $fk \in Nat$,
NEW $fp \in Proc$,
NEW $fq \in Proc$,
NEW $p \in Proc$,
NEW $tt \in$ SUBSET $Point$,
$InvType$,
$NextSendUpdate\_WithPTT(p,\ tt)$
PROVE
$NextSendUpdate\_IncomingInfo\_Conclusion(fk,\ fp,\ fq,\ p,\ tt)$
PROOF
$\langle 1 \rangle\ InvIncomingInfoType$ BY $DeduceInvIncomingInfoType$

$\langle 1\rangle 1.$ $NextSendUpdate\_State\_Conclusion(p, tt)$ BY $NextSendUpdate\_State$
$\langle 1\rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$

Not affected.

$\langle 1\rangle 2.$ CASE $fp \neq p$
$\langle 2\rangle 1.$ UNCHANGED $temp[fp]$ BY $\langle 1\rangle 1$, $\langle 1\rangle 2$
$\langle 2\rangle 2.$ UNCHANGED $msg[fp][fq]$ BY $\langle 1\rangle 1$, $\langle 1\rangle 2$
$\langle 2\rangle$ USE DEF $NextSendUpdate\_IncomingInfo\_Conclusion$
$\langle 2\rangle$ QED BY $\langle 2\rangle 1$, $\langle 2\rangle 2$, $\langle 1\rangle 2$ DEF $IncomingInfo$

Affected.

$\langle 1\rangle 3.$ CASE $fp = p$

Definitions for the current state.

$\langle 2\rangle$ DEFINE $CurrT \triangleq temp[p]$
$\langle 2\rangle$ DEFINE $CurrM \triangleq msg[p][fq]$
$\langle 2\rangle$ DEFINE $CurrSum \triangleq IncomingInfo(fk, p, fq)!:!sum$
$\langle 2\rangle$ DEFINE $CurrII \triangleq IncomingInfo(fk, p, fq)$
$\langle 2\rangle$ DEFINE $LenCurrM \triangleq Len(CurrM)$

$\langle 2\rangle 1.$ $CurrSum = DeltaVecSeqSkipSum(fk, CurrM)$ OBVIOUS
$\langle 2\rangle 2.$ $CurrII = DeltaVecAdd(CurrSum, CurrT)$ BY DEF $IncomingInfo$

$\langle 2\rangle 3.$ $CurrT \in DeltaVecType$ BY DEF $InvType$
$\langle 2\rangle 4.$ $CurrM \in Seq(DeltaVecType)$ BY DEF $InvType$
$\langle 2\rangle 5.$ $CurrSum \in DeltaVecType$ BY $\langle 2\rangle 4$, $DeltaVecSeqSkipSumType$
$\langle 2\rangle 6.$ $CurrII \in DeltaVecType$ BY $\langle 2\rangle 2$, $\langle 2\rangle 5$, $\langle 2\rangle 3$, $DeltaVecAddType$
$\langle 2\rangle 7.$ $LenCurrM \in Nat$ BY $\langle 2\rangle 4$, $LenInNat$

Definitions for the next state.

$\langle 2\rangle$ DEFINE $NextT \triangleq temp[p]'$
$\langle 2\rangle$ DEFINE $NextM \triangleq msg[p][fq]'$
$\langle 2\rangle$ DEFINE $NextSum \triangleq IncomingInfo(fk, p, fq)!:!sum'$
$\langle 2\rangle$ DEFINE $NextII \triangleq IncomingInfo(fk, p, fq)'$
$\langle 2\rangle$ DEFINE $LenNextM \triangleq Len(NextM)$

$\langle 2\rangle 8.$ $NextSum = DeltaVecSeqSkipSum(fk, NextM)$ OBVIOUS
$\langle 2\rangle 9.$ $NextII = DeltaVecAdd(NextSum, NextT)$ BY DEF $IncomingInfo$

$\langle 2\rangle 10.$ $NextT \in DeltaVecType$ BY $\langle 1\rangle 1$ DEF $InvType$
$\langle 2\rangle 11.$ $NextM \in Seq(DeltaVecType)$ BY $\langle 1\rangle 1$ DEF $InvType$
$\langle 2\rangle 12.$ $NextSum \in DeltaVecType$ BY $\langle 2\rangle 11$, $DeltaVecSeqSkipSumType$
$\langle 2\rangle 13.$ $NextII \in DeltaVecType$ BY $\langle 2\rangle 9$, $\langle 2\rangle 12$, $\langle 2\rangle 10$, $DeltaVecAddType$
$\langle 2\rangle 14.$ $LenNextM \in Nat$ BY $\langle 2\rangle 11$, $LenInNat$

Relation between current state and next state.

$\langle 2\rangle$ DEFINE $gamma \triangleq NextSendUpdate\_Gamma(p, tt)$
$\langle 2\rangle 15.$ $gamma \in DeltaVecType$ BY $\langle 1\rangle 1$

$\langle 2\rangle 16.$ $CurrT = DeltaVecAdd(gamma, NextT)$ BY $\langle 1\rangle 1$

$\langle 2 \rangle 17.$ $NextM = Append(CurrM, gamma)$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 3$
$\langle 2 \rangle 18.$ $LenNextM = LenCurrM + 1$
  $\langle 3 \rangle$ HIDE DEF $NextM$, $CurrM$
  $\langle 3 \rangle$ QED BY $\langle 2 \rangle 4$, $\langle 2 \rangle 15$, $\langle 2 \rangle 17$, $AppendProperties$

When $fk > LenCurrM$, we have $NextSum = 0$, which results in $NextII = NextT$.

$\langle 2 \rangle 20.$ ASSUME $fk > LenCurrM$ PROVE $NextII = NextT$
  $\langle 3 \rangle 1.$ $fk \geq LenNextM$
    $\langle 4 \rangle$ HIDE DEF $LenCurrM$, $LenNextM$
    $\langle 4 \rangle$ QED BY $\langle 2 \rangle 7$, $\langle 2 \rangle 14$, $\langle 2 \rangle 18$, $\langle 2 \rangle 20$, $SMTT(10)$
  $\langle 3 \rangle 2.$ $NextSum = DeltaVecZero$ BY $\langle 3 \rangle 1$, $\langle 2 \rangle 8$, $\langle 2 \rangle 11$, $DeltaVecSeqSkipSumSkipAll$
  $\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$, $\langle 2 \rangle 9$, $\langle 2 \rangle 10$, $DeltaVecAddZero$

When $fk \leq LenCurrM$, we have $NextSum = CurrSum + gamma$, which results in $NextII = CurrII$.

$\langle 2 \rangle 21.$ ASSUME $\neg(fk > LenCurrM)$ PROVE $NextII = CurrII$

The action adds gamma to sum.

  $\langle 3 \rangle 1.$ $NextSum = DeltaVecAdd(CurrSum, gamma)$
    $\langle 4 \rangle 1.$ $fk \leq LenCurrM$
      $\langle 5 \rangle$ HIDE DEF $LenCurrM$
      $\langle 5 \rangle$ QED BY $\langle 2 \rangle 7$, $\langle 2 \rangle 21$, $SMTT(10)$
    $\langle 4 \rangle$ HIDE DEF $NextSum$, $NextM$, $CurrSum$, $CurrM$, $gamma$
    $\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 2 \rangle 1$, $\langle 2 \rangle 4$, $\langle 2 \rangle 8$, $\langle 2 \rangle 15$, $\langle 2 \rangle 17$, $DeltaVecSeqSkipSumAppend$

Re-associate the sum of gamma vectors. We have

```
NextII =      NextSum     + NextT
       = (CurrSum +  gamma) + NextT
       =  CurrSum + (gamma  + NextT)
       =  CurrSum +    CurrT
       =          CurrII
```

  $\langle 3 \rangle$ HIDE DEF $CurrT$, $CurrSum$, $CurrII$
  $\langle 3 \rangle$ HIDE DEF $NextT$, $NextSum$, $NextII$
  $\langle 3 \rangle$ HIDE DEF $gamma$
  $\langle 3 \rangle 2.$ $NextII = DeltaVecAdd(DeltaVecAdd(CurrSum, gamma), NextT)$ BY $\langle 3 \rangle 1$, $\langle 2 \rangle 9$
  $\langle 3 \rangle 3.$ $NextII = DeltaVecAdd(CurrSum, DeltaVecAdd(gamma, NextT))$
      BY $\langle 3 \rangle 2$, $\langle 2 \rangle 5$, $\langle 2 \rangle 10$, $\langle 2 \rangle 15$, $DeltaVecAddAssociative$
  $\langle 3 \rangle 4.$ $NextII = DeltaVecAdd(CurrSum, CurrT)$ BY $\langle 3 \rangle 3$, $\langle 2 \rangle 16$
  $\langle 3 \rangle$ QED BY $\langle 3 \rangle 4$, $\langle 2 \rangle 2$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 20$, $\langle 2 \rangle 21$, $\langle 1 \rangle 3$ DEF $NextSendUpdate\_IncomingInfo\_Conclusion$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$

What the $NextReceiveUpdate(p, q)$ action does to $IncomingInfo(fk, fp, fq)$.

$NextReceiveUpdate\_IncomingInfo\_Conclusion(fk, fp, fq, p, q) \triangleq$
IF $fp = p \land fq = q$
THEN $IncomingInfo(fk, fp, fq)' = IncomingInfo(fk + 1, fp, fq)$
ELSE UNCHANGED $IncomingInfo(fk, fp, fq)$

THEOREM $NextReceiveUpdate\_IncomingInfo \triangleq$
ASSUME
NEW $fk \in Nat$,
NEW $fp \in Proc$,
NEW $fq \in Proc$,
NEW $p \in Proc$,
NEW $q \in Proc$,
$InvType$,
$NextReceiveUpdate\_WithPQ(p, q)$
PROVE
$NextReceiveUpdate\_IncomingInfo\_Conclusion(fk, fp, fq, p, q)$
PROOF
$\langle 1 \rangle 1.$ $NextReceiveUpdate\_State\_Conclusion(p, q)$ BY $NextReceiveUpdate\_State$
$\langle 1 \rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$

$\langle 1 \rangle$ $InvIncomingInfoType$ BY $DeduceInvIncomingInfoType$

$\langle 1 \rangle$ $InvType'$ BY $\langle 1 \rangle 1$
$\langle 1 \rangle$ $InvIncomingInfoType'$ BY $DeduceInvIncomingInfoType$

Not affected.

$\langle 1 \rangle 2.$ CASE $\neg(fp = p \land fq = q)$
$\langle 2 \rangle 1.$ UNCHANGED $temp$ BY $\langle 1 \rangle 1$
$\langle 2 \rangle 2.$ UNCHANGED $msg[fp][fq]$ BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$
$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_IncomingInfo\_Conclusion$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 2$, $\langle 2 \rangle 1$, $\langle 2 \rangle 2$ DEF $IncomingInfo$

Affected.

$\langle 1 \rangle 3.$ CASE $fp = p \land fq = q$

Definitions for the current state. Note that these reference $fk + 1$.

$\langle 2 \rangle$ DEFINE $CurrT \triangleq temp[p]$
$\langle 2 \rangle$ DEFINE $CurrM \triangleq msg[p][q]$
$\langle 2 \rangle$ DEFINE $CurrSum \triangleq IncomingInfo(fk + 1, p, q)!:!sum$
$\langle 2 \rangle$ DEFINE $CurrII \triangleq IncomingInfo(fk + 1, p, q)$

$\langle 2 \rangle 1.$ $CurrII = DeltaVecAdd(CurrSum, CurrT)$ BY DEF $IncomingInfo$
$\langle 2 \rangle 2.$ $CurrSum = DeltaVecSeqSkipSum(fk + 1, CurrM)$ OBVIOUS

$\langle 2 \rangle 3.$ $CurrT \in DeltaVecType$ BY DEF $InvType$
$\langle 2 \rangle 4.$ $CurrM \in Seq(DeltaVecType)$ BY DEF $InvType$
$\langle 2 \rangle 5.$ $CurrSum \in DeltaVecType$ BY DEF $InvIncomingInfoType$

$\langle 2 \rangle 6.$ $CurrII \in DeltaVecType$ BY DEF $InvIncomingInfoType$

$\langle 2 \rangle 7.$ $CurrM \neq \langle\rangle$ BY $\langle 1 \rangle 1$

Definitions for the next state.

$\langle 2 \rangle$ DEFINE $NextT \triangleq temp[p]'$
$\langle 2 \rangle$ DEFINE $NextM \triangleq msg[p][q]'$
$\langle 2 \rangle$ DEFINE $NextSum \triangleq IncomingInfo(fk, p, q)!::!sum'$
$\langle 2 \rangle$ DEFINE $NextII \triangleq IncomingInfo(fk, p, q)'$

$\langle 2 \rangle 8.$ $NextII = DeltaVecAdd(NextSum, NextT)$ BY DEF $IncomingInfo$
$\langle 2 \rangle 9.$ $NextSum = DeltaVecSeqSkipSum(fk, NextM)$ OBVIOUS

$\langle 2 \rangle 10.$ $NextT \in DeltaVecType$ BY DEF $InvType$
$\langle 2 \rangle 11.$ $NextM \in Seq(DeltaVecType)$ BY DEF $InvType$
$\langle 2 \rangle 12.$ $NextSum \in DeltaVecType$ BY DEF $InvIncomingInfoType$
$\langle 2 \rangle 13.$ $NextII \in DeltaVecType$ BY DEF $InvIncomingInfoType$

Relation between current state and next state.

$\langle 2 \rangle 14.$ $CurrT = NextT$ BY $\langle 1 \rangle 1$
$\langle 2 \rangle 15.$ $NextM = Tail(CurrM)$ BY $\langle 1 \rangle 1$
$\langle 2 \rangle 16.$ $NextSum = DeltaVecSeqSkipSum(fk, Tail(CurrM))$ BY $\langle 2 \rangle 9$, $\langle 2 \rangle 15$
$\langle 2 \rangle 17.$ $DeltaVecSeqSkipSum(fk, Tail(CurrM)) = DeltaVecSeqSkipSum(fk + 1, CurrM)$
  $\langle 3 \rangle$ HIDE DEF $CurrM$
  $\langle 3 \rangle$ QED BY $\langle 2 \rangle 4$, $\langle 2 \rangle 7$, $DeltaVecSeqSkipSumTail$
$\langle 2 \rangle 18.$ $NextSum = CurrSum$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 15$, $\langle 2 \rangle 16$, $\langle 2 \rangle 17$
$\langle 2 \rangle 19.$ $NextII = CurrII$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 8$, $\langle 2 \rangle 14$, $\langle 2 \rangle 18$
$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_IncomingInfo\_Conclusion$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 19$, $\langle 1 \rangle 3$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 2$, $\langle 1 \rangle 3$

C.19 How the actions affect GlobalIncomingInfo

MODULE $NaiadClockProofAffectGlobalIncomingInfo$

EXTENDS $NaiadClockProofAffectIncomingInfo$

How the actions affect GlobalIncomingInfo.

The initial state for $GlobalIncomingInfo(fk, fp, fq)$.

$Init\_GlobalIncomingInfo\_Conclusion(fk, fp, fq) \triangleq$
  LET
    $GII \triangleq GlobalIncomingInfo(fk, fp, fq)$
  IN
  $GII = DeltaVecZero$

THEOREM $Init\_GlobalIncomingInfo \triangleq$
  ASSUME
    NEW $fk \in Nat$,
    NEW $fp \in Proc$,
    NEW $fq \in Proc$,
    $InvType$,
    $Init$
  PROVE
  $Init\_GlobalIncomingInfo\_Conclusion(fk, fp, fq)$
PROOF
  $\langle 1 \rangle$ DEFINE $GII \triangleq GlobalIncomingInfo(fk, fp, fq)$
  $\langle 1 \rangle$ DEFINE $F \triangleq GlobalIncomingInfo(fk, fp, fq)! : !F$
  $\langle 1 \rangle$ HIDE DEF GII, F

  $\langle 1 \rangle$ $InvGlobalIncomingInfoType$ BY $DeduceInvGlobalIncomingInfoType$
  $\langle 1 \rangle 1.$ $F = GlobalIncomingInfo\_F(fk, fp, fq)$ BY DEF $GlobalIncomingInfo\_F, F$
  $\langle 1 \rangle 2.$ $F \in [Proc \rightarrow DeltaVecType]$ BY $\langle 1 \rangle 1$ DEF $InvGlobalIncomingInfoType$
  $\langle 1 \rangle 3.$ $DeltaVecFunHasFiniteNonZeroRange(F)$ BY $\langle 1 \rangle 1$ DEF $InvGlobalIncomingInfoType$
  $\langle 1 \rangle 4.$ ASSUME NEW $p \in Proc$ PROVE $F[p] = DeltaVecZero$
    $\langle 2 \rangle 1.$ ASSUME NEW $k \in Nat$ PROVE $IncomingInfo(k, p, fq) = DeltaVecZero$
      $\langle 3 \rangle 1.$ $Init\_IncomingInfo\_Conclusion(k, p, fq)$ BY $Init\_IncomingInfo$
      $\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$ DEF $Init\_IncomingInfo\_Conclusion$
    $\langle 2 \rangle 2.$ CASE $fp = p$
      $\langle 3 \rangle 1.$ $fk \in Nat$ OBVIOUS
      $\langle 3 \rangle$ QED BY $\langle 3 \rangle 1, \langle 2 \rangle 1, \langle 2 \rangle 2$ DEF F

$\langle 2\rangle 3.$ CASE $fp \neq p$
$\langle 3\rangle 1.$ $0 \in Nat$ OBVIOUS
$\langle 3\rangle$ QED BY $\langle 3\rangle 1$, $\langle 2\rangle 1$, $\langle 2\rangle 3$ DEF F
$\langle 2\rangle$ QED BY $\langle 2\rangle 2$, $\langle 2\rangle 3$
$\langle 1\rangle 5.$ $DeltaVecFunSum(F) = DeltaVecZero$ BY $\langle 1\rangle 2$, $\langle 1\rangle 3$, $\langle 1\rangle 4$, $DeltaVecFunSumAllZero$
$\langle 1\rangle 6.$ $GII = DeltaVecZero$ BY $\langle 1\rangle 5$ DEF $GlobalIncomingInfo$, GII, F
$\langle 1\rangle$ QED BY $\langle 1\rangle 6$ DEF $Init\_GlobalIncomingInfo\_Conclusion$, GII

What the $NextPerformOperation(p, c, r)$ action does to $GlobalIncomingInfo(0, fq, fq)$.

$NextPerformOperation\_GlobalIncomingInfo\_Conclusion(fq, p, c, r) \triangleq$
LET
$delta \triangleq NextPerformOperation\_Delta(p, c, r)$
$GII \triangleq GlobalIncomingInfo(0, fq, fq)$
IN
$GII' = DeltaVecAdd(GII, delta)$

THEOREM $NextPerformOperation\_GlobalIncomingInfo \triangleq$
ASSUME
NEW $fq \in Proc$,
NEW $p \in Proc$,
NEW $c \in PointToNat$,
NEW $r \in PointToNat$,
$InvType$,
$NextPerformOperation\_WithPCR(p, c, r)$
PROVE
$NextPerformOperation\_GlobalIncomingInfo\_Conclusion(fq, p, c, r)$
PROOF
$\langle 1\rangle$ $InvIncomingInfoType$ BY $DeduceInvIncomingInfoType$
$\langle 1\rangle$ $InvGlobalIncomingInfoType$ BY $DeduceInvGlobalIncomingInfoType$

$\langle 1\rangle$ DEFINE $delta \triangleq NextPerformOperation\_Delta(p, c, r)$
$\langle 1\rangle$ DEFINE $GII \triangleq GlobalIncomingInfo(0, fq, fq)$
$\langle 1\rangle$ DEFINE $F \triangleq GlobalIncomingInfo\_F(0, fq, fq)$

$\langle 1\rangle 1.$ $NextPerformOperation\_State\_Conclusion(p, c, r)$
BY $NextPerformOperation\_State$
$\langle 1\rangle 2.$ ASSUME NEW $k1 \in Nat$, NEW $p1 \in Proc$
PROVE $NextPerformOperation\_IncomingInfo\_Conclusion(k1, p1, fq, p, c, r)$
BY $NextPerformOperation\_IncomingInfo$

$\langle 1 \rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$
$\langle 1 \rangle$ USE DEF $NextPerformOperation\_IncomingInfo\_Conclusion$

$\langle 1 \rangle$ $InvType'$ BY $\langle 1 \rangle 1$
$\langle 1 \rangle$ $InvIncomingInfoType'$ BY $DeduceInvIncomingInfoType$
$\langle 1 \rangle$ $InvGlobalIncomingInfoType'$ BY $DeduceInvGlobalIncomingInfoType$

$\langle 1 \rangle 3.$ $delta \in DeltaVecType$ BY $\langle 1 \rangle 1$ DEF $delta$

$\langle 1 \rangle 4.$ $F \in [Proc \rightarrow DeltaVecType]$ BY DEF $InvGlobalIncomingInfoType$
$\langle 1 \rangle 5.$ $F' \in [Proc \rightarrow DeltaVecType]$ BY DEF $InvGlobalIncomingInfoType$

$\langle 1 \rangle 6.$ ASSUME NEW $p1 \in Proc,\ p1 \neq p$ PROVE $F'[p1] = F[p1]$
  $\langle 2 \rangle 1.$ $0 \in Nat$ OBVIOUS
  $\langle 2 \rangle 2.$ UNCHANGED $IncomingInfo(0,\ p1,\ fq)$ BY $\langle 2 \rangle 1,\ \langle 1 \rangle 2,\ \langle 1 \rangle 6$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 2$ DEF $GlobalIncomingInfo\_F$

$\langle 1 \rangle 7.$ $F'[p] = DeltaVecAdd(F[p],\ delta)$
  $\langle 2 \rangle 1.$ $0 \in Nat$ OBVIOUS
  $\langle 2 \rangle 2.$ $IncomingInfo(0,\ p,\ fq)' = DeltaVecAdd(IncomingInfo(0,\ p,\ fq),\ delta)$ BY $\langle 2 \rangle 1,\ \langle 1 \rangle 2$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1,\ \langle 2 \rangle 2$ DEF $GlobalIncomingInfo\_F$

$\langle 1 \rangle 8.$ $F' = [F$ EXCEPT $![p] = DeltaVecAdd(@,\ delta)]$ BY $\langle 1 \rangle 4,\ \langle 1 \rangle 5,\ \langle 1 \rangle 6,\ \langle 1 \rangle 7$

$\langle 1 \rangle 9.$ $F' = DeltaVecFunAddAt(F,\ p,\ delta)$ BY $\langle 1 \rangle 8$ DEF $DeltaVecFunAddAt$

$\langle 1 \rangle 10.$ $GII' = DeltaVecAdd(GII,\ delta)$
  $\langle 2 \rangle$ $DeltaVecFunSumAddAt\_Conclusion(F,\ p,\ delta)$
    $\langle 3 \rangle$ $DeltaVecFunSumAddAt\_Hypothesis(F,\ p,\ delta)$
      $\langle 4 \rangle$ $DeltaVecFunHasFiniteNonZeroRange(F)$ BY DEF $InvGlobalIncomingInfoType$
      $\langle 4 \rangle$ QED BY $\langle 1 \rangle 3,\ \langle 1 \rangle 4$ DEF $DeltaVecFunSumAddAt\_Hypothesis$
    $\langle 3 \rangle$ QED BY $DeltaVecFunSumAddAt$
  $\langle 2 \rangle$ $GII = DeltaVecFunSum(F)$ BY DEF $GlobalIncomingInfo,\ GlobalIncomingInfo\_F$
  $\langle 2 \rangle$ $GII' = DeltaVecFunSum(F')$ BY DEF $GlobalIncomingInfo,\ GlobalIncomingInfo\_F$
  $\langle 2 \rangle$ QED BY $\langle 1 \rangle 9$ DEF $DeltaVecFunSumAddAt\_Conclusion$

$\langle 1 \rangle$ USE DEF $NextPerformOperation\_GlobalIncomingInfo\_Conclusion$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 10$

What the $NextSendUpdate(p,\ tt)$ action does to $GlobalIncomingInfo(0,\ fq,\ fq)$.

$NextSendUpdate\_GlobalIncomingInfo\_Conclusion(fq,\ p,\ tt) \triangleq$
  LET
    $GII \triangleq GlobalIncomingInfo(0,\ fq,\ fq)$

IN
UNCHANGED GII

THEOREM $NextSendUpdate\_GlobalIncomingInfo \triangleq$
ASSUME
NEW $fq \in Proc$,
NEW $p \in Proc$,
NEW $tt \in$ SUBSET $Point$,
$InvType$,
$NextSendUpdate\_WithPTT(p, tt)$
PROVE
$NextSendUpdate\_GlobalIncomingInfo\_Conclusion(fq, p, tt)$
PROOF
$\langle 1 \rangle$ $InvIncomingInfoType$ BY $DeduceInvIncomingInfoType$
$\langle 1 \rangle$ $InvGlobalIncomingInfoType$ BY $DeduceInvGlobalIncomingInfoType$

$\langle 1 \rangle$ DEFINE $GII \triangleq GlobalIncomingInfo(0, fq, fq)$
$\langle 1 \rangle$ DEFINE $F \triangleq GlobalIncomingInfo\_F(0, fq, fq)$

$\langle 1 \rangle 1.$ $NextSendUpdate\_State\_Conclusion(p, tt)$
BY $NextSendUpdate\_State$
$\langle 1 \rangle 2.$ ASSUME NEW $k1 \in Nat$, NEW $p1 \in Proc$
PROVE $NextSendUpdate\_IncomingInfo\_Conclusion(k1, p1, fq, p, tt)$
BY $NextSendUpdate\_IncomingInfo$

$\langle 1 \rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$
$\langle 1 \rangle$ USE DEF $NextSendUpdate\_IncomingInfo\_Conclusion$

$\langle 1 \rangle$ $InvType'$ BY $\langle 1 \rangle 1$
$\langle 1 \rangle$ $InvIncomingInfoType'$ BY $DeduceInvIncomingInfoType$
$\langle 1 \rangle$ $InvGlobalIncomingInfoType'$ BY $DeduceInvGlobalIncomingInfoType$

$\langle 1 \rangle 3.$ UNCHANGED F
$\langle 2 \rangle$ ASSUME NEW $p1 \in Proc$ PROVE UNCHANGED $IncomingInfo(0, p1, fq)$
$\langle 3 \rangle$ DEFINE $M \triangleq msg[p1][fq]$
$\langle 3 \rangle$ DEFINE $LenM \triangleq Len(M)$
$\langle 3 \rangle 1.$ $M \in Seq(DeltaVecType)$ BY DEF $InvType$
$\langle 3 \rangle 2.$ $LenM \in Nat$ BY $\langle 3 \rangle 1$, $LenInNat$
$\langle 3 \rangle 3.$ $\neg(0 > LenM)$
$\langle 4 \rangle$ HIDE DEF $LenM$
$\langle 4 \rangle$ QED BY $\langle 3 \rangle 2$, $SMTT(10)$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 3$, $\langle 1 \rangle 2$
$\langle 2 \rangle$ QED BY DEF $GlobalIncomingInfo\_F$

$\langle 1 \rangle 4.$ UNCHANGED GII
$\langle 2 \rangle$ $GII = DeltaVecFunSum(F)$ BY DEF $GlobalIncomingInfo$, $GlobalIncomingInfo\_F$
$\langle 2 \rangle$ $GII' = DeltaVecFunSum(F')$ BY DEF $GlobalIncomingInfo$, $GlobalIncomingInfo\_F$

$\langle 2 \rangle$ QED BY $\langle 1 \rangle 3$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 4$ DEF $NextSendUpdate\_GlobalIncomingInfo\_Conclusion$

What the $NextReceiveUpdate(p, q)$ action does to $GlobalIncomingInfo(0, fq, fq)$.

$NextReceiveUpdate\_GlobalIncomingInfo\_Conclusion(fq, p, q) \triangleq$
LET
$GII \triangleq GlobalIncomingInfo(0, fq, fq)$
$kappa \triangleq NextReceiveUpdate\_Kappa(p, q)$
$negkappa \triangleq DeltaVecNeg(kappa)$
IN
IF $fq = q$
THEN
$\land GII = DeltaVecAdd(GII', kappa)$ looking backward
$\land GII' = DeltaVecAdd(GII, negkappa)$ looking forward
ELSE UNCHANGED GII

THEOREM $NextReceiveUpdate\_GlobalIncomingInfo \triangleq$
ASSUME
NEW $fq \in Proc$,
NEW $p \in Proc$,
NEW $q \in Proc$,
$InvType$,
$NextReceiveUpdate\_WithPQ(p, q)$
PROVE
$NextReceiveUpdate\_GlobalIncomingInfo\_Conclusion(fq, p, q)$
PROOF
$\langle 1 \rangle$ $InvIncomingInfoType$ BY $DeduceInvIncomingInfoType$
$\langle 1 \rangle$ $InvGlobalIncomingInfoType$ BY $DeduceInvGlobalIncomingInfoType$

$\langle 1 \rangle$ DEFINE $GII \triangleq GlobalIncomingInfo(0, fq, fq)$
$\langle 1 \rangle$ DEFINE $kappa \triangleq NextReceiveUpdate\_Kappa(p, q)$
$\langle 1 \rangle$ DEFINE $negkappa \triangleq DeltaVecNeg(kappa)$
$\langle 1 \rangle$ DEFINE $F \triangleq GlobalIncomingInfo\_F(0, fq, fq)$

$\langle 1 \rangle$ DEFINE $Add(a, b) \triangleq DeltaVecAdd(a, b)$ a local abbreviation
$\langle 1 \rangle$ DEFINE $Zero \triangleq DeltaVecZero$ a local abbreviation

$\langle 1 \rangle 1.$ $NextReceiveUpdate\_State\_Conclusion(p, q)$
BY $NextReceiveUpdate\_State$

$\langle 1\rangle 2.$ ASSUME NEW $k1 \in Nat$, NEW $p1 \in Proc$
PROVE $NextReceiveUpdate\_IncomingInfo\_Conclusion(k1, p1, fq, p, q)$
BY $NextReceiveUpdate\_IncomingInfo$

$\langle 1\rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$
$\langle 1\rangle$ USE DEF $NextReceiveUpdate\_IncomingInfo\_Conclusion$

$\langle 1\rangle$ $InvType'$ BY $\langle 1\rangle 1$
$\langle 1\rangle$ $InvIncomingInfoType'$ BY $DeduceInvIncomingInfoType$
$\langle 1\rangle$ $InvGlobalIncomingInfoType'$ BY $DeduceInvGlobalIncomingInfoType$

$\langle 1\rangle 3.$ $GII \in DeltaVecType$ BY DEF $InvGlobalIncomingInfoType$
$\langle 1\rangle 4.$ $kappa \in DeltaVecType$ BY $\langle 1\rangle 1$
$\langle 1\rangle 5.$ $negkappa \in DeltaVecType$ BY $\langle 1\rangle 4$, $DeltaVecNegType$

$\langle 1\rangle 6.$ $GII = DeltaVecFunSum(F)$ BY DEF $GlobalIncomingInfo$, $GlobalIncomingInfo\_F$
$\langle 1\rangle 7.$ $GII' = DeltaVecFunSum(F')$ BY DEF $GlobalIncomingInfo$, $GlobalIncomingInfo\_F$
$\langle 1\rangle 8.$ $DeltaVecFunHasFiniteNonZeroRange(F)$ BY DEF $InvGlobalIncomingInfoType$
$\langle 1\rangle 9.$ $F \in [Proc \rightarrow DeltaVecType]$ BY DEF $InvGlobalIncomingInfoType$

No change.

$\langle 1\rangle 10.$ ASSUME $fq \neq q$ PROVE UNCHANGED GII

$\langle 2\rangle 1.$ UNCHANGED F
$\langle 3\rangle 1.$ ASSUME NEW $p1 \in Proc$ PROVE UNCHANGED $IncomingInfo(0, p1, fq)$
BY $\langle 1\rangle 2$, $\langle 1\rangle 10$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1$ DEF $GlobalIncomingInfo\_F$

$\langle 2\rangle$ QED BY $\langle 2\rangle 1$, $\langle 1\rangle 6$, $\langle 1\rangle 7$

Change.

$\langle 1\rangle 11.$ ASSUME $fq = q$ PROVE $GII' = DeltaVecAdd(GII, negkappa)$

$\langle 2\rangle 1.$ $F' = DeltaVecFunAddAt(F, p, negkappa)$

$\langle 3\rangle 1.$ ASSUME NEW $p1 \in Proc$, $p1 \neq p$ PROVE UNCHANGED $IncomingInfo(0, p1, fq)$
$\langle 4\rangle$ QED BY $\langle 3\rangle 1$, $\langle 1\rangle 2$

$\langle 3\rangle$ DEFINE $II0 \triangleq IncomingInfo(0, p, fq)$
$\langle 3\rangle$ DEFINE $II1 \triangleq IncomingInfo(1, p, fq)$

$\langle 3\rangle$ SUFFICES $II0' = Add(II0, negkappa)$
$\langle 4\rangle$ QED BY $\langle 3\rangle 1$ DEF $DeltaVecFunAddAt$, $GlobalIncomingInfo\_F$

$\langle 3\rangle 2.$ $0 \in Nat$ OBVIOUS
$\langle 3\rangle 3.$ $1 \in Nat$ OBVIOUS
$\langle 3\rangle 4.$ $0 + 1 = 1$ BY $SMTT(10)$
$\langle 3\rangle 5.$ $II0 \in DeltaVecType$ BY $\langle 3\rangle 2$ DEF $InvIncomingInfoType$

$\langle 3\rangle 6.\ II1 \in DeltaVecType$ BY $\langle 3\rangle 3$ DEF $InvIncomingInfoType$
$\langle 3\rangle 7.\ II0' = II1$ BY $\langle 3\rangle 4,\ \langle 1\rangle 2,\ \langle 1\rangle 11$
$\langle 3\rangle 8.\ II0 = Add(II1,\ kappa)$
$\langle 4\rangle$ DEFINE $M \triangleq msg[p][fq]$
$\langle 4\rangle$ DEFINE $LenM \triangleq Len(M)$
$\langle 4\rangle 1.\ kappa = M[1]$ BY $\langle 1\rangle 1,\ \langle 1\rangle 11$
$\langle 4\rangle 2.\ 1 \leq LenM$
$\langle 5\rangle 1.\ LenM \in Nat$ BY $\langle 1\rangle 1,\ \langle 1\rangle 11$
$\langle 5\rangle 2.\ LenM \neq 0$ BY $\langle 1\rangle 1,\ \langle 1\rangle 11$
$\langle 5\rangle$ HIDE DEF $LenM$
$\langle 5\rangle$ QED BY $\langle 5\rangle 1,\ \langle 5\rangle 2,\ SMTT(10)$
$\langle 4\rangle$ DEFINE $SS0 \triangleq DeltaVecSeqSkipSum(0,\ M)$
$\langle 4\rangle$ DEFINE $SS1 \triangleq DeltaVecSeqSkipSum(1,\ M)$
$\langle 4\rangle 3.\ M \in Seq(DeltaVecType)$
$\langle 5\rangle$ USE DEF $InvType$
$\langle 5\rangle$ QED BY $ZenonT(20)$ sometimes zenon needs more time
$\langle 4\rangle 4.\ SS0 \in DeltaVecType$ BY $\langle 4\rangle 3,\ \langle 3\rangle 2,\ DeltaVecSeqSkipSumType$
$\langle 4\rangle 5.\ SS1 \in DeltaVecType$ BY $\langle 4\rangle 3,\ \langle 3\rangle 3,\ DeltaVecSeqSkipSumType$
$\langle 4\rangle 6.\ SS0 = Add(SS1,\ kappa)$
$\langle 5\rangle$ HIDE DEF $kappa,\ M$
$\langle 5\rangle$ QED BY $\langle 4\rangle 1,\ \langle 4\rangle 2,\ \langle 4\rangle 3,\ \langle 3\rangle 4,\ DeltaVecSeqSkipSumNext$
$\langle 4\rangle$ DEFINE $tempp \triangleq temp[p]$
$\langle 4\rangle 7.\ tempp \in DeltaVecType$ BY DEF $InvType$
$\langle 4\rangle 8.\ II0 = Add(SS0,\ tempp)$ BY DEF $IncomingInfo$
$\langle 4\rangle 9.\ II1 = Add(SS1,\ tempp)$ BY DEF $IncomingInfo$
$\langle 4\rangle$ QED
$\langle 5\rangle$ HIDE DEF $SS0,\ SS1,\ II0,\ II1,\ tempp,\ kappa$
$\langle 5\rangle$ USE $\langle 4\rangle 4,\ \langle 4\rangle 5,\ \langle 4\rangle 7,\ \langle 3\rangle 5,\ \langle 3\rangle 6,\ \langle 1\rangle 4$
$\langle 5\rangle 1.\ II0 = Add(Add(SS1,\ kappa),\ tempp)$ BY $\langle 4\rangle 6,\ \langle 4\rangle 8$
$\langle 5\rangle 2.\ II0 = Add(SS1,\ Add(kappa,\ tempp))$ BY $\langle 5\rangle 1,\ DeltaVecAddAssociative$
$\langle 5\rangle 3.\ II0 = Add(SS1,\ Add(tempp,\ kappa))$ BY $\langle 5\rangle 2,\ DeltaVecAddCommutative$
$\langle 5\rangle 4.\ II0 = Add(Add(SS1,\ tempp),\ kappa)$ BY $\langle 5\rangle 3,\ DeltaVecAddAssociative$
$\langle 5\rangle 5.\ II0 = Add(II1,\ kappa)$ BY $\langle 5\rangle 4,\ \langle 4\rangle 9$
$\langle 5\rangle$ QED BY $\langle 5\rangle 5$
$\langle 3\rangle 9.\ II1 = Add(II0,\ negkappa)$
$\langle 4\rangle$ HIDE DEF $kappa,\ negkappa,\ II0,\ II1$
$\langle 4\rangle$ USE $\langle 3\rangle 5,\ \langle 3\rangle 6,\ \langle 1\rangle 4,\ \langle 1\rangle 5$
$\langle 4\rangle 2.\ Zero = Add(kappa,\ negkappa)$ BY $DeltaVecAddNeg$ DEF $negkappa$
$\langle 4\rangle 3.\ II1 = Add(II1,\ Add(kappa,\ negkappa))$ BY $\langle 4\rangle 2,\ DeltaVecAddZero$
$\langle 4\rangle 4.\ II1 = Add(Add(II1,\ kappa),\ negkappa)$ BY $\langle 4\rangle 3,\ DeltaVecAddAssociative$
$\langle 4\rangle 5.\ II1 = Add(II0,\ negkappa)$ BY $\langle 4\rangle 4,\ \langle 3\rangle 8$
$\langle 4\rangle$ QED BY $\langle 4\rangle 5$
$\langle 3\rangle$ QED BY $\langle 3\rangle 7,\ \langle 3\rangle 9$

$\langle 2\rangle 2.\ DeltaVecFunSumAddAt\_Conclusion(F,\ p,\ negkappa)$
$\langle 3\rangle$ HIDE DEF $F,\ negkappa$

$\langle 3 \rangle$ USE $\langle 1 \rangle 5$, $\langle 1 \rangle 8$, $\langle 1 \rangle 9$
$\langle 3 \rangle$ $DeltaVecFunSumAddAt\_Hypothesis(F, p, negkappa)$
$\langle 4 \rangle$ QED BY DEF $DeltaVecFunSumAddAt\_Hypothesis$
$\langle 3 \rangle$ QED BY $DeltaVecFunSumAddAt$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 2$, $\langle 1 \rangle 6$, $\langle 1 \rangle 7$, $\langle 1 \rangle 9$ DEF $DeltaVecFunSumAddAt\_Conclusion$

$\langle 1 \rangle 12.$ ASSUME $fq = q$ PROVE $GII = Add(GII', kappa)$
$\langle 2 \rangle$ HIDE DEF $kappa$
$\langle 2 \rangle$ HIDE DEF $negkappa$
$\langle 2 \rangle$ HIDE DEF GII
$\langle 2 \rangle$ USE $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $\langle 1 \rangle 5$
$\langle 2 \rangle 1.$ $Zero = Add(negkappa, kappa)$ BY $\langle 1 \rangle 4$, $DeltaVecAddNeg$ DEF $negkappa$
$\langle 2 \rangle 2.$ $GII = Add(GII, Add(negkappa, kappa))$ BY $\langle 2 \rangle 1$, $DeltaVecAddZero$
$\langle 2 \rangle 3.$ $GII = Add(Add(GII, negkappa), kappa)$ BY $\langle 2 \rangle 2$, $DeltaVecAddAssociative$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 3$, $\langle 1 \rangle 11$, $\langle 1 \rangle 12$

$\langle 1 \rangle$ USE DEF $NextReceiveUpdate\_GlobalIncomingInfo\_Conclusion$
$\langle 1 \rangle$ USE DEF $GlobalIncomingInfo$, $GlobalIncomingInfo\_F$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 10$, $\langle 1 \rangle 11$, $\langle 1 \rangle 12$

What the $NextReceiveUpdate(p, q)$ action does to $GlobalIncomingInfo(fk, p, q)$.

THEOREM $NextReceiveUpdate\_GlobalIncomingInfo1 \triangleq$
ASSUME
NEW $fk \in Nat$,
NEW $p \in Proc$,
NEW $q \in Proc$,
$InvType$,
$NextReceiveUpdate\_WithPQ(p, q)$
PROVE
$GlobalIncomingInfo(fk, p, q)' = GlobalIncomingInfo(fk + 1, p, q)$
PROOF
$\langle 1 \rangle 1.$ $IncomingInfo(fk, p, q)' = IncomingInfo(fk + 1, p, q)$
$\langle 2 \rangle 1.$ $NextReceiveUpdate\_IncomingInfo\_Conclusion(fk, p, q, p, q)$
BY $NextReceiveUpdate\_IncomingInfo$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$ DEF $NextReceiveUpdate\_IncomingInfo\_Conclusion$

$\langle 1 \rangle 2.$ ASSUME NEW $p1 \in Proc$, $p1 \neq p$ PROVE UNCHANGED $IncomingInfo(0, p1, q)$
$\langle 2 \rangle 1.$ $NextReceiveUpdate\_IncomingInfo\_Conclusion(0, p1, q, p, q)$
BY $NextReceiveUpdate\_IncomingInfo$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 1 \rangle 2$ DEF $NextReceiveUpdate\_IncomingInfo\_Conclusion$

$\langle 1 \rangle 3$. $GlobalIncomingInfo\_F(fk,\ p,\ q)' = GlobalIncomingInfo\_F(fk + 1,\ p,\ q)$
BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$ DEF $GlobalIncomingInfo\_F$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 3$ DEF $GlobalIncomingInfo$, $GlobalIncomingInfo\_F$

C.20 Proof of invariant InvType

MODULE $NaiadClockProofInvType$

EXTENDS $NaiadClockProofAffectGlobalIncomingInfo$

Proof of invariant InvType.

$InvType$ holds in the initial state.

THEOREM $ThmInitInvType \triangleq$
$Init \Rightarrow InvType$
PROOF
$\langle 1 \rangle$ SUFFICES ASSUME $Init$PROVE $InvType$ OBVIOUS

$\langle 1 \rangle 1.$ $lleq \in PointRelationType$ BY DEF $Init$
$\langle 1 \rangle 2.$ $IsPartialOrder(lleq)$ BY DEF $Init$

$\langle 1 \rangle 3.$ $nrec \in CountVecType$ BY $AssumePointToNat$ DEF $Init, CountVecType$

$\langle 1 \rangle 4.$ $glob \in [Proc \rightarrow DeltaVecType]$
$\langle 2 \rangle 1.$ $glob = [p \in Proc \mapsto nrec]$ BY DEF $Init$
$\langle 2 \rangle 2.$ $CountVecType \subseteq DeltaVecType$
$\langle 3 \rangle 1.$ $Nat \subseteq Int$ BY $SMTT(10)$
$\langle 3 \rangle$ QED BY DEF $CountVecType, DeltaVecType$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 3, \langle 2 \rangle 1, \langle 2 \rangle 2$

$\langle 1 \rangle 5.$ $temp \in [Proc \rightarrow DeltaVecType]$
BY $DeltaVecZeroType, DeltaVecAddZero$ DEF $Init$

$\langle 1 \rangle 6.$ $msg \in [Proc \rightarrow [Proc \rightarrow Seq(DeltaVecType)]]$
$\langle 2 \rangle 1.$ $\langle \rangle \in Seq(DeltaVecType)$ BY $EmptySeq$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$ DEF $Init$

$\langle 1 \rangle 7.$ $nrecvut \in [Point \rightarrow \text{BOOLEAN}]$
BY DEF $Init, NrecVacantUpto, IsDeltaVecVacantUpto$

$\langle 1 \rangle 8.$ $globvut \in [Proc \rightarrow [Point \rightarrow \text{BOOLEAN}]]$
BY DEF $Init, GlobVacantUpto, IsDeltaVecVacantUpto$

$\langle 1 \rangle 9.$ $IsFiniteTempProcs$

$\langle 2 \rangle$ DEFINE $FP \triangleq \{p \in Proc : temp[p] \neq DeltaVecZero\}$
$\langle 2 \rangle 1.$ $\forall\, p \in Proc : temp[p] = DeltaVecZero$ BY DEF $Init$
$\langle 2 \rangle 2.$ $FP = \{\}$ BY $\langle 2 \rangle 1$
$\langle 2 \rangle 3.$ $IsFiniteSet(FP)$ BY $\langle 2 \rangle 2$, $FiniteSetEmpty$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 3$ DEF $IsFiniteTempProcs$

$\langle 1 \rangle 10.$ $IsFiniteMsgSenders$
$\langle 2 \rangle$ SUFFICES ASSUME NEW $q \in Proc$
PROVE $IsFiniteSet(\{p \in Proc : msg[p][q] \neq \langle\rangle\})$
BY DEF $IsFiniteMsgSenders$
$\langle 2 \rangle 1.$ $\forall\, p \in Proc : msg[p][q] = \langle\rangle$ BY DEF $Init$
$\langle 2 \rangle 2.$ $\{p \in Proc : msg[p][q] \neq \langle\rangle\} = \{\}$ BY $\langle 2 \rangle 1$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$, $FiniteSetEmpty$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1$, $\langle 1 \rangle 2$, $\langle 1 \rangle 3$, $\langle 1 \rangle 4$, $\langle 1 \rangle 5$, $\langle 1 \rangle 6$, $\langle 1 \rangle 7$, $\langle 1 \rangle 8$, $\langle 1 \rangle 9$, $\langle 1 \rangle 10$ DEF $InvType$

$InvType$ carries through a $Next$ step.

THEOREM $ThmNextInvType \triangleq$
$\land\ InvType$
$\land\ [Next]_{vars}$
$\Rightarrow$
$InvType'$
PROOF
$\langle 1 \rangle$ SUFFICES ASSUME $InvType$, $Next$PROVE $InvType'$
Dispose of the stutter step.
$\langle 2 \rangle$ CASE UNCHANGED $vars$ BY DEF $vars$, $InvType$, $IsFiniteTempProcs$, $IsFiniteMsgSenders$
$\langle 2 \rangle$ QED OBVIOUS

If the action is $NextPerformOperation$.
$\langle 1 \rangle 1.$ CASE $NextPerformOperation$
$\langle 2 \rangle 1.$ PICK $p \in Proc$, $c \in PointToNat$, $r \in PointToNat$:
$NextPerformOperation\_WithPCR(p, c, r)$
BY $\langle 1 \rangle 1$ DEF $NextPerformOperation$, $NextPerformOperation\_WithPCR$

$\langle 2 \rangle 2.$ $NextPerformOperation\_State\_Conclusion(p, c, r)$
BY $\langle 2 \rangle 1$, $NextPerformOperation\_State$

$\langle 2 \rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$

If the action is $NextSendUpdate$.

$\langle 1 \rangle$2. CASE $NextSendUpdate$
  $\langle 2 \rangle$1. PICK $p \in Proc,\ tt \in$ SUBSET $Point$:
    $NextSendUpdate\_WithPTT(p,\ tt)$
    BY $\langle 1 \rangle 2$ DEF $NextSendUpdate,\ NextSendUpdate\_WithPTT$

  $\langle 2 \rangle$2. $NextSendUpdate\_State\_Conclusion(p,\ tt)$
    BY $\langle 2 \rangle 1,\ NextSendUpdate\_State$

  $\langle 2 \rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$

  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$

If the action is $NextReceiveUpdate$.

$\langle 1 \rangle$3. CASE $NextReceiveUpdate$
  $\langle 2 \rangle$1. PICK $p \in Proc,\ q \in Proc$:
    $NextReceiveUpdate\_WithPQ(p,\ q)$
    BY $\langle 1 \rangle 3$ DEF $NextReceiveUpdate,\ NextReceiveUpdate\_WithPQ$

  $\langle 2 \rangle$2. $NextReceiveUpdate\_State\_Conclusion(p,\ q)$
    BY $\langle 2 \rangle 1,\ NextReceiveUpdate\_State$

  $\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$

  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 1,\ \langle 1 \rangle 2,\ \langle 1 \rangle 3$ DEF $Next$

$InvType$ holds in all reachable states.

THEOREM $ThmInvType \triangleq$
  $Spec \Rightarrow \Box InvType$
PROOF
  $\langle 1 \rangle$ $Init \Rightarrow InvType$ BY $ThmInitInvType$
  $\langle 1 \rangle$ $InvType \land [Next]_{vars} \Rightarrow InvType'$ BY $ThmNextInvType$
  $\langle 1 \rangle$ $Init \land \Box[Next]_{vars} \Rightarrow \Box InvType$ OMITTED $TLAPS$ cannot check it
  $\langle 1 \rangle$ QED OMITTED BY DEF $Spec$

C.21 Proof of invariant InvTempUpright

MODULE $NaiadClockProofInvTempUpright$

EXTENDS $NaiadClockProofInvType$

Proof of invariant InvTempUpright.

$InvTempUpright$ holds in the initial state.

THEOREM $ThmInitInvTempUpright \triangleq$
  $Init \Rightarrow InvTempUpright$
PROOF
  $\langle 1 \rangle$ SUFFICES ASSUME $Init$ PROVE $InvTempUpright$ OBVIOUS
  $\langle 1 \rangle$ QED BY $DeltaVecUpright\_Zero$ DEF $Init$, $InvTempUpright$

$InvTempUpright$ carries through a $Next$ step.

THEOREM $ThmNextInvTempUpright \triangleq$
  $\land InvType$
  $\land InvTempUpright$
  $\land [Next]_{vars}$
  $\Rightarrow$
  $InvTempUpright'$
PROOF
  $\langle 1 \rangle$ SUFFICES ASSUME
      $InvType$,
      $InvTempUpright$,
      $[Next]_{vars}$
    PROVE $InvTempUpright'$
    OBVIOUS

  $\langle 1 \rangle$ SUFFICES ASSUME $Next$ PROVE $InvTempUpright'$
    $\langle 2 \rangle$ CASE UNCHANGED $vars$ BY DEF $vars$, $InvTempUpright$

$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle$ SUFFICES ASSUME NEW $fp \in Proc$
PROVE $IsDeltaVecUpright(lleq, temp[fp])'$
BY DEF $InvTempUpright$

$\langle 1 \rangle 2.$ $lleq \in PointRelationType$ BY DEF $InvType$
$\langle 1 \rangle 3.$ $IsPartialOrder(lleq)$ BY DEF $InvType$

If the action is $NextPerformOperation$.

$\langle 1 \rangle 4.$ CASE $NextPerformOperation$
$\langle 2 \rangle 1.$ PICK $p \in Proc,\ c \in PointToNat,\ r \in PointToNat$:
$NextPerformOperation\_WithPCR(p, c, r)$
BY $\langle 1 \rangle 4$ DEF $NextPerformOperation,\ NextPerformOperation\_WithPCR$

$\langle 2 \rangle 2.$ $NextPerformOperation\_State\_Conclusion(p, c, r)$
BY $\langle 2 \rangle 1,\ NextPerformOperation\_State$

$\langle 2 \rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$

$\langle 2 \rangle 3.$ UNCHANGED $lleq$ BY $\langle 2 \rangle 2$

$temp[fp]$ unchanged.

$\langle 2 \rangle 4.$ CASE $fp \neq p$
$\langle 3 \rangle 1.$ UNCHANGED $temp[fp]$ BY $\langle 2 \rangle 2,\ \langle 2 \rangle 4$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 2 \rangle 3$ DEF $InvTempUpright$

$temp[fp]$ changed.

$\langle 2 \rangle 5.$ CASE $fp = p$

$\langle 3 \rangle$ DEFINE $tempp \triangleq temp[p]$
$\langle 3 \rangle$ DEFINE $delta \triangleq NextPerformOperation\_Delta(p, c, r)$

$\langle 3 \rangle$ SUFFICES $IsDeltaVecUpright(lleq, tempp')$ BY $\langle 2 \rangle 3,\ \langle 2 \rangle 5$
$\langle 3 \rangle 1.$ $tempp \in DeltaVecType$ BY DEF $InvType$
$\langle 3 \rangle 2.$ $IsDeltaVecUpright(lleq, tempp)$ BY DEF $InvTempUpright$
$\langle 3 \rangle 3.$ $delta \in DeltaVecType$ BY $\langle 2 \rangle 2$
$\langle 3 \rangle 4.$ $IsDeltaVecUpright(lleq, delta)$ BY $\langle 2 \rangle 2$
$\langle 3 \rangle 5.$ $tempp' = DeltaVecAdd(tempp, delta)$ BY $\langle 2 \rangle 2,\ \langle 2 \rangle 5$

$\langle 3 \rangle$ HIDE DEF $delta,\ tempp$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 3 \rangle 2,\ \langle 3 \rangle 3,\ \langle 3 \rangle 4,\ \langle 3 \rangle 5,\ \langle 1 \rangle 2,\ \langle 1 \rangle 3,\ DeltaVecUpright\_Add$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 4,\ \langle 2 \rangle 5$

If the action is $NextSendUpdate$.

$\langle 1 \rangle 5.$ CASE $NextSendUpdate$
$\langle 2 \rangle 1.$ PICK $p \in Proc,\ tt \in$ SUBSET $Point$:

$NextSendUpdate\_WithPTT(p, tt)$
BY $\langle 1 \rangle 5$ DEF $NextSendUpdate$, $NextSendUpdate\_WithPTT$

$\langle 2 \rangle 2$. $NextSendUpdate\_State\_Conclusion(p, tt)$
BY $\langle 2 \rangle 1$, $NextSendUpdate\_State$

$\langle 2 \rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$

$temp[fp]$ unchanged.

$\langle 2 \rangle 6$. CASE $fp \neq p$
$\langle 3 \rangle 1$. UNCHANGED $temp[fp]$ BY $\langle 2 \rangle 6$, $\langle 2 \rangle 2$
$\langle 3 \rangle 2$. UNCHANGED $lleq$ BY $\langle 2 \rangle 2$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$ DEF $InvTempUpright$

$temp[fp]$ changed.

$\langle 2 \rangle 7$. CASE $fp = p$
$\langle 3 \rangle 1$. $IsDeltaVecUpright(lleq, temp[p])$ BY DEF $InvTempUpright$
$\langle 3 \rangle 2$. $IsDeltaVecUpright(lleq, temp[p])'$ BY $\langle 3 \rangle 1$, $\langle 2 \rangle 2$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 2$, $\langle 2 \rangle 7$ DEF $InvTempUpright$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 6$, $\langle 2 \rangle 7$

If the action is $NextReceiveUpdate$.

$\langle 1 \rangle 6$. CASE $NextReceiveUpdate$
$\langle 2 \rangle 1$. PICK $p \in Proc$, $q \in Proc$:
$NextReceiveUpdate\_WithPQ(p, q)$
BY $\langle 1 \rangle 6$ DEF $NextReceiveUpdate$, $NextReceiveUpdate\_WithPQ$

$\langle 2 \rangle 2$. $NextReceiveUpdate\_State\_Conclusion(p, q)$
BY $\langle 2 \rangle 1$, $NextReceiveUpdate\_State$

$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$

$\langle 2 \rangle 3$. UNCHANGED $lleq$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 4$. UNCHANGED $temp$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 3$, $\langle 2 \rangle 4$ DEF $InvTempUpright$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 4$, $\langle 1 \rangle 5$, $\langle 1 \rangle 6$ DEF $Next$

$InvTempUpright$ holds in all reachable states.

THEOREM $ThmInvTempUpright \triangleq$
$Spec \Rightarrow \Box InvTempUpright$

PROOF

$\langle 1 \rangle$ DEFINE $I \triangleq$

$\land$ $InvType$

$\land$ $InvTempUpright$

$\langle 1 \rangle$ $Init \Rightarrow I$

$\langle 2 \rangle$ USE $ThmInitInvType$

$\langle 2 \rangle$ USE $ThmInitInvTempUpright$

$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle$ $I \land [Next]_{vars} \Rightarrow I'$

$\langle 2 \rangle$ USE $ThmNextInvType$

$\langle 2 \rangle$ USE $ThmNextInvTempUpright$

$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle$ $Init \land \Box[Next]_{vars} \Rightarrow \Box I$ OMITTED $TLAPS$ cannot check it

$\langle 1 \rangle$ $Spec \Rightarrow \Box I$ OMITTED BY DEF $Spec$

$\langle 1 \rangle$ QED OMITTED $TLAPS$ cannot check it

C.22 Proof of invariant InvIncomingInfoUpright

MODULE $NaiadClockProofInvIncomingInfoUpright$

EXTENDS $NaiadClockProofInvTempUpright$

Proof of invariant InvIncomingInfoUpright.

$InvIncomingInfoUpright$ holds in the initial state.

THEOREM $ThmInitInvIncomingInfoUpright \triangleq$
  $Init \Rightarrow InvIncomingInfoUpright$
PROOF
  $\langle 1 \rangle$ SUFFICES ASSUME $Init$ PROVE $InvIncomingInfoUpright$ OBVIOUS
  $\langle 1 \rangle$ $InvType$BY $ThmInitInvType$
  $\langle 1 \rangle$ $InvTempUpright$ BY $ThmInitInvTempUpright$

  $\langle 1 \rangle$ $\forall k \in Nat :$
    $\forall p \in Proc :$
    $\forall q \in Proc :$
    $IsDeltaVecUpright(lleq, IncomingInfo(k, p, q))$

    $\langle 2 \rangle$ SUFFICES ASSUME
        NEW $k \in Nat$,
        NEW $p \in Proc$,
        NEW $q \in Proc$
      PROVE $IsDeltaVecUpright(lleq, IncomingInfo(k, p, q))$
      OBVIOUS

    $\langle 2 \rangle$ $IncomingInfo(k, p, q) = DeltaVecZero$
      $\langle 3 \rangle$ $Init_IncomingInfo_Conclusion(k, p, q)$BY $Init\_IncomingInfo$
      $\langle 3 \rangle$ QED BY DEF $Init\_IncomingInfo\_Conclusion$
    $\langle 2 \rangle$ $lleq \in PointRelationType$ BY DEF $InvType$
    $\langle 2 \rangle$ $IsPartialOrder(lleq)$ BY DEF $InvType$
    $\langle 2 \rangle$ QED BY $DeltaVecUpright\_Zero$
  $\langle 1 \rangle$ QED BY DEF $InvIncomingInfoUpright$

InvIncomingInfoUpright carries through a *Next* step.

THEOREM $ThmNextInvIncomingInfoUpright \triangleq$
  $\land\ InvType$
  $\land\ InvTempUpright$
  $\land\ InvIncomingInfoUpright$
  $\land\ [Next]_{vars}$
  $\Rightarrow$
  $InvIncomingInfoUpright'$
PROOF
  $\langle 1 \rangle$ SUFFICES ASSUME
      $InvType,$
      $InvTempUpright,$
      $InvIncomingInfoUpright,$
      $[Next]_{vars}$
    PROVE $InvIncomingInfoUpright'$
    OBVIOUS

  $\langle 1 \rangle$ $InvIncomingInfoType$ BY $DeduceInvIncomingInfoType$
  $\langle 1 \rangle$ $InvType'$ BY $ThmNextInvType$
  $\langle 1 \rangle$ $InvTempUpright'$ BY $ThmNextInvTempUpright$
  $\langle 1 \rangle$ $InvIncomingInfoType'$ BY $DeduceInvIncomingInfoType$

Dispose of the stutter step.

  $\langle 1 \rangle 1.$ CASE UNCHANGED $vars$
    $\langle 2 \rangle$ USE DEF $vars$
    $\langle 2 \rangle$ USE DEF $InvIncomingInfoUpright$
    $\langle 2 \rangle$ USE DEF $IncomingInfo$
    $\langle 2 \rangle$ QED BY $\langle 1 \rangle 1$

Set up to prove $InvIncomingInfoUpright'$.

  $\langle 1 \rangle$ SUFFICES ASSUME $Next$ PROVE $InvIncomingInfoUpright'$ BY $\langle 1 \rangle 1$
  $\langle 1 \rangle$ SUFFICES ASSUME
      NEW $fk \in Nat,$
      NEW $fp \in Proc,$
      NEW $fq \in Proc$
    PROVE $IsDeltaVecUpright(lleq,\ IncomingInfo(fk,\ fp,\ fq))'$
    BY DEF $InvIncomingInfoUpright$

If the action is $NextPerformOperation$.

  $\langle 1 \rangle 2.$ CASE $NextPerformOperation$
    $\langle 2 \rangle 1.$ PICK $p \in Proc,\ c \in PointToNat,\ r \in PointToNat$:
        $NextPerformOperation\_WithPCR(p,\ c,\ r)$
        BY $\langle 1 \rangle 2$ DEF $NextPerformOperation,\ NextPerformOperation\_WithPCR$
    $\langle 2 \rangle 2.$ $NextPerformOperation\_State\_Conclusion(p,\ c,\ r)$
        BY $\langle 2 \rangle 1,\ NextPerformOperation\_State$

$\langle 2 \rangle 3.$ $NextPerformOperation\_IncomingInfo\_Conclusion(fk,\ fp,\ fq,\ p,\ c,\ r)$
BY $\langle 2 \rangle 1$, $NextPerformOperation\_IncomingInfo$
$\langle 2 \rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$
$\langle 2 \rangle$ USE DEF $NextPerformOperation\_IncomingInfo\_Conclusion$

$\langle 2 \rangle$ DEFINE $II \triangleq IncomingInfo(fk,\ fp,\ fq)$
$\langle 2 \rangle$ DEFINE $delta \triangleq NextPerformOperation\_Delta(p,\ c,\ r)$

$\langle 2 \rangle 4.$ UNCHANGED $lleq$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 5.$ $lleq' \in PointRelationType$ BY DEF $InvType$
$\langle 2 \rangle 6.$ $IsPartialOrder(lleq')$ BY DEF $InvType$
$\langle 2 \rangle 7.$ $II \in DeltaVecType$ BY DEF $InvIncomingInfoType$
$\langle 2 \rangle 8.$ $delta \in DeltaVecType$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 9.$ $IsDeltaVecUpright(lleq',\ delta)$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 10.$ $IsDeltaVecUpright(lleq',\ II)$ BY $\langle 2 \rangle 4$ DEF $InvIncomingInfoUpright$
$\langle 2 \rangle 11.$ $IsDeltaVecUpright(lleq',\ DeltaVecAdd(II,\ delta))$
$\langle 3 \rangle$ HIDE DEF $delta,\ II$
$\langle 3 \rangle$ QED BY $\langle 2 \rangle 5$, $\langle 2 \rangle 6$, $\langle 2 \rangle 7$, $\langle 2 \rangle 8$, $\langle 2 \rangle 9$, $\langle 2 \rangle 10$, $DeltaVecUpright\_Add$
$\langle 2 \rangle 12.$ $II' =$ IF $fp = p$ THEN $DeltaVecAdd(II,\ delta)$ ELSE II BY $\langle 2 \rangle 3$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 10$, $\langle 2 \rangle 11$, $\langle 2 \rangle 12$

If the action is $NextSendUpdate$.

$\langle 1 \rangle 3.$ CASE $NextSendUpdate$
$\langle 2 \rangle 1.$ PICK $p \in Proc,\ tt \in$ SUBSET $Point$:
$NextSendUpdate\_WithPTT(p,\ tt)$
BY $\langle 1 \rangle 3$ DEF $NextSendUpdate$, $NextSendUpdate\_WithPTT$
$\langle 2 \rangle 2.$ $NextSendUpdate\_State\_Conclusion(p,\ tt)$
BY $\langle 2 \rangle 1$, $NextSendUpdate\_State$
$\langle 2 \rangle 3.$ $NextSendUpdate\_IncomingInfo\_Conclusion(fk,\ fp,\ fq,\ p,\ tt)$
BY $\langle 2 \rangle 1$, $NextSendUpdate\_IncomingInfo$
$\langle 2 \rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$
$\langle 2 \rangle$ USE DEF $NextSendUpdate\_IncomingInfo\_Conclusion$

$\langle 2 \rangle$ DEFINE $II \triangleq IncomingInfo(fk,\ fp,\ fq)$
$\langle 2 \rangle$ DEFINE $msgpq \triangleq msg[fp][fq]$
$\langle 2 \rangle$ DEFINE $tempp \triangleq temp[fp]$

$\langle 2 \rangle 4.$ UNCHANGED $lleq$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 5.$ $II' =$ IF $fp = p \wedge fk > Len(msgpq)$ THEN $tempp'$ ELSE II BY $\langle 2 \rangle 3$
$\langle 2 \rangle 6.$ $IsDeltaVecUpright(lleq',\ tempp')$ BY DEF $InvTempUpright$
$\langle 2 \rangle 7.$ $IsDeltaVecUpright(lleq',\ II)$ BY $\langle 2 \rangle 4$ DEF $InvIncomingInfoUpright$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 5$, $\langle 2 \rangle 6$, $\langle 2 \rangle 7$

If the action is $NextReceiveUpdate$.

$\langle 1 \rangle 4.$ CASE $NextReceiveUpdate$
$\langle 2 \rangle 1.$ PICK $p \in Proc,\ q \in Proc$:
$NextReceiveUpdate\_WithPQ(p,\ q)$

BY $\langle 1\rangle 4$ DEF $NextReceiveUpdate$, $NextReceiveUpdate\_WithPQ$
$\langle 2\rangle 2.$ $NextReceiveUpdate\_State\_Conclusion(p, q)$
BY $\langle 2\rangle 1$, $NextReceiveUpdate\_State$
$\langle 2\rangle 3.$ $NextReceiveUpdate\_IncomingInfo\_Conclusion(fk, fp, fq, p, q)$
BY $\langle 2\rangle 1$, $NextReceiveUpdate\_IncomingInfo$
$\langle 2\rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$
$\langle 2\rangle$ USE DEF $NextReceiveUpdate\_IncomingInfo\_Conclusion$

$\langle 2\rangle$ DEFINE $II \triangleq IncomingInfo(fk, fp, fq)$
$\langle 2\rangle$ DEFINE $IIk1 \triangleq IncomingInfo(fk + 1, fp, fq)$

$\langle 2\rangle 4.$ UNCHANGED $lleq$ BY $\langle 2\rangle 2$
$\langle 2\rangle 5.$ $fk + 1 \in Nat$ BY $SMTT(10)$
$\langle 2\rangle 6.$ $IsDeltaVecUpright(lleq', II)$ BY $\langle 2\rangle 4$ DEF $InvIncomingInfoUpright$
$\langle 2\rangle 7.$ $IsDeltaVecUpright(lleq', IIk1)$ BY $\langle 2\rangle 4$, $\langle 2\rangle 5$ DEF $InvIncomingInfoUpright$
$\langle 2\rangle 8.$ $II' =$ IF $fp = p \land fq = q$ THEN $IIk1$ ELSE II BY $\langle 2\rangle 3$
$\langle 2\rangle$ QED BY $\langle 2\rangle 6$, $\langle 2\rangle 7$, $\langle 2\rangle 8$

$\langle 1\rangle$ QED BY $\langle 1\rangle 2$, $\langle 1\rangle 3$, $\langle 1\rangle 4$ DEF $Next$

$InvIncomingInfoUpright$ holds in all reachable states.

THEOREM $ThmInvIncomingInfoUpright \triangleq$
$Spec \Rightarrow \Box InvIncomingInfoUpright$
PROOF
$\langle 1\rangle$ DEFINE $I \triangleq$
$\land\ InvType$
$\land\ InvTempUpright$
$\land\ InvIncomingInfoUpright$

$\langle 1\rangle$ $Init \Rightarrow I$
$\langle 2\rangle$ USE $ThmInitInvType$
$\langle 2\rangle$ USE $ThmInitInvTempUpright$
$\langle 2\rangle$ USE $ThmInitInvIncomingInfoUpright$
$\langle 2\rangle$ QED OBVIOUS

$\langle 1\rangle$ $I \land [Next]_{vars} \Rightarrow I'$
$\langle 2\rangle$ USE $ThmNextInvType$
$\langle 2\rangle$ USE $ThmNextInvTempUpright$
$\langle 2\rangle$ USE $ThmNextInvIncomingInfoUpright$
$\langle 2\rangle$ QED OBVIOUS

$\langle 1\rangle$ $Init \land \Box[Next]_{vars} \Rightarrow \Box I$ OMITTED $TLAPS$ cannot check it

$\langle 1 \rangle$ $Spec \Rightarrow \Box I$ OMITTED BY DEF $Spec$
$\langle 1 \rangle$ QED OMITTED $TLAPS$ cannot check it

C.23 Proof of invariant InvInfoAtBetaUpright

MODULE $NaiadClockProofInvInfoAtBetaUpright$

EXTENDS $NaiadClockProofInvIncomingInfoUpright$

Proof of invariant InvInfoAtBetaUpright.

$InvInfoAtBetaUpright$ says that for all skip counts k, sending processors p, and receiving processors q, $InfoAt(k, p, q)$ is $IncomingInfo(k, p, q)$-upright. $InfoAt(k, p, q)$ is the information at position k on the message queue from p to q. $IncomingInfo(k, p, q)$ is the sum of all subsequent information from p to q.

To be $IncomingInfo(k, p, q)$-upright means that for each positive point in $InfoAt(k, p, q)$ there is a strictly lower point that either is negative in $InfoAt(k, p, q)$ or is negative in $IncomingInfo(k, p, q)$ and neither that point nor any yet lower point is positive in $InfoAt(k, p, q)$.

The invariant holds in the initial state because initially the message queues are empty, and hence no matter what position k is chosen we have $InfoAt(k, p, q) = 0$, and hence there are no positive points.

The invariant carries through each next step because:

(1) A $NextPerformOperation$ action adds delta to $temp[p]$. This has the effect of adding delta to the subsequent information $IncomingInfo(k, p, q)$ of each item $InfoAt(k, p, q)$ on any given message queue sent from p. However, since both $IncomingInfo(k, p, q)$ and delta must be upright, this preserves the beta-upright property of $InfoAt(k, p, q)$.

(2) A $NextSendUpdate$ action takes gamma from $temp[p]$ and appends it onto the message queue from p to q for all q. For all previously existing items $InfoAt(k, p, q)$ on the message queue, $IncomingInfo(k, p, q)$ is unchanged and so the beta-upright properties are unchanged. The only question is the new item just appended onto the message queue. In other words, we require that gamma is $temp[p]$'-upright. But this follows from the fact that gamma must positive imply $temp[p]$.

(3) A $NextReceiveUpdate$ action removes the head item from a message queue incoming at q. The positions of all items on the queue shift up, but all of their existing beta-upright properties are unchanged.

$InvInfoAtBetaUptight$ holds in the initial state.

THEOREM $ThmInitInvInfoAtBetaUpright \triangleq$
  $Init \Rightarrow InvInfoAtBetaUpright$
PROOF
  $\langle 1 \rangle$ SUFFICES ASSUME $Init$ PROVE $InvInfoAtBetaUpright$ OBVIOUS
  $\langle 1 \rangle$ $InvType$ BY $ThmInitInvType$
  $\langle 1 \rangle$ $InvIncomingInfoType$ BY $DeduceInvIncomingInfoType$

  $\langle 1 \rangle$ ASSUME
      NEW $k \in Nat$,
      NEW $p \in Proc$,
      NEW $q \in Proc$
    PROVE $IsDeltaVecBetaUpright(lleq, InfoAt(k, p, q), IncomingInfo(k, p, q))$

⟨2⟩ $InfoAt(k, p, q) = DeltaVecZero$
⟨3⟩ $Init\_InfoAt\_Conclusion(k, p, q)$ BY $Init\_InfoAt$
⟨3⟩ QED BY DEF $Init\_InfoAt\_Conclusion$

⟨2⟩ $IncomingInfo(k, p, q) \in DeltaVecType$ BY DEF $InvIncomingInfoType$

⟨2⟩ $lleq \in PointRelationType$ BY DEF $InvType$
⟨2⟩ $IsPartialOrder(lleq)$ BY DEF $InvType$
⟨2⟩ QED BY $DeltaVecBetaUpright\_Zero$
⟨1⟩ QED BY DEF $InvInfoAtBetaUpright$

$InvInfoAtBetaUpright$ carries through a $Next$ step.

THEOREM $ThmNextInvInfoAtBetaUpright \triangleq$
$\land$ $InvType$
$\land$ $InvTempUpright$
$\land$ $InvIncomingInfoUpright$
$\land$ $InvInfoAtBetaUpright$
$\land$ $[Next]_{vars}$
$\Rightarrow$
$InvInfoAtBetaUpright'$
PROOF
⟨1⟩ SUFFICES ASSUME
$InvType$,
$InvTempUpright$,
$InvIncomingInfoUpright$,
$InvInfoAtBetaUpright$,
$[Next]_{vars}$
PROVE $InvInfoAtBetaUpright'$
OBVIOUS

⟨1⟩ $InvInfoAtType$ BY $DeduceInvInfoAtType$
⟨1⟩ $InvIncomingInfoType$ BY $DeduceInvIncomingInfoType$
⟨1⟩ $InvType'$ BY $ThmNextInvType$
⟨1⟩ $InvTempUpright'$ BY $ThmNextInvTempUpright$
⟨1⟩ $InvIncomingInfoUpright'$ BY $ThmNextInvIncomingInfoUpright$
⟨1⟩ $InvInfoAtType'$ BY $DeduceInvInfoAtType$
⟨1⟩ $InvIncomingInfoType'$ BY $DeduceInvIncomingInfoType$

Dispose of the stutter step.

⟨1⟩1. CASE UNCHANGED $vars$
⟨2⟩ USE DEF $vars$

$\langle 2 \rangle$ USE DEF $InvInfoAtBetaUpright$
$\langle 2 \rangle$ USE DEF $InfoAt$
$\langle 2 \rangle$ USE DEF $IncomingInfo$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 1$

Set up to prove $InvInfoAtBetaUpright'$.

$\langle 1 \rangle$ SUFFICES ASSUME $Next$ PROVE $InvInfoAtBetaUpright'$ BY $\langle 1 \rangle 1$
$\langle 1 \rangle$ SUFFICES ASSUME
NEW $fk \in Nat$,
NEW $fp \in Proc$,
NEW $fq \in Proc$
PROVE $IsDeltaVecBetaUpright(lleq,\ InfoAt(fk,\ fp,\ fq),\ IncomingInfo(fk,\ fp,\ fq))'$
BY DEF $InvInfoAtBetaUpright$

$\langle 1 \rangle$ DEFINE $IA \triangleq InfoAt(fk,\ fp,\ fq)$
$\langle 1 \rangle$ DEFINE $II \triangleq IncomingInfo(fk,\ fp,\ fq)$

$\langle 1 \rangle 2.$ $II \in DeltaVecType$ BY DEF $InvIncomingInfoType$
$\langle 1 \rangle 3.$ $IA \in DeltaVecType$ BY DEF $InvInfoAtType$
$\langle 1 \rangle 4.$ $lleq \in PointRelationType$ BY DEF $InvType$
$\langle 1 \rangle 5.$ $IsPartialOrder(lleq)$ BY DEF $InvType$

If the action is $NextPerformOperation$.

$\langle 1 \rangle 6.$ CASE $NextPerformOperation$
$\langle 2 \rangle 1.$ PICK $p \in Proc,\ c \in PointToNat,\ r \in PointToNat$:
$NextPerformOperation\_WithPCR(p,\ c,\ r)$
BY $\langle 1 \rangle 6$ DEF $NextPerformOperation,\ NextPerformOperation\_WithPCR$

$\langle 2 \rangle 2.$ $NextPerformOperation\_State\_Conclusion(p,\ c,\ r)$
BY $\langle 2 \rangle 1,\ NextPerformOperation\_State$
$\langle 2 \rangle 3.$ $NextPerformOperation\_InfoAt\_Conclusion(fk,\ fp,\ fq,\ p,\ c,\ r)$
BY $\langle 2 \rangle 1,\ NextPerformOperation\_InfoAt$
$\langle 2 \rangle 4.$ $NextPerformOperation\_IncomingInfo\_Conclusion(fk,\ fp,\ fq,\ p,\ c,\ r)$
BY $\langle 2 \rangle 1,\ NextPerformOperation\_IncomingInfo$

$\langle 2 \rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$
$\langle 2 \rangle$ USE DEF $NextPerformOperation\_InfoAt\_Conclusion$
$\langle 2 \rangle$ USE DEF $NextPerformOperation\_IncomingInfo\_Conclusion$

$\langle 2 \rangle$ DEFINE $delta \triangleq NextPerformOperation\_Delta(p,\ c,\ r)$

$\langle 2 \rangle 5.$ $delta \in DeltaVecType$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 6.$ UNCHANGED $lleq$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 7.$ UNCHANGED IA BY $\langle 2 \rangle 3$

$\langle 2 \rangle 8.$ CASE $fp = p$
$\langle 3 \rangle 1.$ $II' = DeltaVecAdd(II,\ delta)$ BY $\langle 2 \rangle 4,\ \langle 2 \rangle 8$
$\langle 3 \rangle 2.$ $IsDeltaVecUpright(lleq,\ II)$ BY DEF $InvIncomingInfoUpright$

$\langle 3\rangle 3.$ $IsDeltaVecBetaUpright(lleq,\ IA,\ II)$ BY DEF $InvInfoAtBetaUpright$
$\langle 3\rangle 4.$ $IsDeltaVecUpright(lleq,\ delta)$ BY $\langle 2\rangle 2$
$\langle 3\rangle$ USE $\langle 3\rangle 1,\ \langle 3\rangle 2,\ \langle 3\rangle 3,\ \langle 3\rangle 4$
$\langle 3\rangle$ USE $\langle 2\rangle 5,\ \langle 2\rangle 6,\ \langle 2\rangle 7,\ \langle 1\rangle 2,\ \langle 1\rangle 3,\ \langle 1\rangle 4,\ \langle 1\rangle 5$
$\langle 3\rangle$ QED BY $DeltaVecBetaUpright\_Add$

$\langle 2\rangle 9.$ CASE $fp \neq p$
$\langle 3\rangle 1.$ UNCHANGED II BY $\langle 2\rangle 4,\ \langle 2\rangle 9$
$\langle 3\rangle$ USE DEF $InvInfoAtBetaUpright$
$\langle 3\rangle$ USE DEF $InfoAt$
$\langle 3\rangle$ USE DEF $IncomingInfo$
$\langle 3\rangle$ QED BY $\langle 3\rangle 1,\ \langle 2\rangle 6,\ \langle 2\rangle 7$

$\langle 2\rangle$ QED BY $\langle 2\rangle 8,\ \langle 2\rangle 9$

If the action is $NextSendUpdate$.

$\langle 1\rangle 7.$ CASE $NextSendUpdate$
$\langle 2\rangle 1.$ PICK $p \in Proc,\ tt \in$ SUBSET $Point$:
$NextSendUpdate\_WithPTT(p,\ tt)$
BY $\langle 1\rangle 7$ DEF $NextSendUpdate,\ NextSendUpdate\_WithPTT$

$\langle 2\rangle 2.$ $NextSendUpdate\_State\_Conclusion(p,\ tt)$
BY $\langle 2\rangle 1,\ NextSendUpdate\_State$
$\langle 2\rangle 3.$ $NextSendUpdate\_InfoAt\_Conclusion(fk,\ fp,\ fq,\ p,\ tt)$
BY $\langle 2\rangle 1,\ NextSendUpdate\_InfoAt$
$\langle 2\rangle 4.$ $NextSendUpdate\_IncomingInfo\_Conclusion(fk,\ fp,\ fq,\ p,\ tt)$
BY $\langle 2\rangle 1,\ NextSendUpdate\_IncomingInfo$

$\langle 2\rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$
$\langle 2\rangle$ USE DEF $NextSendUpdate\_InfoAt\_Conclusion$
$\langle 2\rangle$ USE DEF $NextSendUpdate\_IncomingInfo\_Conclusion$

$\langle 2\rangle 5.$ UNCHANGED $lleq$ BY $\langle 2\rangle 2$

$\langle 2\rangle 8.$ CASE $fp = p$
$\langle 3\rangle$ DEFINE $M \triangleq msg[fp][fq]$
$\langle 3\rangle$ DEFINE $LenM \triangleq Len(M)$
$\langle 3\rangle$ HIDE DEF $M,\ LenM$
$\langle 3\rangle 1.$ $M \in Seq(DeltaVecType)$ BY $ZenonT(20)$ DEF $InvType,\ M$
$\langle 3\rangle 2.$ $LenM \in Nat$ BY $\langle 3\rangle 1,\ LenInNat$ DEF $LenM$

$\langle 3\rangle 3.$ CASE $fk = LenM + 1$
$\langle 4\rangle 1.$ $IsDeltaVecBetaUpright(lleq,\ IA',\ temp'[p])$
$\langle 5\rangle 1.$ $IA' \in DeltaVecType$ BY DEF $InvInfoAtType$
$\langle 5\rangle 2.$ $IA' = NextSendUpdate\_Gamma(p,\ tt)$ BY $\langle 3\rangle 3,\ \langle 2\rangle 3,\ \langle 2\rangle 8$ DEF $LenM,\ M$
$\langle 5\rangle 3.$ $IsDeltaVecPositiveImplies(IA',\ temp[p])$ BY $\langle 5\rangle 2,\ \langle 2\rangle 2$
$\langle 5\rangle 4.$ $temp[p] = DeltaVecAdd(IA',\ temp'[p])$ BY $\langle 5\rangle 2,\ \langle 2\rangle 2$

$\langle 5\rangle 5.$ $IsDeltaVecUpright(lleq,\ temp[p])$ BY DEF $InvTempUpright$
$\langle 5\rangle 6.$ $temp'[p] \in DeltaVecType$ BY DEF $InvType$
$\langle 5\rangle$ USE $\langle 5\rangle 1,\ \langle 5\rangle 3,\ \langle 5\rangle 4,\ \langle 5\rangle 5,\ \langle 5\rangle 6,\ \langle 1\rangle 4,\ \langle 1\rangle 5$
$\langle 5\rangle$ QED BY $DeltaVecBetaUpright\_PositiveImplies$
$\langle 4\rangle 2.$ $II' = temp'[p]$
$\langle 5\rangle 1.$ $fk > LenM$ BY $\langle 3\rangle 3,\ \langle 3\rangle 2,\ SMTT(10)$
$\langle 5\rangle$ QED BY $\langle 5\rangle 1,\ \langle 2\rangle 4,\ \langle 2\rangle 8$ DEF $LenM,\ M$
$\langle 4\rangle 3.$ $IsDeltaVecBetaUpright(lleq',\ IA',\ II')$ BY $\langle 4\rangle 1,\ \langle 4\rangle 2,\ \langle 2\rangle 5$
$\langle 4\rangle$ QED BY $\langle 4\rangle 3$

$\langle 3\rangle 4.$ CASE $fk < LenM + 1$
$\langle 4\rangle 1.$ UNCHANGED IA
$\langle 5\rangle$ $fk \neq LenM + 1$ BY $\langle 3\rangle 2,\ \langle 3\rangle 4,\ SMTT(10)$
$\langle 5\rangle$ QED BY $\langle 2\rangle 3,\ \langle 2\rangle 8$ DEF $LenM,\ M$
$\langle 4\rangle 2.$ UNCHANGED II
$\langle 5\rangle$ $\neg(fk > LenM)$ BY $\langle 3\rangle 2,\ \langle 3\rangle 4,\ SMTT(10)$
$\langle 5\rangle$ QED BY $\langle 2\rangle 4,\ \langle 2\rangle 8$ DEF $LenM,\ M$
$\langle 4\rangle 3.$ $IsDeltaVecBetaUpright(lleq',\ IA',\ II')$
$\langle 5\rangle$ USE DEF $InvInfoAtBetaUpright$
$\langle 5\rangle$ USE DEF $InfoAt$
$\langle 5\rangle$ USE DEF $IncomingInfo$
$\langle 5\rangle$ QED BY $\langle 4\rangle 1,\ \langle 4\rangle 2,\ \langle 2\rangle 5$
$\langle 4\rangle$ QED BY $\langle 4\rangle 3$

$\langle 3\rangle 5.$ CASE $fk > LenM + 1$
$\langle 4\rangle 1.$ $IA = DeltaVecZero$
$\langle 5\rangle$ $fk\ > LenM$ BY $\langle 3\rangle 2,\ \langle 3\rangle 5,\ SMTT(10)$
$\langle 5\rangle$ USE DEF $InvInfoAtType,\ LenM,\ M$
$\langle 5\rangle$ QED BY $DeduceInvInfoAtType$
$\langle 4\rangle 2.$ UNCHANGED IA
$\langle 5\rangle$ $fk \neq LenM + 1$ BY $\langle 3\rangle 2,\ \langle 3\rangle 5,\ SMTT(10)$
$\langle 5\rangle$ QED BY $\langle 2\rangle 3,\ \langle 2\rangle 8$ DEF $LenM,\ M$
$\langle 4\rangle 3.$ $IA' = DeltaVecZero$ BY $\langle 4\rangle 1,\ \langle 4\rangle 2$
$\langle 4\rangle 4.$ $II'\ \in DeltaVecType$ BY DEF $InvIncomingInfoType$
$\langle 4\rangle 5.$ $IsDeltaVecBetaUpright(lleq',\ IA',\ II')$
$\langle 5\rangle$ USE $\langle 4\rangle 3,\ \langle 4\rangle 4,\ \langle 2\rangle 5,\ \langle 1\rangle 4,\ \langle 1\rangle 5$
$\langle 5\rangle$ QED BY $DeltaVecBetaUpright\_Zero$
$\langle 4\rangle$ QED BY $\langle 4\rangle 5$

$\langle 3\rangle$ QED BY $\langle 3\rangle 2,\ \langle 3\rangle 3,\ \langle 3\rangle 4,\ \langle 3\rangle 5,\ SMTT(10)$

$\langle 2\rangle 9.$ CASE $fp \neq p$
$\langle 3\rangle 1.$ UNCHANGED II BY $\langle 2\rangle 4,\ \langle 2\rangle 9$
$\langle 3\rangle 2.$ UNCHANGED IA BY $\langle 2\rangle 3,\ \langle 2\rangle 9$
$\langle 3\rangle$ USE DEF $InvInfoAtBetaUpright$
$\langle 3\rangle$ USE DEF $InfoAt$
$\langle 3\rangle$ USE DEF $IncomingInfo$

$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 2 \rangle 5$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 8$, $\langle 2 \rangle 9$

If the action is $NextReceiveUpdate$.

$\langle 1 \rangle 8.$ CASE $NextReceiveUpdate$
$\langle 2 \rangle 1.$ PICK $p \in Proc,\ q \in Proc$:
$NextReceiveUpdate\_WithPQ(p,\ q)$
BY $\langle 1 \rangle 8$ DEF $NextReceiveUpdate,\ NextReceiveUpdate\_WithPQ$

$\langle 2 \rangle 2.$ $NextReceiveUpdate\_State\_Conclusion(p,\ q)$
BY $\langle 2 \rangle 1,\ NextReceiveUpdate\_State$
$\langle 2 \rangle 3.$ $NextReceiveUpdate\_InfoAt\_Conclusion(fk,\ fp,\ fq,\ p,\ q)$
BY $\langle 2 \rangle 1,\ NextReceiveUpdate\_InfoAt$
$\langle 2 \rangle 4.$ $NextReceiveUpdate\_IncomingInfo\_Conclusion(fk,\ fp,\ fq,\ p,\ q)$
BY $\langle 2 \rangle 1,\ NextReceiveUpdate\_IncomingInfo$

$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$
$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_InfoAt\_Conclusion$
$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_IncomingInfo\_Conclusion$

$\langle 2 \rangle 5.$ UNCHANGED $lleq$ BY $\langle 2 \rangle 2$

$\langle 2 \rangle 8.$ CASE $fp = p \land fq = q$
$\langle 3 \rangle 1.$ CASE $fk = 0$
$\langle 4 \rangle 1.$ $IA' = DeltaVecZero$
$\langle 5 \rangle$ DEFINE $M \triangleq msg[fp][fq]$
$\langle 5 \rangle$ DEFINE $LenM \triangleq Len(M)$
$\langle 5 \rangle$ HIDE DEF $M,\ LenM$
$\langle 5 \rangle 1.$ $M' \in Seq(DeltaVecType)$ BY DEF $InvType,\ M$
$\langle 5 \rangle 2.$ $LenM' \in Nat$ BY $\langle 5 \rangle 1,\ LenInNat$ DEF $LenM$
$\langle 5 \rangle 3.$ $\neg(0 < fk \land fk \leq LenM')$ BY $\langle 3 \rangle 1,\ \langle 5 \rangle 2,\ SMTT(10)$
$\langle 5 \rangle$ QED BY $\langle 5 \rangle 3$ DEF $InfoAt,\ LenM,\ M$
$\langle 4 \rangle 2.$ $II' \in DeltaVecType$ BY DEF $InvIncomingInfoType$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1,\ \langle 4 \rangle 2,\ \langle 2 \rangle 5,\ \langle 1 \rangle 4,\ \langle 1 \rangle 5,\ DeltaVecBetaUpright\_Zero$

$\langle 3 \rangle 2.$ CASE $fk > 0$
$\langle 4 \rangle$ DEFINE $IIk1 \triangleq IncomingInfo(fk + 1,\ fp,\ fq)$
$\langle 4 \rangle$ DEFINE $IAk1 \triangleq InfoAt(fk + 1,\ fp,\ fq)$
$\langle 4 \rangle 1.$ $II' = IIk1$ BY $\langle 2 \rangle 4,\ \langle 2 \rangle 8$
$\langle 4 \rangle 2.$ $IA' = IAk1$ BY $\langle 3 \rangle 2,\ \langle 2 \rangle 3,\ \langle 2 \rangle 8$
$\langle 4 \rangle 3.$ $IsDeltaVecBetaUpright(lleq,\ IAk1,\ IIk1)$
$\langle 5 \rangle$ $fk + 1 \in Nat$ BY $SMTT(10)$
$\langle 5 \rangle$ QED BY DEF $InvInfoAtBetaUpright$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1,\ \langle 4 \rangle 2,\ \langle 4 \rangle 3,\ \langle 2 \rangle 5$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 3 \rangle 2,\ SMTT(10)$

$\langle 2 \rangle 9.$ CASE $\neg(fp = p \land fq = q)$
$\langle 3 \rangle 1.$ UNCHANGED II BY $\langle 2 \rangle 4,\ \langle 2 \rangle 9$
$\langle 3 \rangle 2.$ UNCHANGED IA BY $\langle 2 \rangle 3,\ \langle 2 \rangle 9$
$\langle 3 \rangle$ USE DEF $InvInfoAtBetaUpright$
$\langle 3 \rangle$ USE DEF $InfoAt$
$\langle 3 \rangle$ USE DEF $IncomingInfo$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 3 \rangle 2,\ \langle 2 \rangle 5$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 8,\ \langle 2 \rangle 9$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 6,\ \langle 1 \rangle 7,\ \langle 1 \rangle 8$ DEF $Next$

$InvInfoAtBetaUpright$ holds in all reachable states.

THEOREM $ThmInvInvInfoAtBetaUpright \triangleq$
$Spec \Rightarrow \Box InvInfoAtBetaUpright$
PROOF
$\langle 1 \rangle$ DEFINE $I \triangleq$
$\land\ InvType$
$\land\ InvTempUpright$
$\land\ InvIncomingInfoUpright$
$\land\ InvInfoAtBetaUpright$

$\langle 1 \rangle$ $Init \Rightarrow I$
$\langle 2 \rangle$ USE $ThmInitInvType$
$\langle 2 \rangle$ USE $ThmInitInvTempUpright$
$\langle 2 \rangle$ USE $ThmInitInvIncomingInfoUpright$
$\langle 2 \rangle$ USE $ThmInitInvInfoAtBetaUpright$
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle$ $I \land [Next]_{vars} \Rightarrow I'$
$\langle 2 \rangle$ USE $ThmNextInvType$
$\langle 2 \rangle$ USE $ThmNextInvTempUpright$
$\langle 2 \rangle$ USE $ThmNextInvIncomingInfoUpright$
$\langle 2 \rangle$ USE $ThmNextInvInfoAtBetaUpright$
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle$ $Init \land \Box[Next]_{vars} \Rightarrow \Box I$ OMITTED $TLAPS$ cannot check it
$\langle 1 \rangle$ $Spec \Rightarrow \Box I$ OMITTED BY DEF $Spec$
$\langle 1 \rangle$ QED OMITTED $TLAPS$ cannot check it

C.24 Proof of invariant InvGlobalRecordCount

MODULE $NaiadClockProofInvGlobalRecordCount$

EXTENDS $NaiadClockProofInvInfoAtBetaUpright$

Proof of invariant InvGlobalRecordCount.

$InvGlobalRecordCount$ says that for all processors q, the sum of all information incoming at q, plus $glob[q]$, equals $nrec$. This invariant holds in the initial state and carries through each next step.

The invariant holds in the initial state because initially $glob[q] = nrec$, $temp[p] = 0$ for all processors p, and all message queues are empty, meaning that there is no information incoming at q.

The invariant carries through each next step because:

(1) A $NextPerformOperation$ action adds delta to both $nrec$ and $temp[p]$. Since $temp[p]$ is included in the sum of all information incoming at q this preserves the invariant.

(2) A $NextSendUpdate$ action removes gamma from $temp[p]$ and appends it onto all of the message queues from p. Since this has no net effect on the sum of information incoming at q, the invariant is preserved.

(3) A $NextReceiveUpdate$ action removes kappa from a message queue incoming at q and adds it to $glob[q]$. Since there is no change to $nrec$, this also preserves the invariant.

$InvGlobalRecordCount$ holds in the initial state.

THEOREM $ThmInitInvGlobalRecordCount \triangleq$
  $Init \Rightarrow InvGlobalRecordCount$
PROOF
  $\langle 1 \rangle$ SUFFICES ASSUME $Init$ PROVE $InvGlobalRecordCount$ OBVIOUS
  $\langle 1 \rangle$ $InvType$ BY $ThmInitInvType$
  $\langle 1 \rangle$ $InvGlobalRecordCount$
    $\langle 2 \rangle$ SUFFICES ASSUME NEW $q \in Proc$
      PROVE $nrec = DeltaVecAdd(GlobalIncomingInfo(0, q, q), glob[q])$
      BY DEF $InvGlobalRecordCount$
    $\langle 2 \rangle$ DEFINE $GII \triangleq GlobalIncomingInfo(0, q, q)$
    $\langle 2 \rangle 1.$ $nrec = glob[q]$ BY DEF $Init$
    $\langle 2 \rangle 2.$ $glob[q] \in DeltaVecType$ BY DEF $InvType$
    $\langle 2 \rangle 3.$ $GII = DeltaVecZero$
      $\langle 3 \rangle$ $Init\_GlobalIncomingInfo\_Conclusion(0, q, q)$ BY $Init\_GlobalIncomingInfo$
      $\langle 3 \rangle$ QED BY DEF $Init\_GlobalIncomingInfo\_Conclusion$
    $\langle 2 \rangle 4.$ $glob[q] = DeltaVecAdd(GII, glob[q])$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 3$, $DeltaVecAddZero$
    $\langle 2 \rangle$ QED BY $\langle 2 \rangle 1$, $\langle 2 \rangle 4$

$\langle 1 \rangle$ QED OBVIOUS

$InvGlobalRecordCount$ carries through a $Next$ step.

THEOREM $ThmNextInvGlobalRecordCount \triangleq$
$\wedge\ InvType$
$\wedge\ InvTempUpright$
$\wedge\ InvIncomingInfoUpright$
$\wedge\ InvGlobalRecordCount$
$\wedge\ [Next]_{vars}$
$\Rightarrow$
$InvGlobalRecordCount'$
PROOF
$\langle 1 \rangle$ SUFFICES ASSUME
$InvType$,
$InvTempUpright$,
$InvIncomingInfoUpright$,
$InvGlobalRecordCount$,
$[Next]_{vars}$
PROVE $InvGlobalRecordCount'$
OBVIOUS

$\langle 1 \rangle$ $InvGlobalIncomingInfoType$ BY $DeduceInvGlobalIncomingInfoType$
$\langle 1 \rangle$ $InvGlobalIncomingInfoUpright$ BY $DeduceInvGlobalIncomingInfoUpright$
$\langle 1 \rangle$ $InvType'$ BY $ThmNextInvType$
$\langle 1 \rangle$ $InvTempUpright'$ BY $ThmNextInvTempUpright$
$\langle 1 \rangle$ $InvIncomingInfoUpright'$ BY $ThmNextInvIncomingInfoUpright$
$\langle 1 \rangle$ $InvGlobalIncomingInfoType'$ BY $DeduceInvGlobalIncomingInfoType$
$\langle 1 \rangle$ $InvGlobalIncomingInfoUpright'$ BY $DeduceInvGlobalIncomingInfoUpright$

Dispose of the stutter step.

$\langle 1 \rangle 1.$ CASE UNCHANGED $vars$
$\langle 2 \rangle$ USE DEF $vars$
$\langle 2 \rangle$ USE DEF $InvGlobalRecordCount$
$\langle 2 \rangle$ USE DEF $GlobalIncomingInfo$
$\langle 2 \rangle$ USE DEF $IncomingInfo$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 1$

Set up to prove $InvGlobalRecordCount'$.

$\langle 1 \rangle$ SUFFICES ASSUME $Next$ PROVE $InvGlobalRecordCount'$ BY $\langle 1 \rangle 1$
$\langle 1 \rangle$ SUFFICES ASSUME NEW $fq \in Proc$
PROVE $nrec' = DeltaVecAdd(GlobalIncomingInfo(0, fq, fq)', glob[fq]')$
BY DEF $InvGlobalRecordCount$

$\langle 1 \rangle$ DEFINE $GII \triangleq GlobalIncomingInfo(0, fq, fq)$
$\langle 1 \rangle$ DEFINE $globfq \triangleq glob[fq]$

$\langle 1 \rangle$ SUFFICES $nrec' = DeltaVecAdd(GII', globfq')$ OBVIOUS

$\langle 1 \rangle 2.$ $GII \in DeltaVecType$ BY DEF $InvGlobalIncomingInfoType$
$\langle 1 \rangle 3.$ $GII' \in DeltaVecType$ BY DEF $InvGlobalIncomingInfoType$
$\langle 1 \rangle 4.$ $globfq \in DeltaVecType$ BY DEF $InvType$
$\langle 1 \rangle 5.$ $globfq' \in DeltaVecType$ BY DEF $InvType$
$\langle 1 \rangle 6.$ $nrec = DeltaVecAdd(GII, globfq)$ BY DEF $InvGlobalRecordCount$

If the action is $NextPerformOperation$.

Adds delta to both $nrec$ and GII, and so preserves the invariant.

$\langle 1 \rangle 7.$ CASE $NextPerformOperation$
  $\langle 2 \rangle 1.$ PICK $p \in Proc, c \in PointToNat, r \in PointToNat$:
    $NextPerformOperation\_WithPCR(p, c, r)$
    BY $\langle 1 \rangle 7$ DEF $NextPerformOperation, NextPerformOperation\_WithPCR$

  $\langle 2 \rangle 2.$ $NextPerformOperation\_State\_Conclusion(p, c, r)$
    BY $\langle 2 \rangle 1$, $NextPerformOperation\_State$
  $\langle 2 \rangle 3.$ $NextPerformOperation\_GlobalIncomingInfo\_Conclusion(fq, p, c, r)$
    BY $\langle 2 \rangle 1$, $NextPerformOperation\_GlobalIncomingInfo$

  $\langle 2 \rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$
  $\langle 2 \rangle$ USE DEF $NextPerformOperation\_GlobalIncomingInfo\_Conclusion$

  $\langle 2 \rangle$ DEFINE $delta \triangleq NextPerformOperation\_Delta(p, c, r)$

  $\langle 2 \rangle 4.$ $delta \in DeltaVecType$ BY $\langle 2 \rangle 2$
  $\langle 2 \rangle 5.$ $nrec' = DeltaVecAdd(nrec, delta)$ BY $\langle 2 \rangle 2$
  $\langle 2 \rangle 6.$ UNCHANGED $globfq$ BY $\langle 2 \rangle 2$
  $\langle 2 \rangle 7.$ $GII' = DeltaVecAdd(GII, delta)$ BY $\langle 2 \rangle 3$

  $\langle 2 \rangle$ QED   by commutative and associative properties of $DeltaVecAdd$
    $\langle 3 \rangle$ HIDE DEF $GII, globfq, delta$ hide the complicated definitions
    $\langle 3 \rangle$ USE $\langle 2 \rangle 4, \langle 1 \rangle 2, \langle 1 \rangle 4$ know that they are delta vectors
    $\langle 3 \rangle$ USE $DeltaVecAddCommutative$ know that add is commutative
    $\langle 3 \rangle$ USE $DeltaVecAddAssociative$ know that add is associative

    $\langle 3 \rangle 1.$ $nrec' = DeltaVecAdd(DeltaVecAdd(GII, globfq), delta)$ BY $\langle 2 \rangle 5, \langle 1 \rangle 6$
    $\langle 3 \rangle 2.$ $nrec' = DeltaVecAdd(GII, DeltaVecAdd(globfq, delta))$ BY $\langle 3 \rangle 1$
    $\langle 3 \rangle 3.$ $nrec' = DeltaVecAdd(GII, DeltaVecAdd(delta, globfq))$ BY $\langle 3 \rangle 2$
    $\langle 3 \rangle 4.$ $nrec' = DeltaVecAdd(DeltaVecAdd(GII, delta), globfq)$ BY $\langle 3 \rangle 3$
    $\langle 3 \rangle 5.$ $nrec' = DeltaVecAdd(GII', globfq)$ BY $\langle 3 \rangle 4, \langle 2 \rangle 7$
    $\langle 3 \rangle 6.$ $nrec' = DeltaVecAdd(GII', globfq')$ BY $\langle 3 \rangle 5, \langle 2 \rangle 6$
    $\langle 3 \rangle$ QED BY $\langle 3 \rangle 6$

If the action is $NextSendUpdate$.

No change to $nrec$, $globfq$, or GII and so preserves the invariant.

$\langle 1 \rangle 8.$ CASE $NextSendUpdate$
  $\langle 2 \rangle 1.$ PICK $p \in Proc,\ tt \in$ SUBSET $Point$:
      $NextSendUpdate\_WithPTT(p,\ tt)$
      BY $\langle 1 \rangle 8$ DEF $NextSendUpdate,\ NextSendUpdate\_WithPTT$

  $\langle 2 \rangle 2.$ $NextSendUpdate\_State\_Conclusion(p,\ tt)$
      BY $\langle 2 \rangle 1,\ NextSendUpdate\_State$
  $\langle 2 \rangle 3.$ $NextSendUpdate\_GlobalIncomingInfo\_Conclusion(fq,\ p,\ tt)$
      BY $\langle 2 \rangle 1,\ NextSendUpdate\_GlobalIncomingInfo$

  $\langle 2 \rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$
  $\langle 2 \rangle$ USE DEF $NextSendUpdate\_GlobalIncomingInfo\_Conclusion$

  $\langle 2 \rangle 4.$ UNCHANGED $nrec$ BY $\langle 2 \rangle 2$
  $\langle 2 \rangle 5.$ UNCHANGED $globfq$ BY $\langle 2 \rangle 2$
  $\langle 2 \rangle 6.$ UNCHANGED GII BY $\langle 2 \rangle 3$
  $\langle 2 \rangle$ QED BY $\langle 2 \rangle 4,\ \langle 2 \rangle 5,\ \langle 2 \rangle 6,\ \langle 1 \rangle 6$

If the action is $NextReceiveUpdate$.

No change to $nrec$. When $fq \neq q$, no change to GII or $globfq$. When $fq = q$, removes delta from GII and adds it to $globfq$. In either case preserves the invariant.

$\langle 1 \rangle 9.$ CASE $NextReceiveUpdate$
  $\langle 2 \rangle 1.$ PICK $p \in Proc,\ q \in Proc$:
      $NextReceiveUpdate\_WithPQ(p,\ q)$
      BY $\langle 1 \rangle 9$ DEF $NextReceiveUpdate,\ NextReceiveUpdate\_WithPQ$

  $\langle 2 \rangle 2.$ $NextReceiveUpdate\_State\_Conclusion(p,\ q)$
      BY $\langle 2 \rangle 1,\ NextReceiveUpdate\_State$
  $\langle 2 \rangle 3.$ $NextReceiveUpdate\_GlobalIncomingInfo\_Conclusion(fq,\ p,\ q)$
      BY $\langle 2 \rangle 1,\ NextReceiveUpdate\_GlobalIncomingInfo$

  $\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$
  $\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_GlobalIncomingInfo\_Conclusion$

  $\langle 2 \rangle 4.$ UNCHANGED $nrec$ BY $\langle 2 \rangle 2$

  GII and $globfq$ are unchanged if $fq \neq q$.

  $\langle 2 \rangle 5.$ ASSUME $fq \neq q$ PROVE $nrec' = DeltaVecAdd(GII',\ globfq')$
    $\langle 3 \rangle 1.$ UNCHANGED GII BY $\langle 2 \rangle 3,\ \langle 2 \rangle 5$
    $\langle 3 \rangle 2.$ UNCHANGED $globfq$ BY $\langle 2 \rangle 2,\ \langle 2 \rangle 5$
    $\langle 3 \rangle$ QED BY $\langle 3 \rangle 1,\ \langle 3 \rangle 2,\ \langle 2 \rangle 4,\ \langle 1 \rangle 6$

  Transfer delta from GII to $globfq$ if $fq = q$.

  $\langle 2 \rangle 6.$ ASSUME $fq = q$ PROVE $nrec' = DeltaVecAdd(GII',\ globfq')$
    $\langle 3 \rangle$ DEFINE $delta \triangleq NextReceiveUpdate\_Kappa(p,\ q)$

$\langle 3 \rangle 1.\ delta \in DeltaVecType$ BY $\langle 2 \rangle 2$
$\langle 3 \rangle 2.\ GII = DeltaVecAdd(GII',\ delta)$ BY $\langle 2 \rangle 3,\ \langle 2 \rangle 6$
$\langle 3 \rangle 3.\ globfq' = DeltaVecAdd(globfq,\ delta)$ BY $\langle 2 \rangle 2,\ \langle 2 \rangle 6$

$\langle 3 \rangle$ QED   by commutative and associative properties of $DeltaVecAdd$
$\langle 4 \rangle$ HIDE DEF $delta,\ GII,\ globfq$ hide the complicated definitions
$\langle 4 \rangle$ USE $\langle 3 \rangle 1,\ \langle 1 \rangle 3,\ \langle 1 \rangle 4$ know that they are delta vectors
$\langle 4 \rangle$ USE $DeltaVecAddCommutative$ know that add is commutative
$\langle 4 \rangle$ USE $DeltaVecAddAssociative$ know that add is associative

$\langle 4 \rangle 1.\ nrec = DeltaVecAdd(DeltaVecAdd(GII',\ delta),\ globfq)$ BY $\langle 3 \rangle 2,\ \langle 1 \rangle 6$
$\langle 4 \rangle 2.\ nrec = DeltaVecAdd(GII',\ DeltaVecAdd(delta,\ globfq))$ BY $\langle 4 \rangle 1$
$\langle 4 \rangle 3.\ nrec = DeltaVecAdd(GII',\ DeltaVecAdd(globfq,\ delta))$ BY $\langle 4 \rangle 2$
$\langle 4 \rangle 4.\ nrec = DeltaVecAdd(GII',\ globfq')$ BY $\langle 4 \rangle 3,\ \langle 3 \rangle 3$
$\langle 4 \rangle 5.\ nrec' = DeltaVecAdd(GII',\ globfq')$ BY $\langle 4 \rangle 4,\ \langle 2 \rangle 4$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 5$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 5,\ \langle 2 \rangle 6$

$\langle 1 \rangle$ QED BY $\langle 1 \rangle 7,\ \langle 1 \rangle 8,\ \langle 1 \rangle 9$ DEF $Next$

$InvGlobalRecordCount$ holds in all reachable states.

THEOREM $ThmInvGlobalRecordCount \triangleq$
$Spec \Rightarrow \Box InvGlobalRecordCount$
PROOF
$\langle 1 \rangle$ DEFINE $I \triangleq$
$\land\ InvType$
$\land\ InvTempUpright$
$\land\ InvIncomingInfoUpright$
$\land\ InvGlobalRecordCount$

$\langle 1 \rangle\ Init \Rightarrow I$
$\langle 2 \rangle$ USE $ThmInitInvType$
$\langle 2 \rangle$ USE $ThmInitInvTempUpright$
$\langle 2 \rangle$ USE $ThmInitInvIncomingInfoUpright$
$\langle 2 \rangle$ USE $ThmInitInvGlobalRecordCount$
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle\ I \land [Next]_{vars} \Rightarrow I'$
$\langle 2 \rangle$ USE $ThmNextInvType$
$\langle 2 \rangle$ USE $ThmNextInvTempUpright$
$\langle 2 \rangle$ USE $ThmNextInvIncomingInfoUpright$

$\langle 2 \rangle$ USE $ThmNextInvGlobalRecordCount$
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle$ $Init \land \Box[Next]_{vars} \Rightarrow \Box I$ OMITTED $TLAPS$ cannot check it
$\langle 1 \rangle$ $Spec \Rightarrow \Box I$ OMITTED BY DEF $Spec$
$\langle 1 \rangle$ QED OMITTED $TLAPS$ cannot check it

C.25 Proof of invariant InvStickyNrecVacantUpto

MODULE $NaiadClockProofInvStickyNrecVacantUpto$

EXTENDS $NaiadClockProofInvGlobalRecordCount$

Proof of invariant InvStickyNrecVacantUpto.

$InvStickyNrecVacantUpto$ says that for each point t, if all points up to t have no records in the current state, then all points up to t will have no records in the next state.

This invariant is proved by the following argument:

(1) the number of records at each point is non-negative and

(2) the only action that changes the number of records is $NextPerformOperation$, which makes a change expressed as a regular delta vector.

$NrecVacantUpto$ is sticky.

THEOREM $ThmStickyNrecVacantUpto \triangleq$
  ASSUME
    $InvType$,
    $[Next]_{vars}$,
    NEW $ft \in Point$,
    $NrecVacantUpto(ft)$
  PROVE
  $NrecVacantUpto(ft)'$
PROOF
  $\langle 1 \rangle$ $InvType'$ BY $ThmNextInvType$

Dispose of the stutter step.

  $\langle 1 \rangle 1.$ CASE UNCHANGED $vars$
    $\langle 2 \rangle$ USE DEF $vars$
    $\langle 2 \rangle$ USE DEF $NrecVacantUpto$
    $\langle 2 \rangle$ QED BY $\langle 1 \rangle 1$

Set up to prove that $NrecVacantUpto(ft)$ is sticky.

  $\langle 1 \rangle$ SUFFICES ASSUME $Next$ PROVE $NrecVacantUpto(ft)'$ BY $\langle 1 \rangle 1$ DEF $Next$

  $\langle 1 \rangle 2.$ $lleq \in PointRelationType$ BY DEF $InvType$
  $\langle 1 \rangle 3.$ $IsPartialOrder(lleq)$ BY DEF $InvType$

  $\langle 1 \rangle$ DEFINE $a \preceq b \triangleq lleq[a][b]$
  $\langle 1 \rangle$ DEFINE $a \prec b \triangleq a \preceq b \land a \neq b$

If the action is $NextPerformOperation$.

$\langle 1\rangle 4.$ CASE $NextPerformOperation$
$\langle 2\rangle 1.$ PICK $p \in Proc,\ c \in PointToNat,\ r \in PointToNat :$
$NextPerformOperation\_WithPCR(p,\ c,\ r)$
BY $\langle 1\rangle 4$ DEF $NextPerformOperation,\ NextPerformOperation\_WithPCR$

$\langle 2\rangle 2.$ $NextPerformOperation\_State\_Conclusion(p,\ c,\ r)$
BY $\langle 2\rangle 1,\ NextPerformOperation\_State$

$\langle 2\rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$

$\langle 2\rangle$ DEFINE $delta \triangleq NextPerformOperation\_Delta(p,\ c,\ r)$

$\langle 2\rangle 3.$ UNCHANGED $lleq$ BY $\langle 2\rangle 2$
$\langle 2\rangle 4.$ $delta \in DeltaVecType$ BY $\langle 2\rangle 2$
$\langle 2\rangle 5.$ $IsDeltaVecUpright(lleq,\ delta)$ BY $\langle 2\rangle 2$
$\langle 2\rangle 6.$ $nrec' = DeltaVecAdd(nrec,\ delta)$ BY $\langle 2\rangle 2$

$\langle 2\rangle$ HIDE DEF $delta$

It suffices to show that $\forall\, s \preceq ft : nrec'[s] = 0$. So assume we have a counterexample and prove a contradiction.

$\langle 2\rangle 7.$ SUFFICES ASSUME NEW $s \in Point,\ s \preceq ft,\ nrec'[s] \neq 0$
PROVE FALSE
BY $\langle 2\rangle 3$ DEF $NrecVacantUpto,\ IsDeltaVecVacantUpto$

$\langle 2\rangle 8.$ $nrec'[s] = nrec[s] + delta[s]$ BY $\langle 2\rangle 6$ DEF $DeltaVecAdd$
$\langle 2\rangle 9.$ $nrec'[s] \in Nat$ BY DEF $InvType,\ CountVecType$
$\langle 2\rangle 10.$ $delta[s] \in Int$ BY $\langle 2\rangle 4$ DEF $DeltaVecType$
$\langle 2\rangle 11.$ $nrec[s] = 0$ BY $\langle 2\rangle 3,\ \langle 2\rangle 7$ DEF $NrecVacantUpto,\ IsDeltaVecVacantUpto$
$\langle 2\rangle 12.$ $delta[s] > 0$ BY $\langle 2\rangle 7,\ \langle 2\rangle 8,\ \langle 2\rangle 9,\ \langle 2\rangle 10,\ \langle 2\rangle 11,\ SMTT(10)$

$\langle 2\rangle 13.$ PICK $u \in Point : u \preceq s \land delta[u] < 0$
BY $\langle 2\rangle 4,\ \langle 2\rangle 5,\ \langle 2\rangle 12,\ \langle 1\rangle 2,\ \langle 1\rangle 3,\ DeltaVecUpright\_ExistsSupport$

$\langle 2\rangle 14.$ $u \preceq ft$ BY $\langle 2\rangle 3,\ \langle 2\rangle 7,\ \langle 2\rangle 13,\ \langle 1\rangle 2,\ \langle 1\rangle 3,\ PartialOrderTransitive$
$\langle 2\rangle 15.$ $nrec'[u] = nrec[u] + delta[u]$ BY $\langle 2\rangle 6$ DEF $DeltaVecAdd$
$\langle 2\rangle 16.$ $nrec'[u] \in Nat$ BY DEF $InvType,\ CountVecType$
$\langle 2\rangle 17.$ $nrec[u] = 0$ BY $\langle 2\rangle 3,\ \langle 2\rangle 14$ DEF $NrecVacantUpto,\ IsDeltaVecVacantUpto$
$\langle 2\rangle 18.$ $delta[u] \in Int$ BY $\langle 2\rangle 4$ DEF $DeltaVecType$

$\langle 2\rangle 19.$ $nrec'[u] < 0$ BY $\langle 2\rangle 13,\ \langle 2\rangle 15,\ \langle 2\rangle 17,\ \langle 2\rangle 18,\ SMTT(10)$
$\langle 2\rangle$ QED BY $\langle 2\rangle 16,\ \langle 2\rangle 19,\ SMTT(10)$

If the action is $NextSendUpdate$.

$\langle 1\rangle 5.$ CASE $NextSendUpdate$
$\langle 2\rangle 1.$ PICK $p \in Proc,\ tt \in$ SUBSET $Point :$
$NextSendUpdate\_WithPTT(p,\ tt)$
BY $\langle 1\rangle 5$ DEF $NextSendUpdate,\ NextSendUpdate\_WithPTT$

$\langle 2\rangle 2.$ $NextSendUpdate\_State\_Conclusion(p,\ tt)$
BY $\langle 2\rangle 1,$ $NextSendUpdate\_State$

$\langle 2\rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$

$\langle 2\rangle 4.$ UNCHANGED $nrec$ BY $\langle 2\rangle 2$
$\langle 2\rangle 5.$ UNCHANGED $lleq$ BY $\langle 2\rangle 2$
$\langle 2\rangle 6.$ UNCHANGED $NrecVacantUpto(ft)$ BY $\langle 2\rangle 4,$ $\langle 2\rangle 5$ DEF $NrecVacantUpto$
$\langle 2\rangle$ QED BY $\langle 2\rangle 6$

If the action is $NextReceiveUpdate$.

$\langle 1\rangle 6.$ CASE $NextReceiveUpdate$
$\langle 2\rangle 1.$ PICK $p \in Proc,\ q \in Proc :$
$NextReceiveUpdate\_WithPQ(p,\ q)$
BY $\langle 1\rangle 6$ DEF $NextReceiveUpdate,$ $NextReceiveUpdate\_WithPQ$

$\langle 2\rangle 2.$ $NextReceiveUpdate\_State\_Conclusion(p,\ q)$
BY $\langle 2\rangle 1,$ $NextReceiveUpdate\_State$

$\langle 2\rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$

$\langle 2\rangle 4.$ UNCHANGED $nrec$ BY $\langle 2\rangle 2$
$\langle 2\rangle 5.$ UNCHANGED $lleq$ BY $\langle 2\rangle 2$
$\langle 2\rangle 6.$ UNCHANGED $NrecVacantUpto(ft)$ BY $\langle 2\rangle 4,$ $\langle 2\rangle 5$ DEF $NrecVacantUpto$
$\langle 2\rangle$ QED BY $\langle 2\rangle 6$

$\langle 1\rangle$ QED BY $\langle 1\rangle 4,$ $\langle 1\rangle 5,$ $\langle 1\rangle 6$ DEF $Next$

$InvStickyNrecVacantUpto$ holds in the initial state.

THEOREM $ThmInitInvStickyNrecVacantUpto \triangleq$
$Init \Rightarrow InvStickyNrecVacantUpto$
PROOF
$\langle 1\rangle$ QED BY DEF $Init,$ $InvStickyNrecVacantUpto$

$InvStickyNrecVacantUpto$ carries through a $Next$ step.

THEOREM $ThmNextInvStickyNrecVacantUpto \triangleq$

$\wedge\ InvType$

$\wedge\ InvStickyNrecVacantUpto$

$\wedge\ [Next]_{vars}$

$\Rightarrow$

$InvStickyNrecVacantUpto'$

PROOF

$\langle 1 \rangle$ SUFFICES ASSUME

$InvType,$

$InvStickyNrecVacantUpto,$

$[Next]_{vars}$

PROVE $InvStickyNrecVacantUpto'$

OBVIOUS

Dispose of the stutter step.

$\langle 1 \rangle 1.$ CASE UNCHANGED $vars$

$\langle 2 \rangle$ USE DEF $vars$

$\langle 2 \rangle$ USE DEF $InvStickyNrecVacantUpto$

$\langle 2 \rangle$ USE DEF $NrecVacantUpto$

$\langle 2 \rangle$ QED BY $\langle 1 \rangle 1$

Set up to prove $InvStickyNrecVacantUpto'$.

$\langle 1 \rangle$ SUFFICES ASSUME $Next$ PROVE $InvStickyNrecVacantUpto'$ BY $\langle 1 \rangle 1$

$\langle 1 \rangle$ SUFFICES ASSUME NEW $ft \in Point$

PROVE $nrecvut[ft]' \Rightarrow NrecVacantUpto(ft)'$

BY DEF $InvStickyNrecVacantUpto$

$\langle 1 \rangle$ SUFFICES $nrecvut[ft]' = NrecVacantUpto(ft)$ BY $ThmStickyNrecVacantUpto$

If the action is $NextPerformOperation$.

$\langle 1 \rangle 7.$ CASE $NextPerformOperation$

$\langle 2 \rangle 1.$ PICK $p \in Proc,\ c \in PointToNat,\ r \in PointToNat :$

$NextPerformOperation\_WithPCR(p,\ c,\ r)$

BY $\langle 1 \rangle 7$ DEF $NextPerformOperation,\ NextPerformOperation\_WithPCR$

$\langle 2 \rangle 2.$ $NextPerformOperation\_State\_Conclusion(p,\ c,\ r)$

BY $\langle 2 \rangle 1,\ NextPerformOperation\_State$

$\langle 2 \rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$

If the action is $NextSendUpdate$.

$\langle 1 \rangle 8.$ CASE $NextSendUpdate$

$\langle 2 \rangle 1.$ PICK $p \in Proc,\ tt \in$ SUBSET $Point :$

$NextSendUpdate\_WithPTT(p, tt)$
BY $\langle 1 \rangle 8$ DEF $NextSendUpdate, NextSendUpdate\_WithPTT$

$\langle 2 \rangle 2.$ $NextSendUpdate\_State\_Conclusion(p, tt)$
BY $\langle 2 \rangle 1$, $NextSendUpdate\_State$

$\langle 2 \rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$

If the action is $NextReceiveUpdate$.

$\langle 1 \rangle 9.$ CASE $NextReceiveUpdate$
$\langle 2 \rangle 1.$ PICK $p \in Proc, q \in Proc$:
$NextReceiveUpdate\_WithPQ(p, q)$
BY $\langle 1 \rangle 9$ DEF $NextReceiveUpdate, NextReceiveUpdate\_WithPQ$

$\langle 2 \rangle 2.$ $NextReceiveUpdate\_State\_Conclusion(p, q)$
BY $\langle 2 \rangle 1$, $NextReceiveUpdate\_State$

$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 7$, $\langle 1 \rangle 8$, $\langle 1 \rangle 9$ DEF $Next$

$InvStickyNrecVacantUpto$ holds in all reachable states.

THEOREM $ThmInvStickyNrecVacantUpto \triangleq$
$Spec \Rightarrow \Box InvStickyNrecVacantUpto$
PROOF
$\langle 1 \rangle$ DEFINE $I \triangleq$
$\land InvType$
$\land InvStickyNrecVacantUpto$

$\langle 1 \rangle$ $Init \Rightarrow I$
$\langle 2 \rangle$ USE $ThmInitInvType$
$\langle 2 \rangle$ USE $ThmInitInvStickyNrecVacantUpto$
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle$ $I \land [Next]_{vars} \Rightarrow I'$
$\langle 2 \rangle$ USE $ThmNextInvType$
$\langle 2 \rangle$ USE $ThmNextInvStickyNrecVacantUpto$
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle$ $Init \land \Box[Next]_{vars} \Rightarrow \Box I$ OMITTED $TLAPS$ cannot check it

$\langle 1 \rangle$ $Spec \Rightarrow \Box I$ OMITTED BY DEF $Spec$
$\langle 1 \rangle$ QED OMITTED $TLAPS$ cannot check it

C.26 Proof of invariant InvStickyGlobVacantUpto

MODULE $NaiadClockProofInvStickyGlobVacantUpto$

EXTENDS $NaiadClockProofInvStickyNrecVacantUpto$

Proof of invariant InvStickyGlobVacantUpto.

$InvStickyGlobVacantUpto$ says that for each processor fq and point t, if $GlobVacantUpto(fq,\ t)$ is TRUE in the current state, then it will be TRUE in the next state. $GlobVacantUpto(fq,\ t)$ says that all points $s \preceq t$ have $glob[fq][s] = 0$.

This fact is proved as follows.

$NextPerformOperation$ makes no change to $glob[fq]$.

$NextSendUpdate$ makes no change to $glob[fq]$.

$NextReceiveUpdate$ takes the oldest update from $msg[p][q]$ and adds it to $glob[q]$. When $fq = q$ this is a change to $glob[fq]$. Let

$GII0 \triangleq GlobalIncomingInfo(0,\ q,\ q)$
$GII1 \triangleq GLobalIncomingInfo(1,\ q,\ q)$
$kappa \triangleq msg[p][q][1]$

We know that

$glob[q]$ is vacant up to t
$nrec$ is vacant up to t
$nrec = glob[q] + GII0$

Hence we have

$GII0$ is vacant up to t

We know that

$GII0 = kappa + GII1$ kappa is $GII1$-upright $GII1$ is upright

Hence by the $DeltaVecVacantUpto\_BetaUpright$ theorem we know that

kappa is vacant up to t

Since

$glob[q]' = glob[q]$ + kappa

We know that

$glob[q]'$ is vacant up to t

This completes the proof.

$GlobVacantUpto$ is sticky.

THEOREM $ThmStickyGlobVacantUpto \triangleq$

ASSUME
$InvType$,
$InvTempUpright$,
$InvIncomingInfoUpright$,
$InvGlobalRecordCount$,
$InvInfoAtBetaUpright$,
$[Next]_{vars}$,
NEW $fq \in Proc$,
NEW $ft \in Point$,
$GlobVacantUpto(fq, ft)$
PROVE
$GlobVacantUpto(fq, ft)'$
PROOF
$\langle 1 \rangle$ $InvGlobalIncomingInfoType$ BY $DeduceInvGlobalIncomingInfoType$
$\langle 1 \rangle$ $InvGlobalIncomingInfoSkip0$ BY $DeduceInvGlobalIncomingInfoSkip0$
$\langle 1 \rangle$ $InvGlobalIncomingInfoUpright$ BY $DeduceInvGlobalIncomingInfoUpright$
$\langle 1 \rangle$ $InvGlobalInfoAtBetaUpright$ BY $DeduceInvGlobalInfoAtBetaUpright$
$\langle 1 \rangle$ $InvGlobVacantUptoImpliesNrec$ BY $DeduceInvGlobVacantUptoImpliesNrec$

$\langle 1 \rangle$ $InvType'$ BY $ThmNextInvType$
$\langle 1 \rangle$ $InvTempUpright'$ BY $ThmNextInvTempUpright$
$\langle 1 \rangle$ $InvIncomingInfoUpright'$ BY $ThmNextInvIncomingInfoUpright$
$\langle 1 \rangle$ $InvGlobalRecordCount'$ BY $ThmNextInvGlobalRecordCount$
$\langle 1 \rangle$ $InvInfoAtBetaUpright'$ BY $ThmNextInvInfoAtBetaUpright$

$\langle 1 \rangle$ $InvGlobalIncomingInfoType'$ BY $DeduceInvGlobalIncomingInfoType$
$\langle 1 \rangle$ $InvGlobalIncomingInfoSkip0'$ BY $DeduceInvGlobalIncomingInfoSkip0$
$\langle 1 \rangle$ $InvGlobalIncomingInfoUpright'$ BY $DeduceInvGlobalIncomingInfoUpright$
$\langle 1 \rangle$ $InvGlobalInfoAtBetaUpright'$ BY $DeduceInvGlobalInfoAtBetaUpright$
$\langle 1 \rangle$ $InvGlobVacantUptoImpliesNrec'$ BY $DeduceInvGlobVacantUptoImpliesNrec$

Dispose of the stutter step.

$\langle 1 \rangle 1.$ CASE UNCHANGED $vars$
$\langle 2 \rangle$ USE DEF $vars$
$\langle 2 \rangle$ USE DEF $InvStickyGlobVacantUpto$
$\langle 2 \rangle$ USE DEF $GlobVacantUpto$
$\langle 2 \rangle$ QED BY $\langle 1 \rangle 1$

Set up to prove that $GlobVacantUpto(fq, ft)$ is sticky.

$\langle 1 \rangle$ SUFFICES ASSUME $Next$ PROVE $GlobVacantUpto(fq, ft)'$ BY $\langle 1 \rangle 1$, Isa DEF $Next$

$\langle 1 \rangle 2.$ $lleq \in PointRelationType$ BY DEF $InvType$
$\langle 1 \rangle 3.$ $IsPartialOrder(lleq)$ BY DEF $InvType$
$\langle 1 \rangle 4.$ $NrecVacantUpto(ft)$ BY DEF $InvGlobVacantUptoImpliesNrec$

If the action is $NextPerformOperation$.

$\langle 1 \rangle 6.$ CASE $NextPerformOperation$
$\langle 2 \rangle 1.$ PICK $p \in Proc,\ c \in PointToNat,\ r \in PointToNat :$
$NextPerformOperation\_WithPCR(p,\ c,\ r)$
BY $\langle 1 \rangle 6$ DEF $NextPerformOperation,\ NextPerformOperation\_WithPCR$

$\langle 2 \rangle 2.$ $NextPerformOperation\_State\_Conclusion(p,\ c,\ r)$
BY $\langle 2 \rangle 1,\ NextPerformOperation\_State$

$\langle 2 \rangle$ USE DEF $NextPerformOperation\_State\_Conclusion$

$\langle 2 \rangle 3.$ UNCHANGED $lleq$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 4.$ UNCHANGED $glob$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 5.$ UNCHANGED $GlobVacantUpto(fq,\ ft)$
BY $\langle 2 \rangle 3,\ \langle 2 \rangle 4$ DEF $GlobVacantUpto,\ IsDeltaVecVacantUpto$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 5$

If the action is $NextSendUpdate$.

$\langle 1 \rangle 7.$ CASE $NextSendUpdate$
$\langle 2 \rangle 1.$ PICK $p \in Proc,\ tt \in$ SUBSET $Point :$
$NextSendUpdate\_WithPTT(p,\ tt)$
BY $\langle 1 \rangle 7$ DEF $NextSendUpdate,\ NextSendUpdate\_WithPTT$

$\langle 2 \rangle 2.$ $NextSendUpdate\_State\_Conclusion(p,\ tt)$
BY $\langle 2 \rangle 1,\ NextSendUpdate\_State$

$\langle 2 \rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$

$\langle 2 \rangle 3.$ UNCHANGED $lleq$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 4.$ UNCHANGED $glob$ BY $\langle 2 \rangle 2$
$\langle 2 \rangle 5.$ UNCHANGED $GlobVacantUpto(fq,\ ft)$
BY $\langle 2 \rangle 3,\ \langle 2 \rangle 4$ DEF $GlobVacantUpto,\ IsDeltaVecVacantUpto$
$\langle 2 \rangle$ QED BY $\langle 2 \rangle 5$

If the action is $NextReceiveUpdate$.

$\langle 1 \rangle 8.$ CASE $NextReceiveUpdate$
$\langle 2 \rangle 1.$ PICK $p \in Proc,\ q \in Proc :$
$NextReceiveUpdate\_WithPQ(p,\ q)$
BY $\langle 1 \rangle 8$ DEF $NextReceiveUpdate,\ NextReceiveUpdate\_WithPQ$

$\langle 2 \rangle 2.$ $NextReceiveUpdate\_State\_Conclusion(p,\ q)$
BY $\langle 2 \rangle 1,\ NextReceiveUpdate\_State$
$\langle 2 \rangle 3.$ $NextReceiveUpdate\_IncomingInfo\_Conclusion(0,\ p,\ fq,\ p,\ q)$
BY $\langle 2 \rangle 1,\ NextReceiveUpdate\_IncomingInfo$
$\langle 2 \rangle 4.$ $NextReceiveUpdate\_GlobalIncomingInfo\_Conclusion(fq,\ p,\ q)$
BY $\langle 2 \rangle 1,\ NextReceiveUpdate\_GlobalIncomingInfo$

$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$

$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_IncomingInfo\_Conclusion$
$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_GlobalIncomingInfo\_Conclusion$

$\langle 2 \rangle 6.$ UNCHANGED $lleq$ BY $\langle 2 \rangle 2$

$glob[fq]$ is unchanged if $fq \neq q$.

$\langle 2 \rangle 7.$ ASSUME $fq \neq q$ PROVE $GlobVacantUpto(fq, ft)'$
$\langle 3 \rangle 1.$ UNCHANGED $glob[fq]$ BY $\langle 2 \rangle 2$, $\langle 2 \rangle 7$
$\langle 3 \rangle$ QED BY $\langle 3 \rangle 1$, $\langle 2 \rangle 6$ DEF $GlobVacantUpto$, $IsDeltaVecVacantUpto$

Transfer kappa from GII to $glob[fq]$ if $fq = q$.

$\langle 2 \rangle 8.$ ASSUME $fq = q$ PROVE $GlobVacantUpto(fq, ft)'$
$\langle 3 \rangle$ DEFINE $kappa \triangleq NextReceiveUpdate\_Kappa(p, q)$
$\langle 3 \rangle$ DEFINE $GII \triangleq GlobalIncomingInfo(0, fq, fq)$
$\langle 3 \rangle$ DEFINE $globfq \triangleq glob[fq]$

$\langle 3 \rangle 1.$ $kappa \in DeltaVecType$ BY $\langle 2 \rangle 2$
$\langle 3 \rangle 2.$ $GII \in DeltaVecType$ BY DEF $InvGlobalIncomingInfoType$
$\langle 3 \rangle 3.$ $GII' \in DeltaVecType$ BY DEF $InvGlobalIncomingInfoType$
$\langle 3 \rangle 4.$ $globfq \in DeltaVecType$ BY DEF $InvType$
$\langle 3 \rangle 5.$ $nrec \in DeltaVecType$ BY DEF $InvType$, $DeltaVecType$, $CountVecType$
$\langle 3 \rangle 6.$ $lleq' \in PointRelationType$ BY $\langle 2 \rangle 6$, $\langle 1 \rangle 2$
$\langle 3 \rangle 7.$ $IsPartialOrder(lleq')$ BY $\langle 2 \rangle 6$, $\langle 1 \rangle 3$

$\langle 3 \rangle 8.$ $IsDeltaVecBetaUpright(lleq', kappa, GII')$
$\langle 4 \rangle$ DEFINE $GII0 \triangleq GlobalIncomingInfo(0, p, fq)$
$\langle 4 \rangle$ DEFINE $GII1 \triangleq GlobalIncomingInfo(1, p, fq)$
$\langle 4 \rangle$ DEFINE $IA1 \triangleq InfoAt(1, p, fq)$
$\langle 4 \rangle 1.$ $IsDeltaVecBetaUpright(lleq, IA1, GII1)$ BY DEF $InvGlobalInfoAtBetaUpright$
$\langle 4 \rangle 2.$ $kappa = IA1$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 8$, $NextReceiveUpdate\_InfoAt1$
$\langle 4 \rangle 3.$ $GII' = GII0'$ BY DEF $InvGlobalIncomingInfoSkip0$
$\langle 4 \rangle 4.$ $GII0' = GII1$ BY $\langle 2 \rangle 1$, $\langle 2 \rangle 8$, $NextReceiveUpdate\_GlobalIncomingInfo1$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 4 \rangle 3$, $\langle 4 \rangle 4$, $\langle 2 \rangle 6$

$\langle 3 \rangle 9.$ $GII = DeltaVecAdd(kappa, GII')$
$\langle 4 \rangle 1.$ $GII = DeltaVecAdd(GII', kappa)$ BY $\langle 2 \rangle 4$, $\langle 2 \rangle 8$
$\langle 4 \rangle$ HIDE DEF GII, $kappa$
$\langle 4 \rangle$ USE $\langle 3 \rangle 1$, $\langle 3 \rangle 3$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $DeltaVecAddCommutative$

$\langle 3 \rangle 10.$ $IsDeltaVecVacantUpto(lleq', globfq, ft)$ BY $\langle 2 \rangle 6$ DEF $GlobVacantUpto$

$\langle 3 \rangle 11.$ $IsDeltaVecVacantUpto(lleq', GII, ft)$
$\langle 4 \rangle 1.$ $nrec = DeltaVecAdd(GII, globfq)$ BY DEF $InvGlobalRecordCount$
$\langle 4 \rangle 2.$ $IsDeltaVecVacantUpto(lleq', nrec, ft)$ BY $\langle 2 \rangle 6$, $\langle 1 \rangle 4$ DEF $NrecVacantUpto$
$\langle 4 \rangle$ HIDE DEF GII, $globfq$
$\langle 4 \rangle$ USE $\langle 3 \rangle 2$, $\langle 3 \rangle 4$, $\langle 3 \rangle 5$, $\langle 3 \rangle 6$
$\langle 4 \rangle$ QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$, $\langle 3 \rangle 10$, $DeltaVecVacantUpto\_Add$

$\langle 3\rangle 12.$ $IsDeltaVecVacantUpto(lleq', DeltaVecAdd(kappa, GII'), ft)$ BY $\langle 3\rangle 9$, $\langle 3\rangle 11$

$\langle 3\rangle 13.$ $IsDeltaVecVacantUpto(lleq', kappa, ft)$
$\langle 4\rangle 1.$ $IsDeltaVecUpright(lleq', GII')$ BY DEF $InvGlobalIncomingInfoUpright$
$\langle 4\rangle$ HIDE DEF GII, $kappa$
$\langle 4\rangle$ USE $\langle 3\rangle 1$, $\langle 3\rangle 3$, $\langle 3\rangle 6$, $\langle 3\rangle 7$
$\langle 4\rangle$ QED BY $\langle 4\rangle 1$, $\langle 3\rangle 8$, $\langle 3\rangle 12$, $DeltaVecVacantUpto\_BetaUpright$

$\langle 3\rangle 14.$ $IsDeltaVecVacantUpto(lleq', globfq', ft)$
$\langle 4\rangle 1.$ $globfq' = DeltaVecAdd(globfq, kappa)$ BY $\langle 2\rangle 2$, $\langle 2\rangle 8$
$\langle 4\rangle$ HIDE DEF $kappa$, $globfq$
$\langle 4\rangle$ USE $\langle 3\rangle 1$, $\langle 3\rangle 4$, $\langle 3\rangle 6$
$\langle 4\rangle$ QED BY $\langle 4\rangle 1$, $\langle 3\rangle 10$, $\langle 3\rangle 13$, $DeltaVecVacantUpto\_Add$

$\langle 3\rangle$ QED BY $\langle 3\rangle 14$ DEF $GlobVacantUpto$
$\langle 2\rangle$ QED BY $\langle 2\rangle 7$, $\langle 2\rangle 8$
$\langle 1\rangle$ QED BY $\langle 1\rangle 6$, $\langle 1\rangle 7$, $\langle 1\rangle 8$ DEF $Next$

$InvStickyGlobVacantUpto$ holds in the initial state.

THEOREM $ThmInitInvStickyGlobVacantUpto \triangleq$
$Init \Rightarrow InvStickyGlobVacantUpto$
PROOF
$\langle 1\rangle$ QED BY Isa DEF $Init$, $InvStickyGlobVacantUpto$

$InvStickyGlobVacantUpto$ carries through a $Next$ step.

THEOREM $ThmNextInvStickyGlobVacantUpto \triangleq$
$\wedge\ InvType$
$\wedge\ InvTempUpright$
$\wedge\ InvIncomingInfoUpright$
$\wedge\ InvGlobalRecordCount$
$\wedge\ InvInfoAtBetaUpright$
$\wedge\ InvStickyGlobVacantUpto$
$\wedge\ [Next]_{vars}$
$\Rightarrow$
$InvStickyGlobVacantUpto'$

PROOF
⟨1⟩ SUFFICES ASSUME
$InvType$,
$InvTempUpright$,
$InvIncomingInfoUpright$,
$InvGlobalRecordCount$,
$InvInfoAtBetaUpright$,
$InvStickyGlobVacantUpto$,
$[Next]_{vars}$
PROVE $InvStickyGlobVacantUpto'$
OBVIOUS

Dispose of the stutter step.

⟨1⟩1. CASE UNCHANGED $vars$
⟨2⟩ USE DEF $vars$
⟨2⟩ USE DEF $InvStickyGlobVacantUpto$
⟨2⟩ USE DEF $GlobVacantUpto$
⟨2⟩ QED BY ⟨1⟩1

Set up to prove $InvStickyGlobVacantUpto'$.

⟨1⟩ SUFFICES ASSUME $Next$ PROVE $InvStickyGlobVacantUpto'$ BY ⟨1⟩1

⟨1⟩ SUFFICES ASSUME NEW $fq \in Proc$, NEW $ft \in Point$
PROVE $globvut[fq][ft]' \Rightarrow GlobVacantUpto(fq, ft)'$
BY DEF $InvStickyGlobVacantUpto$

⟨1⟩ SUFFICES $globvut[fq][ft]' = GlobVacantUpto(fq, ft)$ BY $ThmStickyGlobVacantUpto$

If the action is $NextPerformOperation$.

⟨1⟩7. CASE $NextPerformOperation$
⟨2⟩1. PICK $p \in Proc,\ c \in PointToNat,\ r \in PointToNat$:
$NextPerformOperation\_WithPCR(p, c, r)$
BY ⟨1⟩7 DEF $NextPerformOperation$, $NextPerformOperation\_WithPCR$

⟨2⟩2. $NextPerformOperation\_State\_Conclusion(p, c, r)$
BY ⟨2⟩1, $NextPerformOperation\_State$

⟨2⟩ USE DEF $NextPerformOperation\_State\_Conclusion$

⟨2⟩ QED BY ⟨2⟩2

If the action is $NextSendUpdate$.

⟨1⟩8. CASE $NextSendUpdate$
⟨2⟩1. PICK $p \in Proc,\ tt \in$ SUBSET $Point$:
$NextSendUpdate\_WithPTT(p, tt)$
BY ⟨1⟩8 DEF $NextSendUpdate$, $NextSendUpdate\_WithPTT$

$\langle 2 \rangle 2.$ $NextSendUpdate\_State\_Conclusion(p,\ tt)$
BY $\langle 2 \rangle 1,\ NextSendUpdate\_State$

$\langle 2 \rangle$ USE DEF $NextSendUpdate\_State\_Conclusion$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$

If the action is $NextReceiveUpdate$.

$\langle 1 \rangle 9.$ CASE $NextReceiveUpdate$
$\langle 2 \rangle 1.$ PICK $p \in Proc,\ q \in Proc$:
$NextReceiveUpdate\_WithPQ(p,\ q)$
BY $\langle 1 \rangle 9$ DEF $NextReceiveUpdate,\ NextReceiveUpdate\_WithPQ$

$\langle 2 \rangle 2.$ $NextReceiveUpdate\_State\_Conclusion(p,\ q)$
BY $\langle 2 \rangle 1,\ NextReceiveUpdate\_State$

$\langle 2 \rangle$ USE DEF $NextReceiveUpdate\_State\_Conclusion$

$\langle 2 \rangle$ QED BY $\langle 2 \rangle 2$
$\langle 1 \rangle$ QED BY $\langle 1 \rangle 7,\ \langle 1 \rangle 8,\ \langle 1 \rangle 9$ DEF $Next$

$InvStickyGlobVacantUpto$ holds in all reachable states.

THEOREM $ThmInvStickyGlobVacantUpto \triangleq$
$Spec \Rightarrow \Box InvStickyGlobVacantUpto$
PROOF
$\langle 1 \rangle$ DEFINE $I \triangleq$
$\land\ InvType$
$\land\ InvTempUpright$
$\land\ InvIncomingInfoUpright$
$\land\ InvGlobalRecordCount$
$\land\ InvInfoAtBetaUpright$
$\land\ InvStickyGlobVacantUpto$

$\langle 1 \rangle$ $Init \Rightarrow I$
$\langle 2 \rangle$ USE $ThmInitInvType$
$\langle 2 \rangle$ USE $ThmInitInvTempUpright$
$\langle 2 \rangle$ USE $ThmInitInvIncomingInfoUpright$
$\langle 2 \rangle$ USE $ThmInitInvGlobalRecordCount$
$\langle 2 \rangle$ USE $ThmInitInvInfoAtBetaUpright$
$\langle 2 \rangle$ USE $ThmInitInvStickyGlobVacantUpto$
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle$ $I \land [Next]_{vars} \Rightarrow I'$
  $\langle 2 \rangle$ USE $ThmNextInvType$
  $\langle 2 \rangle$ USE $ThmNextInvTempUpright$
  $\langle 2 \rangle$ USE $ThmNextInvIncomingInfoUpright$
  $\langle 2 \rangle$ USE $ThmNextInvGlobalRecordCount$
  $\langle 2 \rangle$ USE $ThmNextInvInfoAtBetaUpright$
  $\langle 2 \rangle$ USE $ThmNextInvStickyGlobVacantUpto$
  $\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle$ $Init \land \Box[Next]_{vars} \Rightarrow \Box I$ OMITTED *TLAPS* cannot check it
$\langle 1 \rangle$ $Spec \Rightarrow \Box I$ OMITTED BY DEF *Spec*
$\langle 1 \rangle$ QED OMITTED *TLAPS* cannot check it

C.27 The top-level proof module

MODULE $NaiadClockProof$

EXTENDS $NaiadClockProofInvStickyGlobVacantUpto$

The top-level proof module.

This module presents the top-level theorems, which are proved by appealing to earlier theorems and temporal deductions. Unfortunately, $TLAPS$ is unable to check the temporal deductions.

In any execution that obeys $Spec$, the safety property $SafeStickyNrecVacantUpto$ always holds.

$TLAPS$ is unable to check the temporal steps in this proof.

THEOREM $ThmSafeStickyNrecVacantUpto \triangleq$
  $Spec \Rightarrow \Box SafeStickyNrecVacantUpto$
PROOF
  $\langle 1 \rangle$ SUFFICES ASSUME NEW $t \in Point$
    PROVE
      $(Init \land \Box[Next]_{vars})$
      $\Rightarrow$
      $\Box(NrecVacantUpto(t) \Rightarrow \Box NrecVacantUpto(t))$
    OMITTED   BY DEF $Spec$, $SafeStickyNrecVacantUpto$

  $\langle 1 \rangle$ DEFINE $I \triangleq$
      $\land InvType$

  $\langle 1 \rangle 1.$ $Init \Rightarrow I$
    $\langle 2 \rangle$ USE $ThmInitInvType$
    $\langle 2 \rangle$ QED OBVIOUS

  $\langle 1 \rangle 2.$ $(I \land [Next]_{vars}) \Rightarrow I'$
    $\langle 2 \rangle$ USE $ThmNextInvType$
    $\langle 2 \rangle$ QED OBVIOUS

  $\langle 1 \rangle 3.$ $(I \land NrecVacantUpto(t) \land [Next]_{vars}) \Rightarrow NrecVacantUpto(t)'$
    BY $ThmStickyNrecVacantUpto$

  $\langle 1 \rangle$ QED OMITTED $TLAPS$ cannot check it

In any execution that obeys $Spec$, the safety property $SafeStickyGlobVacantUpto$ always holds.

$TLAPS$ is unable to check the temporal steps in this proof.

THEOREM $ThmSafeStickyGlobVacantUpto \triangleq$
$Spec \Rightarrow \Box SafeStickyGlobVacantUpto$
PROOF
$\langle 1 \rangle$ SUFFICES ASSUME NEW $q \in Proc$, NEW $t \in Point$
PROVE
$(Init \wedge \Box[Next]_{vars})$
$\Rightarrow$
$\Box(GlobVacantUpto(q,\ t) \Rightarrow \Box GlobVacantUpto(q,\ t))$
OMITTED BY DEF $Spec$, $SafeStickyGlobVacantUpto$

$\langle 1 \rangle$ DEFINE $I \triangleq$
$\wedge\ InvType$
$\wedge\ InvTempUpright$
$\wedge\ InvIncomingInfoUpright$
$\wedge\ InvGlobalRecordCount$
$\wedge\ InvInfoAtBetaUpright$

$\langle 1 \rangle 1.\ Init \Rightarrow I$
$\langle 2 \rangle$ USE $ThmInitInvType$
$\langle 2 \rangle$ USE $ThmInitInvTempUpright$
$\langle 2 \rangle$ USE $ThmInitInvIncomingInfoUpright$
$\langle 2 \rangle$ USE $ThmInitInvGlobalRecordCount$
$\langle 2 \rangle$ USE $ThmInitInvInfoAtBetaUpright$
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle 2.\ (I \wedge [Next]_{vars}) \Rightarrow I'$
$\langle 2 \rangle$ USE $ThmNextInvType$
$\langle 2 \rangle$ USE $ThmNextInvTempUpright$
$\langle 2 \rangle$ USE $ThmNextInvIncomingInfoUpright$
$\langle 2 \rangle$ USE $ThmNextInvGlobalRecordCount$
$\langle 2 \rangle$ USE $ThmNextInvInfoAtBetaUpright$
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle 3.\ (I \wedge GlobVacantUpto(q,\ t) \wedge [Next]_{vars}) \Rightarrow GlobVacantUpto(q,\ t)'$
BY $ThmStickyGlobVacantUpto$

$\langle 1 \rangle$ QED OMITTED $TLAPS$ cannot check it

In any execution that obeys $Spec$, the safety property $SafeGlobVacantUptoImpliesStickyNrec$ always holds.

$TLAPS$ is unable to check the temporal steps in this proof.

THEOREM $ThmSafeGlobVacantUptoImpliesStickyNrec \triangleq$
$Spec \Rightarrow \Box SafeGlobVacantUptoImpliesStickyNrec$
PROOF
$\langle 1 \rangle$ SUFFICES ASSUME NEW $q \in Proc$, NEW $t \in Point$
PROVE
$(Init \land \Box[Next]_{vars})$
$\Rightarrow$
$\Box(GlobVacantUpto(q, t) \Rightarrow \Box NrecVacantUpto(t))$
OMITTED BY DEF $Spec$, $SafeGlobVacantUptoImpliesStickyNrec$

$\langle 1 \rangle$ DEFINE $I \triangleq$
$\land\ InvType$
$\land\ InvTempUpright$
$\land\ InvIncomingInfoUpright$
$\land\ InvGlobalIncomingInfoUpright$
$\land\ InvGlobalRecordCount$
$\land\ InvGlobVacantUptoImpliesNrec$

$\langle 1 \rangle 1.$ $Init \Rightarrow I$
$\langle 2 \rangle$ USE $ThmInitInvType$
$\langle 2 \rangle$ USE $ThmInitInvTempUpright$
$\langle 2 \rangle$ USE $ThmInitInvIncomingInfoUpright$
$\langle 2 \rangle$ USE $ThmInitInvGlobalRecordCount$
$\langle 2 \rangle$ USE $DeduceInvGlobalIncomingInfoUpright$
$\langle 2 \rangle$ USE $DeduceInvGlobVacantUptoImpliesNrec$
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle 2.$ $(I \land [Next]_{vars}) \Rightarrow I'$
$\langle 2 \rangle$ USE $ThmNextInvType$
$\langle 2 \rangle$ USE $ThmNextInvTempUpright$
$\langle 2 \rangle$ USE $ThmNextInvIncomingInfoUpright$
$\langle 2 \rangle$ USE $ThmNextInvGlobalRecordCount$
$\langle 2 \rangle$ USE $DeduceInvGlobalIncomingInfoUpright$
$\langle 2 \rangle$ USE $DeduceInvGlobVacantUptoImpliesNrec$
$\langle 2 \rangle$ QED OBVIOUS

$\langle 1 \rangle 3.$ $(I \land GlobVacantUpto(q, t)) \Rightarrow NrecVacantUpto(t)$
BY DEF $InvGlobVacantUptoImpliesNrec$

$\langle 1 \rangle 4.$ $(I \land NrecVacantUpto(t) \land [Next]_{vars}) \Rightarrow NrecVacantUpto(t)'$
BY $ThmStickyNrecVacantUpto$

$\langle 1 \rangle$ QED OMITTED $TLAPS$ cannot check it

Index of Theorems