Contractor Report

Mechanical Verification of a
Schematic Byzantine Clock
Synchronization Algorithm
Mechanical Verification of a Schematic Byzantine Clock Synchronization Algorithm

Natarajan Shankar
SRI International
Computer Science Laboratory
Menlo Park, California

Prepared for
Langley Research Center
under Contract NAS1-18226
Contents

1 Introduction 1

2 Schneider's Schema for Clock Synchronization 6
   2.1 Defining Clocks .......................... 6
   2.2 Clock conditions ........................ 8
   2.3 The Correctness Proof ...................... 14
      2.3.1 Overview ............................ 15
      2.3.2 The Proof ............................ 16
   2.4 ICA as an instance of Schneider's scheme 22

3 The Verification of Schneider's Protocol using EHDM 25
   3.1 The Clock Assumptions ........................ 25
   3.2 The Proof Highlights ....................... 30

4 Conclusions 34

References 37

A Proof Summary 38

B The Complete EHDM Proof 40
   Multiplication .................................. 40
   Absolutes ...................................... 42
   Division ....................................... 44
   Arith ......................................... 48
   Countmod ...................................... 49
   Clockassumptions .............................. 51
   Basics ........................................ 55
   Readbounds ................................... 63
Lemma 3 ............................................. 68
Lemma_final ........................................... 79
Ica ....................................................... 82
Ica2 ..................................................... 85
Ica3 ..................................................... 94
Ica4 ..................................................... 98

C Proof Chain Analysis .................................... 108
C.1 Proof Chain for Agreement .............................. 108
C.2 Proof Chain for ICA Translation Invariance ........... 114
C.3 Proof Chain for ICA Precision Enhancement ............ 117
C.4 Proof Chain for ICA Accuracy Preservation ............. 122
List of Figures

3.1 Declarations from module clockassumptions .............. 26
3.2 Constants in module clockassumptions .................. 27
3.3 Physical clock axioms in module clockassumptions ........ 28
3.4 Clock definitions in module clockassumptions .......... 28
3.5 Conditions on Logical Clocks in module clockassumptions . 29
3.6 Main Theorem in module lemma_final .................. 30
3.7 Skew immediately following resynchronization from module readbounds ........................................... 30
3.8 Skew up to ith resynchronization from module lemma3 .... 31
3.9 Egocentric mean from module ica ....................... 32
3.10 Properties of egocentric mean from modules ica, ica3, and ica4 33
Chapter 1

Introduction

Synchronizing clocks in the presence of faults is a classic problem in distributed computing. Even the most accurate clocks do drift at significant rates, both with respect to a time standard and relative to each other. In order for independent processors to exhibit cooperative behavior, it is often required that their local clocks be synchronized. Such synchrony is the basis for distributed algorithms that use timeouts, time stamps, and rounds of message passing. Synchronization is also assumed when the same computation is executed on multiple, independent processors in order to mask processor failures. Digital avionics systems constitute a typical example of the need for synchronized clocks. In these systems, the results of multiple redundant processors are voted to ensure a high degree of fault tolerance, and the processor clocks must be synchronized in order to carry this out. Clock synchronization problems led to the scrubbing of the first scheduled launch of the NASA Space Shuttle [4], and to anomalous behavior of the Voyager spacecraft [5]. Butler [6] presents a survey of various clock synchronization protocols.

Synchronizing clocks in the presence of faults is a difficult problem. If synchrony is maintained by periodically broadcasting a global clock value to each of the processors, the failure of the global clock then becomes critical. On the other hand, if each processor has its own local clock and these clocks are initially synchronized, they might slowly drift apart so that with time the system loses its ability to behave synchronously. It is therefore necessary to periodically resynchronize the clocks. We are concerned here with algorithms that perform this resynchronization in a fault tolerant manner. In the cases we consider, the clocks are required to be synchronized only
with respect to each other and not with respect to some external standard clock. The primary requirement that any solution must satisfy is that at any instant, the absolute difference, or the skew, between two clock readings should be within some bound $\delta$. The secondary requirement is that there must be a small bound on the correction required to keep clocks in synchrony. The latter requirement prevents trivial solutions that, for example, reset the clocks to zero at each round of synchronization. We restrict our focus to the primary requirement, since the secondary requirement turns out to be a straightforward consequence of one of the assumptions for the operation of the protocol studied here.

To implement synchronized clocks, each processor has a physical clock whose drift rate with respect to a fixed standard time is bounded. We refer to the fixed standard time as real time. In addition to the physical clock, each processor maintains a logical, or virtual, clock that is computed by periodically applying an adjustment to the reading of the physical clock. The adjustment to be applied at the end of each period is determined by means of a synchronization protocol. The application of such an adjustment could be continuous so that the individual clock ticks are either sped up or slowed down, but no clock ticks are dropped or repeated. Alternately, the adjustment could be applied in an instantaneous manner, in which case, some clock ticks might be dropped or repeated. In the latter situation, critical events should not be scheduled during these clock ticks. This report only considers the case of instantaneous clock adjustments. These results are therefore applicable to the class of systems that have a synchronization phase followed by a period of normal operation in each cycle of synchronization. The results here can be extended to the case of continuous clock adjustments. Schneider [1] presents an analysis of continuous adjustments.

To take a somewhat coarse look at clock synchronization, suppose that the various physical clocks start synchronized and drift apart from real time at a rate not exceeding $\rho$. For example, a clock might gain or lose up to a minute every hour. The processors operate normally for a period $R$ of, say, an hour. The processors then engage in a round of synchronization during which they exchange clock values. Assume for simplicity that the communication between clocks occurs instantaneously. At some mutually agreeable instant, the processors reset their clocks to some mutually agreeable value such as the average of their clock readings. Thus at the end of such a round of synchronization, the skew between clocks vanishes. Clearly, if we want the clocks to be no more than $\delta$ apart, the period $R$ between synchronizations should not exceed $\delta/2\rho$. Given that $\rho$ is a minute per hour, and $R$ is
an hour, $\delta$ can be no less than two minutes.

The above outline obviously makes a great many simplifying assumptions, but it does capture the basic process of clock synchronization. The most significant invalid assumption is that clocks and processors do not fail. Clock synchronization protocols ought to be able to tolerate a certain number of processor failures since they are often used to synchronize multiple processors in fault-tolerant architectures. When processors do fail, they could do so in the worst possible way by exhibiting arbitrarily different behaviors towards different processors, e.g., by "maliciously" communicating different clock values to different processors. Such failures are known as Byzantine failures [7]. Consider the case of three clocks $a$, $b$, and $c$, when $a$ reads 12 noon, $b$ reads 11:59 am, and $c$ has failed. To resynchronize, they exchange clock values and $c$ maliciously communicates its value as 12:01 pm to $a$ and as 11:58 am to $b$. Suppose each clock is resynchronized by taking the average of all the clock values observed by it, then $a$ resets itself to 12 noon and $b$ resets itself to 11:59 am. The clocks are thus no closer following resynchronization than immediately prior to resynchronization. Thus the clocks can continue to drift even further apart until the next round of synchronization.

The above scenario illustrates one of the earliest clock synchronization protocols capable of tolerating Byzantine processor failures: the Interactive Convergence Algorithm (ICA) of Lamport and Melliar-Smith [3]. ICA tolerates up to $\lfloor (N-1)/3 \rfloor$ failures for $N$ processors. In ICA, a processor $p$ resynchronizes for the $i$'th time when its clock reads $iR$. Processor $p$ then reads the difference between the other clock readings and its own clock reading. By ignoring clock differences larger than a certain value $\Delta$, processor $p$ computes the egocentric mean of the acceptable clock differences as the correction required to resynchronize its clock. Rushby and von Henke [8] have subjected Lamport and Melliar-Smith's proof of correctness to mechanical scrutiny using EHDM. As is often the case with fault-tolerant distributed protocols, the original proof is both subtle and complex. The mechanical verification was able to identify and correct several minor flaws, and to significantly streamline the proof.

Schneider [1] presents a clock synchronization scheme that generalizes protocols such as ICA. Schneider's clock synchronization scheme (abbreviated here as SCS) regards each logical clock as being periodically reset to a value computed by a convergence function. The egocentric mean of ICA is an instance of such a convergence function. Schneider places certain natural conditions on the behavior of suitable convergence functions and shows
that these conditions are sufficient for bounding the skew between the resulting logical clocks. He also shows that the convergence functions used by a number of existing protocols satisfy these restrictions. Such a schematic presentation of Byzantine clock synchronization provides an elegant framework for understanding various individual protocols, and greatly simplifies the proofs of their correctness.

Since the SCS protocol captures the mathematics behind Byzantine clock synchronization in an abstract and schematic manner, it makes an interesting candidate for verification. The schematic nature of the SCS protocol makes it convenient to subsequently verify a number of specific protocols as instances of the SCS protocol. Also, Schneider's analysis employs a global "real time" rather than clock time as its frame of reference, i.e., clocks map real time to clock time. Lamport and Melliar-Smith's analysis [3] of ICA and the verification by Rushby and von Henke [8] were both carried out in terms of clocks that mapped clock time to real time. The use of clock time as a frame of reference makes some of the mathematics is fairly cumbersome and also makes the specification harder to understand. It seems reasonable to assume that to each real time instant, there is a unique clock reading, but not quite as reasonable to insist that there is a unique real time instant corresponding to a clock reading since a failed clock could exhibit the same reading at different real time instants. It is, of course, possible to explain away such objections. The question of what is the best framework for specifying such protocols is, to our knowledge, still open.

The mechanical verification of the SCS protocol was carried out using the E\textsc{HDM} verification system developed at the Computer Science Laboratory of SRI International. The egocentric mean function of the ICA protocol was also verified as satisfying Schneider's restrictions. The SCS protocol and its informal proof are presented in Chapter 2. An overview of the mechanically checked proof is presented in Chapter 3. The appendices contain the complete listing of the proof that was presented as input to E\textsc{HDM}.

The use of E\textsc{HDM} to check the proof led to the clarification of a number of details from Schneider's original presentation without tampering unduly with the outline and intent of his argument. Schneider's proof employs a monotonicity condition on convergence functions that was found to be inessential for the proof. The monotonicity condition actually fails for ICA and other similar convergence functions (see Section 2.4). Schneider's proof requires certain relations to hold between the convergence behavior of the convergence function, the drift rate of the physical clocks, the error in communicating clock values, and the time between synchronization rounds. The
machine proof clears up some minor inaccuracies in Schneider's derivation of these relations.

Acknowledgements. John Rushby supplied much of the background and guidance for this work. Friedrich von Henke helped me get started with EHDM. I am also grateful to Fred Schneider and Rick Butler for their encouragement.
Chapter 2

Schneider’s Schema for Clock Synchronization

Schneider shows that a number of known algorithms for synchronizing Byzantine clocks can be presented in a uniform manner so that their individual proofs are greatly simplified [1]. The exposition below follows Schneider’s outline quite closely, but revises a number of the details in the description of the protocol as well as the proof. Section 2.1 describes how the logical clock is computed from the physical clock using the convergence function. Section 2.2 describes the conditions on the behavior of clocks and on suitable convergence functions. The proof of correctness of clock synchronization from the conditions of Section 2.2 is outlined in Section 2.3.

2.1 Defining Clocks

The physical and logical clocks are presented as functions from real time (as given by some external standard) to clock readings. This real time thus forms the frame of reference and is often referred to simply as “time.” The variable \( t \) ranges over this real time. Synchronization takes place in rounds. The time at which processor \( p \) adjusts its clock following the \( i \)'th round of synchronization is represented by \( t_p^i \). The starting time \( t_p^0 \) which is the time from which the system is observed, is taken to be zero.

In our abstraction, both the real time and the clock readings can be interpreted as ranging over the real numbers or the rationals. The ordered

\[1\] In the original presentation of the interactive convergence algorithm, clocks are represented as functions from clock time to the external standard time [3, 8].
field axioms that are used are satisfied by both the real numbers and the rationals. The term $PC_p(t)$ is the reading of $p$'s physical clock at real time $t$. The adjusted virtual clock reading at time $t_p^i$ is computed by applying an adjustment $adj_p^i$ to the physical clock reading $PC_p(t_p^i)$. In its $i$'th interval of operation, i.e., when $t_p^i \leq t < t_p^{i+1}$, the virtual clock reading, $VC_p(t)$ is given by $PC_p(t) + adj_p^i$. At round 0, the adjustment $adj_p^0$ is taken to be 0 so that for $t < t_p^1$, the reading $VC_p(t)$ is just $PC_p(t)$. In other words, in the first period of operation, each clock takes its physical clock reading as its virtual clock reading. This means that for synchronization over the first period, we need as a condition, a bound on the initial skews between the physical clocks of nonfaulty processors.

For $i > 0$, we let $\Theta_p^i$ be an array of clock readings so that $\Theta_p^i(q)$ is $p$'s reading of $q$'s clock at time $t_p^i$. In the EHDM formalization, the array of observed clock readings $\Theta_p^i$, is actually represented as a function from clocks to readings. The corrected value of $VC_p(t_p^i)$ is computed by a convergence function, $cfn(p, \Theta_p^i)$. The adjustment $adj_p^i$ to be applied to the physical clock is therefore given by the difference $cfn(p, \Theta_p^i) - PC_p(t_p^i)$. Since $\Theta_p^i$ is a function, $cfn$ is a higher-order function.

The above explanation of $\Theta_p^i(q)$ does not specify whether $q$'s physical or virtual clock is the one that is read by clock $p$. Note that if $t_p^i$ preceded $t_p^i$, then $q$'s virtual clock has already been adjusted for the $i$'th time at time $t_p^i$. In Schneider's model, $\Theta_p^i(q)$ is a reading of $q$'s virtual clock at time $t_p^i$ but ignoring the $i$'th correction that may have already been applied to $q$'s clock. This value is represented by an abstraction called the interval clock. The interval clock reading $IC_q^i(t)$ is given by $PC_q(t) + adj_q^i$. Thus for $i > 0$, the value $\Theta_p^i(q)$ is $p$'s reading of $IC_q^{i-1}(t_p^i)$. The rationale for introducing an interval clock is that the observed clock readings in the protocol are based on readings exchanged prior to synchronization. The interval clock is an abstraction that is useful for describing the protocol and it need not actually be implemented. The physical and virtual clocks are of course both implemented.

The above description leads to following definitions where $i$ ranges over the natural numbers and $t > 0$.

\[
adj_p^{i+1} = cfn(p, \Theta_p^{i+1}) - PC_p(t_p^{i+1}) \tag{2.1.1}
\]

\[
adj_p^0 = 0 \tag{2.1.2}
\]

\[
IC_p^i(t) = PC_p(t) + adj_p^i \tag{2.1.3}
\]

\[
VC_p(t) = IC_p^i(t), \text{ for } t_p^i \leq t < t_p^{i+1} \tag{2.1.4}
\]
It is easy to derive the following from Definitions (2.1.1), (2.1.3), and (2.1.4).

\[
    VC_p(t_{i+1}^p) = IC_p(t_{i+1}^p) = cfn(p, \Theta_p^{i+1}) \quad (2.1.5)
\]

\[
    IC_p(t) = cfn(p, \Theta_p^i) + PC_p(t) - PC_p(t_{i+1}^p) \quad (2.1.6)
\]

So far we have merely defined the virtual and interval clock functions in terms of the physical clock function \(PC_p(t)\), the synchronization times \(t_i^p\), and the convergence function \(cfn\) applied to the clock readings \(\Theta_p^i\). In the next section, we enumerate Schneider’s constraints on these quantities when \(p\) is a nonfaulty, or correct, processor. The main result we obtain from these constraints and the above definitions is a bound \(\delta\) on the skew between the logical clocks of two correct processors \(p\) and \(q\).

**Theorem 2.1.1 (bounded skew)** For any two clocks \(p\) and \(q\) that are nonfaulty at time \(t\),

\[
    |VC_p(t) - VC_q(t)| \leq \delta \quad (2.1.7)
\]

The proof of Theorem 2.1.1 is outlined in Section 2.3.1.

### 2.2 Clock conditions

In formalizing the laws constraining the behavior of individual clocks, we must ensure that no assumptions are made regarding the faulty clocks since we are dealing with Byzantine failures. These laws which are conditions on the behavior of clocks are enumerated as axioms within the boxes below. Individual protocols and clock implementations are expected to satisfy these conditions.

The conditions constraining the behavior of clocks employ a number of constants represented by lowercase Greek letters. All of these constants are taken to be non-negative.

Section 2.1 above described how the processors go through rounds of synchronization. The proof of Theorem 2.1.7 is by induction on the number of rounds. The main idea of the proof is to show that the virtual clocks are within \(\delta_S\) immediately following a round of synchronization, and the skew between them does not exceed \(\delta\) in the following period until the next round of synchronization. To start, the following condition asserts that the nonfaulty clocks are synchronized to within the quantity \(\delta_S\) at time 0.
Condition 1 (initial skew) For nonfaulty processors $p$ and $q$

$$|PC_p(0) - PC_q(0)| \leq \delta_s$$  \hspace{1cm} (2.2.8)

The nonfaulty physical clocks must keep good enough time so that they do not drift away from real time by a rate greater than $\rho$.

Condition 2 (bounded drift) There is a nonnegative constant $\rho$ such that if clock $p$ is nonfaulty at time $s$, $s \geq t$, then

$$(1 - \rho)(s - t) \leq PC_p(s) - PC_p(t) \leq (1 + \rho)(s - t)$$  \hspace{1cm} (2.2.9)

A useful corollary to bounded drift is that two physical clocks $p$ and $q$ that are not faulty at time $s$, for $s \geq t$, can drift further apart over the interval $s - t$ by $2\rho(s - t)$, since both $p$ and $q$ can drift by $\rho(s - t)$ with respect to real time, but in opposite directions.

$$|PC_p(s) - PC_q(s)| \leq |PC_p(t) - PC_q(t)| + 2\rho(s - t)$$  \hspace{1cm} (2.2.10)

Each protocol has some mechanism for triggering the resynchronization of the clocks. Schneider postulates the existence of a global synchronization signal, $t^i_G$, which occurs at a period bounded from above and below. One can usually interpret $t^i_G$ as the real-time instant when the first nonfaulty processor decides to resynchronize for the $i$'th time. Schneider's conditions on $t^i_G$ are stated in terms of positive constants which we name $lo$, $hi$, and $wid$. His first condition is that the period $t^{i+1}_G - t^i_G$ is bounded from below by $lo$, and from above by $hi$. The second condition bounds the delay in receiving the trigger so that $t^i_G - t^i_G \leq wid$, for nonfaulty $p$.

Our description of the proof uses a slightly different set of parameters in order to dispense with the notion of a global synchronization signal used in Schneider's formulation. The parameters below seem easier to identify.

---

2In the description of the machine verification, great pains are taken to indicate the times at which the clocks are required to be nonfaulty. The rest of the informal outline of the proof makes the simplifying assumption that clocks are either faulty or nonfaulty, and disregards the time at which clocks are asserted as being nonfaulty.
for the various instances of Schneider's protocol. The different choice of parameters do not affect the proof of correctness in any significant way. For individual synchronization protocols, it should be possible to derive one set of parameters from the other.

**Condition 3 (bounded interval)** *For nonfaulty clock p*

$$0 < r_{\text{min}} \leq t_{p}^{i+1} - t_{p}^{i} \leq r_{\text{max}}$$  \hspace{1cm} (2.2.11)

**Condition 4 (bounded delay)** *For nonfaulty clocks p and q*

$$|t_{q}^{i} - t_{p}^{i}| \leq \beta$$ \hspace{1cm} (2.2.12)

**Condition 5 (initial synchronization)** *For nonfaulty clock p*

$$t_{p}^{0} = 0$$ \hspace{1cm} (2.2.13)

From the conditions of *bounded interval* and *bounded delay* above, it follows that if $\beta \leq r_{\text{min}}$, then $t_{p}^{i} \leq r_{q}^{i+1}$ for nonfaulty clocks p and q; i.e., there is no overlap between the i'th and the (i + 1)'th rounds of synchronization. Since we do want the synchronization rounds not to overlap, we state the following as a condition. If the periods were allowed to overlap, then the protocol would be difficult to implement since p could have started its (i + 1)'th clock before another processor q had started its i'th clock.

**Condition 6 (nonoverlap)**

$$\beta \leq r_{\text{min}}$$ \hspace{1cm} (2.2.14)

Another corollary of the *bounded interval* and *bounded delay* conditions is that for any two nonfaulty clocks p and q, we can derive,

$$0 \leq t_{p}^{i+1} - t_{q}^{i} \leq r_{\text{max}} + \beta.$$ \hspace{1cm} (2.2.15)

For nonfaulty clocks p and q, $\Theta_{p}^{i+1}(q)$ represents p's observation of q's i'th clock reading at time $t_{p}^{i+1}$, i.e., it is p's estimate of $IC_{q}^{i}(t_{p}^{i+1})$. The error
in this reading is assumed to be bounded by $\Lambda$.

**Condition 7 (reading error)** For nonfaulty clocks $p$ and $q$,

$$|IC_q^i(t_p^{i+1}) - \Theta_p^{i+1}(q)| \leq \Lambda \quad (2.2.16)$$

The above conditions turn out to be sufficient to bound the skew in the period between successive rounds of synchronization in terms of the skew bound $\delta_S$ immediately following synchronization. The conditions below of **bounded faults**, **translation invariance**, and **precision enhancement**, are needed to derive the skew bound $\delta_S$. The condition of **accuracy preservation** below is needed to bound the skew between virtual clocks when, for instance, $q$ has synchronized for the $i$'th time but $p$ has not.

The parameter $N$ is the total number of processors, and $F$ is the maximum number of faulty clocks that the algorithm is expected to tolerate. This property of the system is captured by the following condition.

**Condition 8 (bounded faults)** At any time $t$, the number processors faulty at time $t$ is at most $F$.

The conditions below are mathematical constraints placed on the convergence function, e.g., clocks, drifts, and failures, do not play any role in the statements. The isolation of the constraints makes it possible to demonstrate that the egocentric mean function of ICA satisfies the conditions of **translation invariance**, **precision enhancement**, and **accuracy preservation**, in purely mathematical terms. Note that these conditions do not make any distinction between the faulty and the nonfaulty clocks but are instead given in terms of a subset $C$ of clocks satisfying certain mathematical constraints.

Suppose that $t_p^i \geq t_q^i$ for nonfaulty $p$ and $q$, then in order to compute $\delta_S$, we are interested in comparing the clock times for $p$ and $q$ at $t_p^i$, the time when clocks $p$ and $q$ have both just been synchronized for the $i$'th time. Processor $q$ starts its $i$'th interval clock at $t_q^i$ with value $cfn(q, \Theta_q^i)$, so that its reading at $t_p^i$ is $cfn(q, \Theta_q^i) + x$, where $x = PC_q(t_p^i) - PC_q(t_q^i)$. The condition of **translation invariance** indicates that adding $x$ to the value
of the convergence function should be the same as adding \( x \) to each clock reading instead. Recall that the array of clock readings is represented by a function from clocks to readings so that \( cfn \) is a higher-order function.

\[
\text{Condition 9 (translation invariance) For any function } \theta \text{ mapping clocks to clock values,}
\]

\[
cfn(p, (\lambda n: \theta(n) + x)) = cfn(p, \theta) + x \tag{2.2.17}
\]

As a consequence of translation invariance, we know that at \( t_p^i \), both \( p \) and \( q \) have been resynchronized and \( VC_q(t_p^i) = cfn(q, (\lambda n: \Theta_q(n) + x)) \) for some \( x \), and \( VC_p(t_p^{i+1}) = cfn(p, \Theta_p) \). We clearly need some condition to bound the difference between these two values of the convergence function to within \( \delta_s \). The condition of precision enhancement allows exactly such a comparison between values of the convergence function based on the range of values of some subset of the clock readings that intuitively correspond to the readings of nonfaulty clocks.

In the statement of precision enhancement, \( \gamma \) and \( \theta \) are any two arrays (or functions) of clock readings, and \( C \) is to be intuitively interpreted as the subset of nonfaulty processors. This interpretation of \( C \) is permissible by the bounded faults condition. The reason it is not directly taken to be the set of nonfaulty clocks is because the protocol cannot assume that any individual clock can distinguish the faulty from the nonfaulty clocks. The convergence functions for some protocols can neglect readings of nonfaulty clocks while considering readings of faulty clocks.

Precision enhancement is used to bound the skew between two clocks immediately after both have been resynchronized whereas accuracy preservation is used to bound the skew between a clock that has been resynchronized and one that has yet to be resynchronized in the \( i \)th round. The condition of precision enhancement bounds the skew between two clocks as computed by the convergence function, based on the skews between the clock readings that are inputs to the convergence function. We will refer to the clocks in \( C \) as \( C \)-clocks. Precision enhancement then asserts that if the readings of different \( C \)-clocks in \( \gamma \) fall within a range \( y \) as do the \( C \)-clock readings in \( \theta \), and the corresponding readings in \( \gamma \) and in \( \theta \) of any \( C \)-clock differ by no
more than $x$, then $cfn(p, \gamma)$ and $cfn(q, \theta)$ are within $\pi(x, y)$ of each other.\(^4\) The parameter $y$ will roughly correspond to the amount by which the clocks have drifted relative to each other and $x$ roughly indicates the message delay in communicating clock values. Typically, the parameter $y$ dominates $x$. The quantity $\pi(x, y)$ provides the bound on the skew $\delta_S$ immediately following resynchronization. For the precision to be truly enhanced, it is crucial for $\pi(x, y)$ to be smaller than $y$.

**Condition 10 (precision enhancement)** Given any subset $C$ of the $N$ clocks with $|C| \geq N - F$, and clocks $p$ and $q$ in $C$, then for any readings $\gamma$ and $\theta$ satisfying the conditions

1. for any $l$ in $C$, $|\gamma(l) - \theta(l)| \leq x$
2. for any $l, m$ in $C$, $|\gamma(l) - \gamma(m)| \leq y$
3. for any $l, m$ in $C$, $|\theta(l) - \theta(m)| \leq y$

there is a bound $\pi(x, y)$, such that

$$|cfn(p, \gamma) - cfn(q, \theta)| \leq \pi(x, y) \quad (2.2.18)$$

The final condition of accuracy preservation bounds the distance between the value of $cfn(p, \theta)$ and the nonfaulty entries in $\theta$. If $t_q^i \leq t_p^i$, then accuracy preservation\(^5\) can be used to bound the difference between $IC^i_q(t_q^i + 1)$ and $IC^i_p(t_p^i + 1)$.

\(^4\)Note that the order of arguments to $\pi$ are reversed from their order in Schneider's description [1].

\(^5\)Footnote 7 in Schneider [1] explains the choice of the terms precision enhancement and accuracy preservation. ‘Precision’ is defined as the closeness with which a measurement can be reproduced, whereas ‘accuracy’ is the proximity of the measurement to the actual value being measured. The virtual clocks represent various measurements of real time. The condition of precision enhancement characterizes the closeness of these measurements to each other. The condition of accuracy preservation can be seen as bounding the drift rate of the virtual clock with respect to real time.
Condition 11 (accuracy preservation) Given any subset $C$ of the $N$ clocks with $|C| \geq N - F$, and clock readings $\theta$ such that for any $l$ and $m$ in $C$, the bound $|\theta(l) - \theta(m)| \leq x$ holds, there is a bound $\alpha(x)$ such that for any $q$ in $C$

$$|cfn(p, \theta) - \theta(q)| \leq \alpha(x)$$ (2.2.19)

In addition to the conditions enumerated above, Schneider presents a condition called monotonicity that is actually not satisfied by several clock synchronization protocols. Fortunately, this condition turns out to be unnecessary in the derivation. The monotonicity condition asserts that if for each processor $l$, $\theta(l) \geq \gamma(l)$, then $cfn(p, \theta) \geq cfn(p, \gamma)$. The failure of the monotonicity condition for ICA is demonstrated in Section 2.4.

### 2.3 The Correctness Proof

The proof described below closely follows Schneider's outline. A few of the details are different, mainly reflecting corrections or perceived improvements. These seemingly small revisions do, however, lead to drastic changes in the statements of many of the theorems. The details of the correctness proof are both conceptually and notationally complicated. The formal arguments are extremely delicate to carry out carefully and correctly due to the additional consideration of processor failure. The true difficulty of constructing watertight proofs may not be apparent in the descriptions below since they only capture the end result of a mechanical verification and not the tenuous intermediate steps. It would be extremely difficult for even the most diligent mathematician to correctly capture all the details of such proofs without machine assistance. One difficulty is the care that is needed to ensure that no assumptions are made regarding failed clocks. Schneider [1], for instance, asserts, “We make no assumptions about the behavior of clocks at faulty processors — not even that they can be modeled by functions.” The present formulation does not go as far as to avoid the use of functions to model the behavior of failed clocks but no constraints are placed on the values of these functions when a processor has failed. The use of functions does not seem to contradict any intuitive understanding of the physical behavior of failed clocks. The possibility of processor failure adds significantly to the complexity of the formalization as well as the proof.
The proof described in this section is itself a somewhat simplified rendering of the mechanically verified proof. The main difference is that in the mechanical proof, the faultiness of a processor is itself a time-varying property, i.e., processors can fail at any time. A brief overview is given below to provide an outline of the detailed proof. The words processor and clock are used interchangeably.

2.3.1 Overview

To establish the main result, Theorem 2.1.1, we must show that the skew, or absolute difference, between the readings of any two nonfaulty clocks $p$ and $q$ at time $t$, given by $|VC_p(t) - VC_q(t)|$, is bounded by a quantity $\delta$. By the definition of $VC$ in (2.1.4), this reduces to the following two cases:

1. When both clocks have been resynchronized for the $i$'th time but not for the $(i+1)$'th time, i.e., if $\max(t_p^i, t_q^i) \leq t < \min(t_p^{i+1}, t_q^{i+1})$, then the skew between $IC_p^i(t)$ and $IC_q^i(t)$ is bounded by $\delta$, and

2. When only one clock, say $q$, has been resynchronized for the $(i+1)$'th time, i.e., if $t_q^{i+1} \leq t < t_p^{i+1}$, then the skew between $IC_p^i(t)$ and $IC_q^{i+1}(t)$ is bounded by $\delta$.

For two nonfaulty clocks $p$ and $q$, the time immediately following their $i$'th round of synchronization is $\max(t_p^i, t_q^i)$. The main step in the argument is to show that the skew between the readings $IC_p^i(t)$ and $IC_q^i(t)$ at time $t = \max(t_p^i, t_q^i)$, is bounded by a quantity $\delta_S$. This is shown by induction on $i$, and employs the conditions of initial skew, translation invariance, and precision enhancement.

We now know that the clocks $IC_p^i$ and $IC_q^i$ start off no more than $\delta_S$ apart at $\max(t_p^i, t_q^i)$. By bounded interval and bounded drift, the skew between $IC_p^i(t)$ and $IC_q^i(t)$ does not increase by more than $2\rho r_{\max}$ in the interval $\max(t_p^i, t_q^i) \leq t < \min(t_p^{i+1}, t_q^{i+1})$. Assuming that $t_q^{i+1} \leq t_p^{i+1}$, then the restriction of accuracy preservation on the convergence function is used to bound the skew between $IC_p^i(t_q^{i+1})$ and $IC_q^{i+1}(t_q^{i+1})$. By bounded delay and bounded drift, the additional skew between the readings $IC_p^i(t)$ and $IC_q^{i+1}(t)$ over the interval $t_q^{i+1} \leq t < t_p^{i+1}$ is no more than $2\rho \beta$. To obtain the final result, we need to constrain the quantities $\rho$, $\delta_S$, $r_{\min}$, $r_{\max}$, and $\beta$ so that the skew bounds derived over the various intervals are within $\delta$. Schneider also shows that the restrictions of translation invariance, precision enhancement,
and accuracy preservation, are satisfied by many of the known Byzantine fault tolerant convergence functions [1].

2.3.2 The Proof

The details of the proof of bounded skew are presented below. Let $t_{p,q}^{i+1}$ denote $\max(t_p^i, t_q^i)$. The first major step in Schneider's proof is to prove:

Theorem 2.3.1 There is a bound $\delta_S$ such that for synchronization round $i$ and any two nonfaulty processors $p$ and $q$

$$|IC_p^i(t_{p,q}^i) - IC_q^i(t_{p,q}^i)| \leq \delta_S.$$  (2.3.20)

Proof. The proof of Theorem 2.3.1 is by induction on the round number $i$.

Base case: When $i = 0$, by (2.2.13) we have $t_p^0 = t_q^0 = 0$. Then by Definitions (2.1.3) and (2.1.1), $IC_p^0(t_p^0) = PC_p(0)$ and $IC_q^0(t_q^0) = PC_q(0)$. The condition of initial skew asserts $|PC_p(0) - PC_q(0)| \leq \delta_S$. Hence, $|IC_p^0(0) - IC_q^0(0)|$ is also bounded by $\delta_S$.

Induction case: The induction hypothesis asserts that for every pair of nonfaulty processors, $l$ and $m$

$$|IC_l^i(t_{l,m}^i) - IC_m^i(t_{l,m}^i)| \leq \delta_S.$$  (2.3.21)

The goal is to establish for any pair of nonfaulty processors $p$ and $q$, that

$$|IC_p^{i+1}(t_{p,q}^{i+1}) - IC_q^{i+1}(t_{p,q}^{i+1})| \leq \delta_S.$$  (2.3.22)

Without loss of generality, assume that $t_q^{i+1}$ precedes $t_p^{i+1}$ so that $t_{p,q}^{i+1} = t_q^{i+1}$. Then Equation (2.1.6) yields

$$IC_q(t_q^{i+1}) = cfn(q, \Theta_q^{i+1}) + PC_q(t_q^{i+1}) - PC_q(t_q^{i+1}).$$  (2.3.23)

By Equation (2.1.5), we have

$$IC_p^{i+1}(t_p^{i+1}) = cfn(p, \Theta_p^{i+1}).$$  (2.3.24)
The condition of translation invariance provides an estimate of $IC_{q_{i+1}}^{i+1}(t_{p_{i+1}})$ in terms of the convergence function $cfn$. With $\Theta_{q_{i+1}}^{i+1}$ for $\theta$ in Equation (2.2.17), we get

$$cfn(q, \Theta_{q_{i+1}}^{i+1}) + PC_{q}(t_{p_{i+1}}^{i+1}) - PC_{q}(t_{p_{i+1}})$$

$$= cfn(q, (\lambda n: \Theta_{q_{i+1}}^{i+1}(n) + PC_{q}(t_{p_{i+1}}^{i+1}) - PC_{q}(t_{q_{i+1}}^{i+1}))). \quad (2.3.25)$$

By (2.3.24) and (2.3.25), the bound on the initial skews can be rewritten as follows:

$$|IC_{q_{i+1}}^{i+1}(t_{p_{i+1}}^{i+1}) - IC_{p_{i+1}}^{i+1}(t_{p_{i+1}}^{i+1})|$$

$$= |cfn(q, (\lambda n: \Theta_{q_{i+1}}^{i+1}(n) + PC_{q}(t_{p_{i+1}}^{i+1}) - PC_{q}(t_{q_{i+1}}^{i+1}))) - cfn(p, \Theta_{p_{i+1}}^{i+1})|.$$

(2.3.26)

The right-hand side of (2.3.26) can be bounded by $\pi(x, y)$ for some $x$ and $y$ using precision enhancement with $(\lambda n: \Theta_{q_{i+1}}^{i+1}(n) + PC_{q}(t_{p_{i+1}}^{i+1}) - PC_{q}(t_{q_{i+1}}^{i+1}))$ for $\gamma$ and $\Theta_{p_{i+1}}^{i+1}$ for $\theta$. The set $C$ in precision enhancement is taken to be the subset of nonfaulty clocks as permitted by bounded faults. The next few steps demonstrate that the remaining hypotheses of precision enhancement can be satisfied with these substitutions. To satisfy Hypothesis 1, we need to find an $z$ such that for any nonfaulty $l$ we can derive

$$|\Theta_{q_{i+1}}^{i+1}(l) + PC_{q}(t_{p_{i+1}}^{i+1}) - PC_{q}(t_{q_{i+1}}^{i+1}) - \Theta_{p_{i+1}}^{i+1}(l)| \leq z.$$ 

As shown below, the value $2\rho\beta + 2\Lambda$ can be substituted for $x$. By Equation (2.2.16), we easily get

$$|IC_{i_{i+1}}^{i+1}(t_{p_{i+1}}^{i+1}) - \Theta_{i_{i+1}}^{i+1}(l)| \leq \Lambda, \quad (2.3.27)$$

$$|IC_{i_{i+1}}^{i+1}(t_{p_{i+1}}^{i+1}) - \Theta_{i_{i+1}}^{i+1}(l)| \leq \Lambda. \quad (2.3.28)$$

Note that $t_{p_{i+1}}^{i+1} - t_{q_{i+1}}^{i+1} \leq \beta$ by (2.2.12). So from Equation (2.1.3) and bounded drift, we have

$$|(IC_{i_{i+1}}^{i+1}(t_{p_{i+1}}^{i+1}) + PC_{q}(t_{p_{i+1}}^{i+1}) - PC_{q}(t_{q_{i+1}}^{i+1})) - IC_{i_{i+1}}^{i+1}(t_{p_{i+1}}^{i+1})|$$

$$= |(PC_{q}(t_{p_{i+1}}^{i+1}) - PC_{q}(t_{q_{i+1}}^{i+1})) - (IC_{i_{i+1}}^{i+1}(t_{p_{i+1}}^{i+1}) - IC_{i_{i+1}}^{i+1}(t_{p_{i+1}}^{i+1}))|$$

$$= |(PC_{q}(t_{p_{i+1}}^{i+1}) - PC_{q}(t_{q_{i+1}}^{i+1})) - (PC_{q}(t_{p_{i+1}}^{i+1}) - PC_{q}(t_{q_{i+1}}^{i+1}))|$$

$$\leq |(1 + \rho)(t_{p_{i+1}}^{i+1} - t_{q_{i+1}}^{i+1}) - (1 - \rho)(t_{q_{i+1}}^{i+1} - t_{p_{i+1}}^{i+1})|$$

$$= |2\rho(t_{p_{i+1}}^{i+1} - t_{q_{i+1}}^{i+1})|$$

$$\leq 2\rho\beta. \quad (2.3.29)$$
Putting together Equations (2.3.27), (2.3.28), and (2.3.29), we get the required inequality

$$|\Theta_q^{i+1}(l) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}) - \Theta_p^{i+1}(l)| \leq 2\rho + 2\Lambda.$$  \hspace{1cm} (2.3.30)

The substitution $2\rho + 2\Lambda$ for $x$ thus satisfies Hypothesis 1 of precision enhancement.

The next step is to satisfy Hypotheses 2 and 3 of precision enhancement for the specified substitutions. For these, we need a $y$ such that for any nonfaulty processors $l$ and $m$, the following inequalities hold.

$$|\Theta_q^{i+1}(l) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}) - (\Theta_q^{i+1}(m) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}))| \leq y \hspace{1cm} (2.3.31)$$

$$|\Theta_p^{i+1}(l) - \Theta_p^{i+1}(m)| \leq y \hspace{1cm} (2.3.32)$$

Since (2.3.31) can be simplified by cancellation, both (2.3.31) and (2.3.32) can derived by deriving a bound $y$ such that for all nonfaulty clocks $k$, $l$, and $m$, we get

$$|\Theta_k^{i+1}(l) - \Theta_k^{i+1}(m)| \leq y \hspace{1cm} (2.3.33)$$

First note that

$$|\Theta_k^{i+1}(l) - \Theta_k^{i+1}(m)| \leq |\Theta_k^{i+1}(l) - IC_l^i(t_k^{i+1})| + |IC_l^i(t_k^{i+1}) - IC_m^i(t_k^{i+1})| + |\Theta_k^{i+1}(m) - IC_m^i(t_k^{i+1})| \hspace{1cm} (2.3.34)$$

In (2.3.34), we know by Equation (2.2.16) that

$$|\Theta_k^{i+1}(l) - IC_l^i(t_k^{i+1})| \leq \Lambda \hspace{1cm} (2.3.35)$$

$$|\Theta_k^{i+1}(m) - IC_m^i(t_k^{i+1})| \leq \Lambda \hspace{1cm} (2.3.36)$$

By the induction hypothesis (2.3.21), we get

$$|IC_l^i(t_{i,m}) - IC_m^i(t_{i,m})| \leq \delta_s. \hspace{1cm} (2.3.37)$$

We know by (2.2.15) that, $t_k^{i+1} - t_{i,m} \leq \tau_{\text{max}} + \beta$. Then by (2.1.3), (2.2.10), and (2.3.37), we get

$$|IC_l^i(t_k^{i+1}) - IC_m^i(t_k^{i+1})| \leq \delta_s + 2\rho(\tau_{\text{max}} + \beta). \hspace{1cm} (2.3.38)$$
Combining Equations (2.3.34), (2.3.35), (2.3.36), and (2.3.38), we get
\[ |\Theta_k^{t+1}(r(t)) - \Theta_k^{t+1}(m)| \leq \delta_S + 2\rho(r_{max} + \beta) + 2\Lambda. \]  
(2.3.39)

So the expression \( \delta_S + 2\rho(r_{max} + \beta) + 2\Lambda \) is the required bound \( y \) satisfying both Hypotheses 2 and 3 of precision enhancement.

If we now choose \( \delta_S \) so that
\[ \pi(2\Lambda + 2\beta\rho, \delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) \leq \delta_S, \]  
(2.3.40)

then the conclusion of precision enhancement along with Equation (2.1.6) ensures that
\[ |IC_{p}^{i+1}(t_{p}^{i+1}) - IC_{q}^{i+1}(t_{q}^{i+1})| \leq \delta_S \]

to complete the proof of Theorem 2.3.1. \( \blacksquare \)

We have now shown that for any pair of nonfaulty processors \( p \) and \( q \), the skew between their clock readings at \( t_{p,q}^{i} \), given by \( |IC_{p}^{i}(t_{p,q}^{i}) - IC_{q}^{i}(t_{p,q}^{i})| \), does not exceed \( \delta_S \). The next step is to show that for any \( i \), the clock skew between \( t_{p,q}^{i} \) and \( t_{p,q}^{i+1} \), is bounded.

**Theorem 2.3.2** For any two nonfaulty clocks \( p, q \), and \( t_{p,q}^{i} \leq t < t_{p,q}^{i+1} \),
\[ |VC_{p}(t) - VC_{q}(t)| \leq \delta. \]  
(2.3.41)

**Proof.** Assume without loss of generality that \( t_{q}^{i+1} \leq t_{p}^{i+1} \). The proof has two cases according to whether \( t_{p,q}^{i} \leq t < t_{q}^{i+1} \) or \( t_{q}^{i+1} \leq t < t_{p}^{i+1} \).

**Case 1:** Assuming \( t_{p,q}^{i} \leq t < t_{q}^{i+1} \), from bounded interval we get \( t - t_{p,q}^{i} \leq r_{max} \). By Equation (2.1.4), it is clear that for \( t \) in this interval \( VC_{p}(t) = IC_{p}^{i}(t) \) and \( VC_{q}(t) = IC_{q}^{i}(t) \). Then by (2.2.10) and (2.1.3), it follows that
\[ |VC_{p}(t) - VC_{q}(t)| \leq |VC_{p}(t_{p,q}^{i}) - VC_{q}(t_{p,q}^{i})| + 2\rho r_{max}. \]  
(2.3.42)

Recall that Theorem 2.3.1 yields
\[ |VC_{p}(t_{p,q}^{i}) - VC_{q}(t_{p,q}^{i})| \leq \delta_S. \]  
(2.3.43)

Combining Equations (2.3.42) and (2.3.43), we have
\[ |VC_{p}(t) - VC_{q}(t)| \leq \delta_S + 2\rho r_{max}. \]  
(2.3.44)

The bound \( \delta \) should therefore be chosen so that
\[ \delta_S + 2\rho r_{max} \leq \delta. \]  
(2.3.45)
Case 2: Assuming \( t^{i+1}_q < t < t^{i+1}_p \). In this interval, \( VC_q(t) = IC^{i+1}_q(t) \), whereas \( VC_p(t) = IC^{i}_p(t) \). The strategy here is to bound the skew at \( t^{i+1}_q \) and then compute the additional quantity by which the clocks can drift apart in the given interval. By Equations (2.1.5) and (2.1.4), we have

\[
|VC_p(t^{i+1}_q) - VC_q(t^{i+1}_q)| = |IC^{i}_p(t^{i+1}_q) - cf(q, \Theta^{i+1}_q)|. \tag{2.3.46}
\]

We now need to use the condition of accuracy preservation with \( C \) as the subset of nonfaulty processors as allowed by bounded faults. To satisfy the hypothesis of accuracy preservation, we need a bound \( x \) such that, for any pair of nonfaulty clocks \( l \) and \( m \),

\[
|\Theta^{i+1}_q(l) - \Theta^{i+1}_q(m)| \leq x. \tag{2.3.47}
\]

The next few steps are similar to those required to establish Hypotheses 2 and 3 of precision enhancement. By Equation (2.2.16), we have

\[
\begin{align*}
|\Theta^{i+1}_q(l) - IC^{i}_q(t^{i+1}_q)| & \leq \Lambda \tag{2.3.48} \\
|\Theta^{i+1}_q(m) - IC^{i}_m(t^{i+1}_q)| & \leq \Lambda. \tag{2.3.49}
\end{align*}
\]

By Equation (2.2.15), \( t^{i+1}_q - t^{i+1}_m \leq \tau_{max} + \beta \) holds. Theorem 2.3.1 and (2.2.10) can now be applied to get

\[
|IC^{i}_q(t^{i+1}_q) - IC^{i}_m(t^{i+1}_q)| \leq \delta_S + 2\rho(\tau_{max} + \beta). \tag{2.3.50}
\]

Letting \( x \) be \( \delta_S + 2\rho(\tau_{max} + \beta) + 2\Lambda \), and substituting \( p \) for \( q \) and \( q \) for \( p \) in accuracy preservation, we can combine Equations (2.3.48), (2.3.49), and (2.3.50), to get

\[
|cf(q, \Theta^{i+1}_q) - \Theta^{i+1}_q(p)| \leq \alpha(\delta_S + 2\rho(\tau_{max} + \beta) + 2\Lambda). \tag{2.3.51}
\]

Since Equation (2.2.16) yields \( |\Theta^{i+1}_q(p) - IC^{i}_p(t^{i+1}_q)| \leq \Lambda \), it follows from Equations (2.3.51) and (2.3.46), that

\[
\begin{align*}
|VC_p(t^{i+1}_q) - VC_q(t^{i+1}_q)| &= |IC^{i}_p(t^{i+1}_q) - cf(q, \Theta^{i+1}_q)| \\
& \leq \alpha(\delta_S + 2\rho(\tau_{max} + \beta) + 2\Lambda) + \Lambda. \tag{2.3.52}
\end{align*}
\]

Having bounded the skew at \( t^{i+1}_q \), we can bound the skew over the interval \( t^{i+1}_q \leq t < t^{i+1}_p \), by observing that \( t^{i+1}_p - t^{i+1}_q \leq \beta \) by (2.2.12), and applying Equation (2.2.10) to derive the inequality,

\[
|VC_p(t) - VC_q(t)| \leq \alpha(\delta_S + 2\rho(\tau_{max} + \beta) + 2\Lambda) + \Lambda + 2\rho \beta. \tag{2.3.53}
\]
Therefore $\delta$ has to be chosen to satisfy
\[
\alpha(\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) + \Lambda + 2\rho\beta \leq \delta.
\] (2.3.54)

This completes both cases of the proof of Theorem 2.3.2. ■

Theorem 2.3.2 forms the induction step in the proof of the following theorem.

**Theorem 2.3.3** For any two nonfaulty clocks $p, q,$ and $t < t_{p,q}^i$

\[
|VC_p(t) - VC_q(t)| \leq \delta
\] (2.3.55)

**Proof.** The proof is by straightforward induction over $i$. When $i = 0$, the antecedent fails since $t_{p,q}^0 = 0$. The induction hypothesis asserts that for $t < t_{p,q}^i$, the quantity $|VC_p(t) - VC_q(t)|$ does not exceed $\delta$. The induction conclusion requires showing that $\delta$ bounds $|VC_p(t) - VC_q(t)|$ even when $t < t_{p,q}^{i+1}$. We observe that either $t < t_{p,q}^i$, in which case the conclusion follows from the induction hypothesis, or, $t_{p,q}^i \leq t < t_{p,q}^{i+1}$, and the conclusion easily follows from Theorem 2.3.2. ■

One small step remains in the proof of bounded skew from Theorem 2.3.3.

**Theorem 2.3.4** For any $t > 0$ and nonfaulty processors $p$ and $q$, there is an $i$ such that

\[
t < t_{p,q}^i.
\]

**Proof.** By bounded interval, $0 < r_{min} \leq t_{p,j}^{j+1} - t_{p,j}^j$. Thus, $t_{p,j}^{j+1} > jr_{min}$. If we let $i$ be $\lceil t/r_{min} \rceil + 1$, then $t_{p,j}^i > t$. ■

The main result, Theorem 2.1.1, easily follows from the Theorems 2.3.3 and 2.3.4.

We take note of the various conditions on $\delta$ and $\delta_S$:

1. $\pi(2\Lambda + 2\beta \rho, \delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) \leq \delta_S$, by 2.3.40.
2. $\delta_S + 2\rho r_{max} \leq \delta$, by 2.3.45
3. $\alpha(\delta_S + 2\rho(r_{max} + \beta) + 2\Lambda) + \Lambda + 2\rho\beta \leq \delta$, by 2.3.54

This concludes the informal presentation of the proof.

---

6Note that these conditions are significantly different from those derived by Schneider [1] due to various inaccuracies that have been corrected in the mechanical proof.
2.4 ICA as an instance of Schneider’s scheme

The egocentric mean function which is used as a convergence function in the Interactive Convergence Algorithm of Lamport and Melliar-Smith [3] can be shown to satisfy Schneider’s conditions of translation invariance, precision enhancement, and accuracy preservation.

With the interactive convergence algorithm, the convergence function \( cfn_I \) takes the egocentric mean of \( p \)'s estimate of the readings of the \( N \) clocks numbered from 0 to \( N - 1 \), i.e., any readings that are more than \( \Delta \) away from \( p \)'s own reading are replaced by \( p \)'s own reading. This yields the definition

\[
\text{cfn}_I(p, \theta) = \frac{\sum_{i=0}^{N-1} \text{fix}_p(\theta(l))}{N}
\]

(2.4.56)

where

\[
\text{fix}_p(x) = \begin{cases} x & \text{if } |x - \theta(p)| \leq \Delta \\ \theta(p) & \text{otherwise.} \end{cases}
\]

Translation invariance follows from the observation that

\[
\text{fix}_p(\lambda t: \theta(l) + t)(q)) = \text{fix}_p(\theta(q)) + t
\]

(2.4.57)

and

\[
\frac{\sum_{i=0}^{N-1} (\theta(l) + t)}{N} = \frac{\sum_{i=0}^{N-1} \theta(l)}{N} + t
\]

(2.4.58)

To demonstrate precision enhancement, we start with a set of processors \( C \) of cardinality \( |C| \) greater than \( N - F \). Let \( f \) be \( N - |C| \). The hypotheses for precision enhancement are that for any \( l \) and \( m \) in \( C \),

\[
|\gamma(l) - \theta(l)| \leq x
\]

(2.4.59)

\[
|\gamma(l) - \gamma(m)| \leq y
\]

(2.4.60)

\[
|\theta(l) - \theta(m)| \leq y
\]

(2.4.61)

We need to determine \( \pi(x, y) \) so that for any \( p \) and \( q \) in \( C \), we get

\[
|\text{cfn}_I(p, \gamma) - \text{cfn}_I(q, \theta)| \leq \pi(x, y).
\]

(2.4.62)

This difference can be rewritten as

\[
\left| \frac{\sum_{i=0}^{N-1} \text{fix}_p(\gamma(l))}{N} - \frac{\sum_{i=0}^{N-1} \text{fix}_q(\theta(l))}{N} \right|
\]
which is no greater than

\[ \sum_{l=0}^{N-1} \frac{|fix_p(\gamma(l)) - fix_q(\theta(l))|}{N}. \]

This in turn can be rewritten as

\[ \sum_{l \in C} \frac{|fix_p(\gamma(l)) - fix_q(\theta(l))|}{N} + \sum_{l \notin C} \frac{|fix_p(\gamma(l)) - fix_q(\theta(l))|}{N}. \]

Assuming \( y \leq \Delta \) and \( l \in C \), we get \( fix_p(\gamma(l)) \) to be \( \gamma(l) \) and \( fix_q(\theta(l)) \) to be \( \theta(l) \), so that

\[ |fix_p(\gamma(l)) - fix_q(\theta(l))| \leq x \]

and hence,

\[ \sum_{l \in C} \frac{|fix_p(\gamma(l)) - fix_q(\theta(l))|}{N} \leq \frac{(N - f)x}{N}. \]

For \( l \notin C \), the difference

\[ |fix_p(\gamma(l)) - fix_q(\theta(l))| \leq 2\Delta + |\gamma(p) - \theta(q)| \leq 2\Delta + x + y \]

and hence

\[ \sum_{l \notin C} \frac{|fix_p(\gamma(l)) - fix_q(\theta(l))|}{N} \leq \frac{2f\Delta + fx + fy}{N}. \]

We thus get, when \( y \leq \Delta \), that

\[ \pi(x, y) = \frac{(N - f)x}{N} + \frac{2f\Delta + fx + fy}{N}. \quad (2.4.63) \]

In the typical situation when the egocentric mean is computed, the quantity \( x \) representing the reading error is negligible, and \( y \) representing the clock skew is bounded by \( \Delta \). Since the skew following synchronization should be smaller than \( \Delta \), we can see that in Equation (2.4.63), the number of failed processors \( f \) should be below \( N/3 \). Though the derivation of \( \pi(x, y) \) for the case when \( y > \Delta \) is carried out in the machine proof, it is not essential since in practice, \( y \) will not exceed \( \Delta \).

To show that \( cfn_f \) satisfies accuracy preservation, it is sufficient to observe that if all the nonfaulty clocks are within \( x \) of each other, then the nonfaulty clocks can cause the egocentric mean to be at most \( (N - f)x/N \) away from any nonfaulty clock. The faulty clocks can cause the egocentric
mean to be up to $f \times (x + \Delta)/N$ away from a good clock. The total thus yields
\[ \alpha(x) = x + \frac{f \Delta}{N}. \]

The final step is to demonstrate the failure of the monotonicity condition for ICA. The monotonicity condition mentioned at the end of Section 2.2 asserts that if for each processor $l$, $\theta(l) \geq \gamma(l)$, then $\text{cfn}(p, \theta) \geq \text{cfn}(p, \gamma)$. The key reason for the failure of the monotonicity condition is that if some readings in $\gamma$ were ignored because they were more than $\Delta$ below $\gamma(p)$ but were increased in $\theta$ so that they were no longer ignored, then $\text{cfn}(p, \theta)$ could effectively be smaller than $\text{cfn}(p, \gamma)$ even though for every $l$, $\theta(l) \geq \gamma(l)$. More specifically, let $\theta(p) = \gamma(p)$. Observe now that if there is some $l$ such that $\theta(l) + \Delta < \theta(p)$, but with $\gamma(p) > \gamma(l) \geq \gamma(p) - \Delta$, then $\text{fix}_p(\theta(l)) > \text{fix}_p(\gamma(l))$ holds. So, it is possible to have $\text{fix}_p(\theta(l)) > \text{fix}_p(\gamma(l))$, even though we have $\theta(l) < \gamma(l)$.

For the mechanical verification of ICA as an instance of Schneider's protocol, we have verified the constraints, i.e., translation invariance, precision enhancement, and accuracy preservation, hold for the egocentric mean taken as a convergence function. We have not yet instantiated the quantities $r_{\text{min}}, r_{\text{max}}$ and $\beta$, nor verified the conditions of bounded interval, bounded delay and nonoverlap, since these depend on specific implementation choices. It would also be useful to mechanically verify various other Byzantine fault tolerant clock synchronization algorithms to be instances of Schneider's scheme.
Chapter 3

The Verification of Schneider's Protocol using EHDM

The outline in Chapter 2 was adapted from Schneider's description but differs from his presentation in many of the details. The mechanized formalization using EHDM follows the informal description in Chapter 2 fairly closely. We illustrate the highlights of the machine proof below and indicate the correspondence to the informal description. Details regarding the language and capabilities of EHDM are contained in the EHDM tutorial document [2].

3.1 The Clock Assumptions

This section contains the EHDM formalization of the conditions axiomatizing the behavior of clocks. These axioms are contained in a module labeled clockassumptions that is listed in Appendix B starting from page 51. Figure 3.1 contains the type declarations for some of the variables and constants used in clockassumptions. The clockassumptions module makes use of the module arith, which contains the basic arithmetic facts, and countmod, which introduces a counting function. Nonfaultiness is expressed by the predicate correct.

The first few axioms express various minor constraints on the constants as shown in Figure 3.2.

The axioms constraining the physical behavior of the clock appear in Figure 3.3. Since we require the initial skew bound \( \mu \) to not exceed \( \delta_S \),
clockassumptions: Module

Using arith, countmod

Exporting all with countmod, arith

Theory

process: Type is nat
event: Type is nat
time: Type is number
Clocktime: Type is number

l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3: Var process
i, j, k: Var event
x, y, z, r, s, t: Var time
X, Y, Z, R, S, T: Var Clocktime
\gamma, \theta: Var function[process → Clocktime]
\delta, \mu, \rho, r_{\text{min}}, r_{\text{max}}, \beta, \Lambda: number
PC_{s_1}(\ast 2), VC_{s_1}(\ast 2): function[process, time → Clocktime]
t^*_2: function[process, event → time]
\Theta^*_2: function[process, event → function[process → Clocktime]]
IC_{s_1}(\ast 3): function[process, event, time → Clocktime]
correct: function[process, time → bool]
cfn: function[process, function[process → Clocktime] → Clocktime]
\pi: function[Clocktime, Clocktime → Clocktime]
\alpha: function[Clocktime → Clocktime]

Figure 3.1: Declarations from module clockassumptions
\[
\begin{align*}
\text{delta}_0 & : \text{Axiom } \delta \geq 0 \\
\text{mu}_0 & : \text{Axiom } \mu \geq 0 \\
\text{rho}_0 & : \text{Axiom } \rho \geq 0 \\
\text{rho}_1 & : \text{Axiom } \rho < 1 \\
\text{rmin}_0 & : \text{Axiom } r_{\text{min}} > 0 \\
\text{rmax}_0 & : \text{Axiom } r_{\text{max}} > 0 \\
\beta_0 & : \text{Axiom } \beta \geq 0 \\
\lambda_0 & : \text{Axiom } \Lambda \geq 0
\end{align*}
\]

Figure 3.2: Constants in module clockassumptions

The definitions of the virtual clock and the interval clock in terms of the physical clock appear in Figure 3.4. These correspond to (2.1.1), (2.1.4), and (2.1.3), respectively.

The conditions on the convergence function appear in Figure 3.5. The axiom \texttt{Readerror} corresponds to the condition \textit{reading error}. The axiom \texttt{correct\_count} corresponds to \textit{bounded faults}. The remaining correspondences should be self-evident.

Some of the definitions and lemmas from the module \texttt{clockassumptions} have been omitted from this discussion.
init: **Axiom** correct(p, 0) ⊃ PC_p(0) ≥ 0 ∧ PC_p(0) ≤ μ

correct_closed: **Axiom** s ≥ t ∧ correct(p, s) ⊃ correct(p, t)

rate.1: **Axiom** correct(p, s) ∧ s ≥ t ⊃ PC_p(s) - PC_p(t) ≤ (s - t) * (1 + p)

rate.2: **Axiom** correct(p, s) ∧ s ≥ t ⊃ PC_p(s) - PC_p(t) ≥ (s - t) * (1 - p)

rts0: **Axiom** correct(p, t) ∧ t ≤ t_p^{t+1} ⊃ t - t_p^i ≤ r_{max}

rts1: **Axiom** correct(p, t) ∧ t ≥ t_p^{t+1} ⊃ t - t_p^i ≥ r_{min}

rts.0: **Lemma** correct(p, t_p^{t+1}) ⊃ t_p^{t+1} - t_p^i ≤ r_{max}

rts.1: **Lemma** correct(p, t_p^{t+1}) ⊃ t_p^{t+1} - t_p^i ≥ r_{min}

rts2: **Axiom** correct(p, t) ∧ t ≥ t_p^i + β ∧ correct(q, t) ⊃ t ≥ t_p^i

rts.2: **Axiom** correct(p, t_p^i) ∧ correct(q, t_p^i) ⊃ t_p^i - t_q^i ≤ β

synctime.0: **Axiom** t^0_p = 0

Figure 3.3: Physical clock axioms in module clockassumptions

VClock_defn: **Axiom**
correct(p, t) ∧ t ≥ t_p^i ∧ t < t_p^{t+1} ⊃ VC_p(t) = IC_p^i(t)

Adj: function[process, event ∆ Clocktime] =
( λ p, i: ( if i > 0 then cfn(p, t_p^i) - PC_p(t_p^i) else 0 end if))

IClock_defn: **Axiom** correct(p, t) ⊃ IC_p^i(t) = PC_p(t) + Adj(p, i)

Figure 3.4: Clock definitions in module clockassumptions

28
Readerer: **Axiom** \( \text{correct}(p, t_{p}^{t+1}) \land \text{correct}(q, t_{p}^{t+1}) \) 
\( \supset |\Theta_{p}^{t+1}(q) - IC_{t}^{t+1}(q)| \leq \Lambda \)

**translation invariance:** **Axiom**
\( X \geq 0 \supset \text{cfn}(p, (\lambda p_{1} \rightarrow \text{Clocktime}: \gamma(p_{1}) + X)) = \text{cfn}(p, \gamma) + X \)

\( \text{ppred}: \text{Var function}[\text{process} \rightarrow \text{bool}] \)
\( \text{maxfaults}: \text{process} \)
\( \text{okay}_1\text{Readpred}: \text{function}[\text{function}[\text{process} \rightarrow \text{Clocktime}], \text{Clocktime}, \text{function}[\text{process} \rightarrow \text{bool}] \rightarrow \text{bool}] = \)
\( (\lambda \gamma, \theta, \text{ppred}: (\forall l, m: \text{ppred}(l) \land \text{ppred}(m) \supset |\gamma(l) - \gamma(m)| \leq Y)) \)
\( \text{okay}_1\text{pairs}: \text{function}[\text{function}[\text{process} \rightarrow \text{Clocktime}], \text{Clocktime}, \text{function}[\text{process} \rightarrow \text{bool}] \rightarrow \text{bool}] = \)
\( (\lambda \gamma, \theta, \text{ppred}: (\forall p_{z}: \text{ppred}(p_{z}) \supset |\gamma(p_{z}) - \theta(p_{z})| \leq X)) \)
\( N: \text{process} \)

**N.0: Axiom** \( N > 0 \)

**N.maxfaults: Axiom** \( \text{maxfaults} \leq N \)

**precision_enhancement_ax:** **Axiom**
\( \text{count}(\text{ppred}, N) \geq N - \text{maxfaults} \land \text{okay}_1\text{Readpred}(\gamma, Y, \text{ppred}) \land \text{okay}_1\text{Readpred}(\theta, Y, \text{ppred}) \land \text{okay}_1\text{pairs}(\gamma, \theta, X, \text{ppred}) \land \text{ppred}(p) \land \text{ppred}(q) \supset |\text{cfn}(p, \gamma) - \text{cfn}(q, \theta)| \leq \pi(X, Y) \)

**correct_count:** **Axiom** \( \text{count}((\lambda p: \text{correct}(p, t)), N) \geq N - \text{maxfaults} \)

**accuracy_preservation_ax:** **Axiom**
\( \text{okay}_1\text{Readpred}(\gamma, X, \text{ppred}) \land \text{count}(\text{ppred}, N) \geq N - \text{maxfaults} \land \text{ppred}(p) \land \text{ppred}(q) \supset |\text{cfn}(p, \gamma) - \gamma(q)| \leq \alpha(X) \)

**Figure 3.5:** Conditions on Logical Clocks in module clockassumptions
agreement: Lemma $\beta \leq r_{\min}$
\[
\wedge \mu \leq \delta \wedge \pi(2 \cdot \Lambda + 2 \cdot \beta \cdot \rho, \delta_s + 2 \cdot (r_{\max} + \beta) \cdot \rho + \Lambda) \leq \delta_s
\wedge \delta_s + 2 \cdot r_{\max} \cdot \rho \leq \delta
\wedge \alpha(\delta_s + 2 \cdot (r_{\max} + \beta) \cdot \rho + 2 \cdot \Lambda) + \Lambda + 2 \cdot \beta \cdot \rho \leq \delta
\wedge t \geq 0 \wedge \text{correct}(p, t) \land \text{correct}(q, t)
\supset |VC_p(t) - VC_q(t)| \leq \delta
\]

Figure 3.6: Main Theorem in module lemma_final

okaymaxsync: function[nat, Clocktime \rightarrow bool] =
(\lambda i, X:\{\forall p, q:\}
\text{correct}(p, t^i_{p, q}) \land \text{correct}(q, t^i_{q, p})
\supset |IC_p^i(t^i_{p, q}) - IC_q^i(t^i_{q, p})| \leq X))

lemma 2: Lemma $\beta \leq r_{\min}$
\[
\wedge \mu \leq X \wedge \pi(2 \cdot \Lambda + 2 \cdot \beta \cdot \rho, X + 2 \cdot ((r_{\max} + \beta) \cdot \rho + \Lambda)) \leq X
\supset \text{okaymaxsync}(i, X)
\]

Figure 3.7: Skew immediately following resynchronization from module readbounds

3.2 The Proof Highlights

The conclusion corresponding to Theorem 2.1.1 is the theorem agreement that appears in the module lemma_final listed at page 79 of Appendix B. This theorem is displayed in Figure 3.6. It should be compared to the statement of Theorem 2.1.1 (page 8) and to the conditions at the end of Section 2.3.2 (page 21). The axioms, definitions, and lemmas used, whether in a direct or indirect manner, in the proof of agreement are analyzed in Appendix C.1 to ensure that all proof obligations have been discharged. Both the process and the result of checking these dependencies are part of what is termed the proof chain analysis.

The verified version of Theorem 2.3.1 is given in Figure 3.7 extracted from the module readbounds listed at page 63 of Appendix B.

The verified version of Theorem 2.3.2 appears in Figure 3.8 which is taken from the module lemma3 listed at page B of Appendix B. The expression $t^i_{\lfloor p \geq q \rfloor}(i)$ is an alternative notation for $t^i_{p, q}$ since $(p \uparrow q)[i]$ represents $p$ if $t^i_p \geq t^i_q$. 
okayClocks: function[process, process, nat → bool] =
(λ p, q, i: (∀ t:
  t ≥ 0 ∧ t < t^i_pφq_i[t] ∧ correct(p, t) ∧ correct(q, t)
  ⊃ |VC^p_p(t) - VC^q_q(t)| ≤ δ))

lemma3.3: Lemma β ≤ r_{min}
∧ μ ≤ δ_S ∧ π(2 * Λ + 2 * β * ρ, δ_S + 2 * ((r_{max} + β) * ρ + Λ)) ≤ δ_S
∧ δ_S + 2 * r_{max} * ρ ≤ δ
∧ α(δ_S + 2 * (r_{max} + β) * ρ + 2 * Λ) + Λ + 2 * β * ρ ≤ δ
⊂ okayClocks(p, q, i)

Figure 3.8: Skew up to ith resynchronization from module lemma3

and q otherwise.

The EHDM definition of the egocentric mean function is given by icalg in Figure 3.9.

The verification of the translation invariance, precision enhancement, and accuracy preservation properties of the egocentric mean function is presented in Figure 3.10. The proof chain analyses for these theorems appear in Appendices C.2, C.3, and C.4.
Figure 3.9: Egocentric mean from module ica
ica_translation_invariance: Lemma
\[ N > 0 \supset \text{icalg}(p, (\lambda q: \text{fun}(q) + X), Y) = \text{icalg}(p, \text{fun}, Y) + X \]

icalg_precision_enhancement: Lemma
\[
\begin{align*}
\text{ppred}(p) \land \text{ppred}(q) \\
\land \text{count}(\text{ppred}, N) \geq N - \text{maxfaults} \\
\land \text{okay_pairs}(%n1, %n2, X, \text{ppred}) \\
\land \text{okay_Readpred}(%n1, Z, \text{ppred}) \land \text{okay_Readpred}(%n2, Z, \text{ppred}) \\
\supset \text{icalg}(p, \text{fun1}, \Delta) - \text{icalg}(q, \text{fun2}, \Delta) \leq \text{icalg}_{\Pi}(X, Z)
\end{align*}
\]

icalg_accuracy_preservation: Lemma
\[
\begin{align*}
\text{ppred}(p) \land \text{ppred}(q) \\
\land \text{count}(\text{ppred}, N) \geq N - \text{maxfaults} \land \text{okay_Readpred}(\text{fun}, X, \text{ppred}) \\
\supset |\text{icalg}(p, \text{fun}, \Delta) - \text{fun}(q)| \\
\leq ((N - \text{maxfaults}) \ast X + \text{maxfaults} \ast (X + \Delta))/N
\end{align*}
\]

Figure 3.10: Properties of egocentric mean from modules ica, ica3, and ica4
Chapter 4

Conclusions

Rigorously proving the correctness of distributed protocols is an extremely difficult task, with or without mechanical assistance. Fault-tolerant clock synchronization is an excellent example of a problem where the algorithms, though often simple, are not at all easily verified. In such cases, it is extremely important to have certain organizing principles which capture the common features of the various protocols with convincing generality. Schneider's schema for Byzantine clock synchronization provides such principles to unify the presentation and proofs of a number of different protocols. Schneider starts with certain axioms constraining the behaviors of clocks, the selection of synchronization times, and the convergence functions. He uses these constraints to derive a bound on the skew between any two nonfaulty clocks. It is worth noting for the discussion below that Schneider's work is described in an unpublished technical report that has not had the benefit of widespread examination.

The formalization here revises a few details from Schneider's presentation. Schneider's notion of a global signal to trigger resynchronization has been dropped because such a notion is difficult to instantiate for many protocols. Though the quantities $r_{\text{max}}$ and $r_{\text{min}}$ have a different meaning from Schneider's, these differences ought not to matter in any of the bounds derived. For instance, $r_{\text{max}}$ here bounds $t^{i+1}_p - t^i_p$, but Schneider's bound on this quantity would be $r_{\text{max}} + \beta$. However, the significant quantity in the proof is the difference $t^{i+1}_p - t^i_q$ and the bound on this quantity is $r_{\text{max}} + \beta$ in either formalization. In other words, Schneider's bounds on $\delta$ and $\delta_S$ ought to have been the same as those derived in Section 2.3.2, but there were certain minor errors of algebra in his proofs and some latitude in his
argument. The derivation we present is extremely tight, given the structure of the proof. Schneider's monotonicity condition is avoided in the proofs here. This condition is used heavily by Schneider in his arguments, but it actually turns out to be false for many protocols. The statement of accuracy preservation is also slightly different here from that of Schneider. Schneider also presents the proof for the case of continuous resynchronization which is not handled here.

The initial proof using EHD M took about a month. The proof has been considerably revised and improved since that first effort. Verifying that the egocentric mean function of ICA satisfied the conditions of translation invariance, accuracy preservation, and precision enhancement, took about two weeks. The EHD M modules are listed in Appendix B. The proof involves 182 theorems or lemmas. A rerun of the entire proof on a SUN 3/470 takes 3227 CPU seconds (see Appendix A).

An early difficulty in the verification attempt was in arriving at a satisfactory formalization that suitably revised the one from Schneider. The proper treatment of failure proved to be a pervasive and important difficulty. Unlike other similar informal and machine-verified proofs, our formalization was careful to permit processors to fail at any time. Rushby and von Henke [8], for example, regard processors as nonfaulty in an interval between synchronizations only if they have been nonfaulty for the entire interval. This is an adequate model for most practical purposes but it is less general because it does not distinguish between processors that may have failed at the beginning of the interval and those that failed at the very end of an interval. An even coarser model, and the one unwittingly used in most informal presentations of clock synchronization, is one where the only correct processors are those that never fail. In some sense, this is acceptable since often the only significant requirement is that a sufficient number of processors be nonfaulty at any given time. However, such a formalization allows no conclusion to be drawn regarding a processor which has yet to fail but does eventually fail, since it is regarded as always having been faulty.

To illustrate the circularity lurking in the formalization of time and failure, consider the following seemingly natural formalization of nonfaultiness in an interval. Suppose that a processor is described as nonfaulty for an interval if it functions normally through the end of the interval. Let the end of the interval be the time at which the nonfaulty clocks indicate a certain reading or have performed a certain operation such as resetting their readings. Suppose, for example, that the end of the interval is given by the time t when the slowest of the "nonfaulty" clocks p reads T. Now suppose
that \( p \) fails exactly at \( t \). Then clearly the end of the interval is earlier than \( t \), but at any point earlier than \( t \), processor \( p \) is nonfaulty and has yet to read \( T \). This "natural" definition of the end of an interval thus yields a contradiction. Many similar problems arose frequently in attempting to set down the clock axioms. The most natural statement of these axioms often turned out to be either wrong or too restrictive. It is also important to observe that these problems would never have been noticed in most informal presentations since these details, though important, would have been largely ignored.

The most useful features of \( \text{EHDM} \) for this verification were the decision procedures for linear integer and real inequalities and equalities. The informal proof is of course replete with long chains of inequality reasoning, and the decision procedures handled those steps in a fairly mechanical manner. The higher-order features of the language were also used to formalize the conditions of \textit{translation invariance}, \textit{precision enhancement}, and \textit{accuracy preservation}, but these were not essential. These could have also been formalized in terms of lists or finite arrays. The language of \( \text{EHDM} \) underwent a number of improvements during this project, and not all of these improvements have been exploited in this proof. The use of predicate subtypes would have permitted the introduction of types corresponding to the non-negative and the positive numbers.

Fault-tolerant distributed protocols are sufficiently delicate to warrant careful, formal, mechanized analysis. Schneider's presentation of Byzantine fault-tolerant clock synchronization protocols provides a valuable mathematical framework for such an analysis. The machine-checked proof of Schneider's protocol led to a more precise formulation of the protocol and a more closely reasoned proof. It is inconceivable that the same degree of logical rigor and accuracy could be achieved without computational assistance.
References


Appendix A

Proof Summary

The proof summary is the result of executing a command to attempt to prove all the proof declarations in the context. The only failures are in the automatically generated proof declarations for the type correctness conditions (tcc). The time given below is the running time on a SUN 3/470.

Proof summaries for modules on using chain of module top

<table>
<thead>
<tr>
<th>Module</th>
<th>Successful proofs</th>
<th>Failures</th>
<th>Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>division_tcc</td>
<td>7</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>tcc_proofs_tcc</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>ica3_tcc</td>
<td>0</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>ica4_tcc</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>ica_tcc</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>lemma_final_tcc</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>countmod_tcc</td>
<td>3</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>tcc_proofs</td>
<td>14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ica3</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ica2</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ica</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>ica4</td>
<td>8</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>basics</td>
<td>25</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>readbounds</td>
<td>12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lemma3</td>
<td>24</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>countmod</td>
<td>no proofs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>clockassumptions</td>
<td>9</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lemma_final</td>
<td>5</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>absmod</td>
<td>15</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>division</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Module multiplication: 11 successful proofs, 0 failures, 0 errors
Module arith: no proofs
Module top: 1 successful proof, 0 failures, 0 errors

Totals: 182 successful proofs, 15 failures, 0 errors

Total time: 3227 seconds.
Appendix B

The Complete EHDM Proof

Note that the modules ending with _tcc are automatically generated during type checking. The proofs declared in these modules may not succeed, but all the automatically generated theorems have been proved as illustrated by the completeness of the proof chain analyses in Appendix C.

multiplication: Module

Exporting all

Theory

\[ x, y, z, x_1, y_1, z_1, x_2, y_2, z_2: \text{Var number} \]

\[ \times 1 \times 2: \text{function[number, number -> number]} = (\lambda x, y: (x \times y)) \]

mult_distrib: Lemma \[ x \times (y + z) = x \times y + x \times z \]

mult_distrib_minus: Lemma \[ x \times (y - z) = x \times y - x \times z \]

mult_rident: Lemma \[ x \times 1 = x \]

mult_lident: Lemma \[ 1 \times z = z \]

distrib: Lemma \[ (x + y) \times z = x \times z + y \times z \]

distrib_minus: Lemma \[ (x - y) \times z = x \times z - y \times z \]

mult_non_neg: Axiom \[ ((x \geq 0 \land y \geq 0) \lor (x \leq 0 \land y \leq 0)) \Rightarrow x \times y \geq 0 \]

mult_pos: Axiom \[ ((x > 0 \land y > 0) \lor (x < 0 \land y < 0)) \Rightarrow x \times y > 0 \]

mult_com: Lemma \[ x \times y = y \times x \]

pos_product: Lemma \[ x \geq 0 \land y \geq 0 \Rightarrow x \times y \geq 0 \]
mult.leq: Lemma \( z \geq 0 \land x \geq y \supset x \cdot z \geq y \cdot z \)

mult.leq.2: Lemma \( z \geq 0 \land x \geq y \supset z \cdot x \geq z \cdot y \)

mult.0: Axiom \( 0 \cdot x = 0 \)

mult.gt: Lemma \( z > 0 \land x > y \supset x \cdot z > y \cdot z \)

Proof

mult.gt.pr: Prove mult.gt from
\mult.pos \{x \leftarrow x - y, y \leftarrow z\}, distrib.minus

distrib.minus.pr: Prove distrib.minus from
\mult.l.distrib.minus \{x \leftarrow z, y \leftarrow x, z \leftarrow y\},
\mult.com \{x \leftarrow x - y, y \leftarrow z\},
\mult.com \{y \leftarrow z\},
\mult.com \{z \leftarrow y, y \leftarrow z\}

mult.leq.2.pr: Prove mult.leq.2 from
\mult.l.distrib.minus \{x \leftarrow z, y \leftarrow x, z \leftarrow y\},
\mult.non.neg \{z \leftarrow z, y \leftarrow x - y\}

mult.leq.pr: Prove mult.leq from
\mult.com.minus, \mult.non.neg \{x \leftarrow x - y, y \leftarrow z\}

distrib.leq.pr: Prove mult.com from \(*1 \leftrightarrow 2, \ast1 \leftrightarrow 2 \{x \leftarrow y, y \leftarrow x\}

pos.product.pr: Prove pos.product from \mult.non.neg

mult_rident.proof: Prove mult.rident from \(*1 \leftrightarrow 2 \{y \leftarrow 1\}

mult_lident.proof: Prove mult.lident from \(*1 \leftrightarrow 2 \{z \leftarrow 1, y \leftarrow z\}

distrib.proof: Prove distrib from
\ast1 \leftrightarrow 2 \{x \leftarrow x + y, y \leftarrow z\},
\ast1 \leftrightarrow 2 \{y \leftarrow z\},
\ast1 \leftrightarrow 2 \{z \leftarrow y, y \leftarrow z\}

mult.l.distrib.proof: Prove mult.l.distrib from
\ast1 \leftrightarrow 2 \{y \leftrightarrow y + z, x \leftarrow x\}, \ast1 \leftrightarrow 2, \ast1 \leftrightarrow 2 \{y \leftarrow z\}

mult.l.distrib.minus.proof: Prove mult.l.distrib.minus from
\ast1 \leftrightarrow 2 \{y \leftarrow y - z, x \leftarrow x\}, \ast1 \leftrightarrow 2, \ast1 \leftrightarrow 2 \{y \leftarrow z\}

End multiplication
absmod: Module

Using multiplication

Exporting all

Theory

$x, y, z, x_1, y_1, z_1, x_2, y_2, z_2$: Var number

| * 1|: Definition function[number → number] =
  \( (\lambda x:\text{ if } x < 0 \text{ then } -x \text{ else } x \text{ end if}) \)

abs_main: Lemma \(|x| < z \supset (x < z \lor -x < z)\)

abs_leq.0: Lemma \(|x - y| \leq z \supset (x - y) \leq z\)

abs_diff: Lemma \(|x - y| < z \supset ((x - y) < z \lor (y - x) < z)\)

abs_leq: Lemma \(|x| \leq z \supset (x \leq z \lor -x \leq z)\)

abs_bnd: Lemma \(0 \leq z \land 0 \leq z \land z \leq y \land y \leq z \supset |x - y| \leq z\)

abs_leq.1: Lemma \(|x - y| \leq z \supset x \leq y + z\)

abs_leq.2: Lemma \(|x - y| \leq z \supset x \geq y - z\)

abs_leq.3: Lemma \(x \leq y + z \land x \geq y - z \supset |x - y| \leq z\)

abs_drift: Lemma \(|x - y| \leq z \land |x_1 - z| \leq z_1 \supset |x_1 - y| \leq z + z_1\)

abs_com: Lemma \(|x - y| = |y - x|\)

abs_drift.2: Lemma
  \(|x - y| \leq z \land |x_1 - x| \leq z_1 \land |y_1 - y| \leq z_2 \supset |x_1 - y_1| \leq z + z_1 + z_2\)

abs_geq: Lemma \(x \geq y \land y \geq 0 \supset |x| \geq |y|\)

abs_geq.0: Lemma \(z \geq 0 \supset |z| = z\)

abs_plus: Lemma \(|x + y| \leq |x| + |y|\)

abs_diff.3: Lemma \(x - y \leq z \land y - x \leq z \supset |x - y| \leq z\)

Proof

abs_plus.pr: Prove abs_plus from \(|x| \{x - x + y\}, |x| \{x - y\}\)

abs_diff.3.pr: Prove abs_diff.3 from \(|x| \{x - x - y\}\)

abs_geq.0.proof: Prove abs_geq.0 from \(|x| \{x - y\}\)
abs.geq.proof: Prove \textit{abs.geq} from $| \star 1|$, $| \star 1| \{z \leftarrow y\}$

abs.drift.2.proof: Prove \textit{abs.drift.2} from
abs.drift,
\textit{abs.drift} \{z \leftarrow y, y \leftarrow y_1, z \leftarrow z_2, z_1 \leftarrow z + z_1\},
\textit{abs.com} \{z \leftarrow y_1\}

abs.com.proof: Prove \textit{abs.com} from $| \star 1| \{z \leftarrow (x - y)\}, | \star 1| \{z \leftarrow (y - x)\}$

abs.drift.proof: Prove \textit{abs.drift} from
\textit{abs.1.bnd},
\textit{abs.1.bnd} \{z \leftarrow x_1, y \leftarrow x, z \leftarrow z_1\},
\textit{abs.2.bnd},
\textit{abs.2.bnd} \{z \leftarrow x_1, y \leftarrow x, z \leftarrow z_1\},
\textit{abs.3.bnd} \{z \leftarrow x_1, z \leftarrow z + z_1\}

abs.3.bnd.proof: Prove \textit{abs.3.bnd} from $| \star 1| \{z \leftarrow (x - y)\}$

abs.main.proof: Prove \textit{abs.main} from $| \star 1|$

abs.leq.0.proof: Prove \textit{abs.leq.0} from $| \star 1| \{z \leftarrow x - y\}$

abs.diff.proof: Prove \textit{abs.diff} from $| \star 1| \{z \leftarrow (x - y)\}$

abs.leq.proof: Prove \textit{abs.leq} from $| \star 1|$

abs.bnd.proof: Prove \textit{abs.bnd} from $| \star 1| \{z \leftarrow (x - y)\}$

abs.1.bnd.proof: Prove \textit{abs.1.bnd} from $| \star 1| \{z \leftarrow (x - y)\}$

abs.2.bnd.proof: Prove \textit{abs.2.bnd} from $| \star 1| \{z \leftarrow (x - y)\}$

\textbf{End abs.mod}
division: Module

Using multiplication, absmod

Exporting all

Theory

\[ x, y, z, x_1, y_1, z_1, x_2, y_2, z_2: \text{Var number} \]

[\text{ceil_defn: Axiom } [x] \geq x \land [x] - 1 < x]

[\text{mult_div_1: Axiom } z \neq 0 \supset x \cdot y/z = x \cdot (y/z)]

[\text{mult_div_2: Axiom } z \neq 0 \supset x \cdot y/z = (x/z) \cdot y]

[\text{mult_div_3: Axiom } z \neq 0 \supset (z/z) = 1]

[\text{mult_div: Lemma } y \neq 0 \supset (x/y) \cdot y = x]

[\text{div_cancel: Lemma } x \neq 0 \supset x \cdot y/z = y]

[\text{div_distrib: Lemma } z \neq 0 \supset ((x + y)/z) = (x/z) + (y/z)]

[\text{ceil_mult_div: Lemma } y > 0 \supset [x/y] \cdot y \geq x]

[\text{ceil_plus_mult_div: Lemma } y > 0 \supset [x/y] + 1 \cdot y > x]

[\text{div_nonnegative: Lemma } z \geq 0 \land y > 0 \supset (x/y) \geq 0]

[\text{div_minus_distrib: Lemma } z \neq 0 \supset (z + y)/z = (x/z) - (y/z)]

[\text{div_ineq: Lemma } z > 0 \land x \leq y \supset (x/z) \leq (y/z)]

[\text{abs_div: Lemma } y > 0 \supset |x/y| = |x|/y]

[\text{mult_minus: Lemma } y \neq 0 \supset -(x/y) = (-x)/y]

[\text{div_minus_l: Lemma } y > 0 \land x < 0 \supset (x/y) < 0]

Proof

\text{div_nonnegative_pr: Prove div_nonnegative from}

\text{mult_non_neg \{x \leftarrow (if y \neq 0 then (x/y) else 0 end if), mult_div}
div_distrib_pr: Prove div_distrib from
mult_div.1 {x ← x + y, y ← 1, z ← z},
mult_rident {x ← x + y},
mult_div.1 {x ← x, y ← 1, z ← z},
mult_rident,
mult_div.1 {x ← y, y ← 1, z ← z},
mult_rident {x ← y},
distrib {z ← (if z ≠ 0 then (1/z) else 0 end if)}

div_cancel_pr: Prove div_cancel from
mult_div.2 {z ← z}, mult_div.3 {z ← z}, mult_rident {z ← y}

mult_div_pr: Prove mult_div from
mult_div.2 {z ← y}, mult_div.1 {z ← y}, mult_div.3 {z ← y}, mult_rident

abs_div_pr: Prove abs_div from
|*1| {x ← (if y ≠ 0 then (x/y) else 0 end if)},
|*1| ,
div_nonnegative,
div_minus.1,
mult_minus

mult_minus_pr: Prove mult_minus from
mult_div.1 {x ← -1, y ← x, z ← y},
|*1 *|2 {x ← -1, y ← z},
|*1 *|2 {x ← -1, y ← (if y ≠ 0 then (x/y) else 1 end if)}

div_minus.1.pr: Prove div_minus.1 from
mult_div,
pos_product {x ← (if y ≠ 0 then (x/y) else 0 end if), y ← y}

div_minus.distrib_pr: Prove div_minus.distrib from
div_distrib {y ← -y}, mult_minus {x ← y, y ← z}

div.ineq_pr: Prove div.ineq from
mult_div {y ← z},
mult_div {x ← y, y ← z},
mult_gt
{x ← (if z ≠ 0 then (x/z) else 0 end if),
y ← (if z ≠ 0 then (y/z) else 0 end if)}
ceil_plus_mult_div_proof: Prove ceil_plus_mult_div from
ceil_mult_div,
distrib
\{ x \leftarrow [( \text{if } y \neq 0 \text{ then } (x/y) \text{ else } 0 \text{ end if})], 
\quad y \leftarrow 1, 
\quad z \leftarrow y \},
mult_lident \{ x \leftarrow y \} 

ceil_mult_div_proof: Prove ceil_mult_div from
mult_div,
mult_leq
\{ x \leftarrow [( \text{if } y \neq 0 \text{ then } (x/y) \text{ else } 0 \text{ end if})], 
\quad y \leftarrow ( \text{if } y \neq 0 \text{ then } (x/y) \text{ else } 0 \text{ end if}), 
\quad z \leftarrow y \},
ceil_defn \{ x \leftarrow ( \text{if } y \neq 0 \text{ then } (x/y) \text{ else } 0 \text{ end if}) \} 

End division
division_tcc: Module

Using division

Exporting all with division

Theory

\( x: \text{Var number} \)
\( y: \text{Var number} \)
\( z: \text{Var number} \)

mult_div_l_TCC1: Formula \((z \neq 0) \supset (z \neq 0)\)

mult_div_TCC1: Formula \((y \neq 0) \supset (y \neq 0)\)

div_cancel_TCC1: Formula \((x \neq 0) \supset (x \neq 0)\)

ceil_mult_div_TCC1: Formula \((y > 0) \supset (y \neq 0)\)

div_nonnegative_TCC1: Formula \((x \geq 0 \land y > 0) \supset (y \neq 0)\)

div_ineq_TCC1: Formula \((z > 0 \land z \leq y) \supset (z \neq 0)\)

div_minus_l_TCC1: Formula \((y > 0 \land z < 0) \supset (y \neq 0)\)

Proof

mult_div_l_TCC1_PROOF: Prove mult_div_l_TCC1

mult_div_TCC1_PROOF: Prove mult_div_TCC1

div_cancel_TCC1_PROOF: Prove div_cancel_TCC1

ceil_mult_div_TCC1_PROOF: Prove ceil_mult_div_TCC1

div_nonnegative_TCC1_PROOF: Prove div_nonnegative_TCC1

div_ineq_TCC1_PROOF: Prove div_ineq_TCC1

div_minus_l_TCC1_PROOF: Prove div_minus_l_TCC1

End division_tcc
arith: Module

Using multiplication, division, absmod

Exporting all with multiplication, division, absmod

End arith
countmod: Module

Exporting all

Theory

\[ l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3: \text{Var nat} \]
\[ i, j, k: \text{Var nat} \]
\[ x, y, z, r, s, t: \text{Var number} \]
\[ X, Y, Z: \text{Var number} \]
\[ \text{ppred, ppred1, ppred2: Var function[nat --- boool]} \]
\[ \text{fun, fun1, fun2: Var function[nat --- number]} \]
\[ \text{countsize: function[function[nat --- bool], nat --- nat]} = (\lambda \text{ppred, i: i}) \]
\[ \text{count: Recursive function[function[nat --- bool], nat --- nat]} = \]
\[ (\lambda \text{ppred, i: (if i > 0}} \]
\[ \text{then (if ppred(i - 1)}} \]
\[ \text{then 1 + (count(ppred, i - 1))} \]
\[ \text{else count(ppred, i - 1))} \]
\[ \text{end if)} \]
\[ \text{else 0} \]
\[ \text{end if)) by countsize} \]

End countmod
countmod_tcc: Module

Using countmod

Exporting all with countmod

Theory

i: Var naturalnumber
ppred: Var function[naturalnumber -> boolean]

count_TCC1: Formula (i > 0) C (i - 1 \geq 0)
count_TCC2: Formula (ppred(i - 1)) A (i > 0) C (i - 1 \geq 0)
count_TCC3: Formula (¬(ppred(i - 1))) A (i > 0) C (i - 1 \geq 0)
count_TCC4: Formula (ppred(i - 1)) A (i > 0) C countsize(ppred, i) > countsize(ppred, i - 1)
count_TCC5: Formula (¬(ppred(i - 1))) A (i > 0) C countsize(ppred, i) > countsize(ppred, i - 1)

Proof

count_TCC1_PROOF: Prove count.TCC1
count_TCC2_PROOF: Prove count.TCC2
count_TCC3_PROOF: Prove count.TCC3
count_TCC4_PROOF: Prove count.TCC4
count_TCC5_PROOF: Prove count.TCC5

End countmod_tcc
clockassumptions: Module

Using arith, countmod

Exporting all with countmod, arith

Theory

process: Type is nat

event: Type is nat

time: Type is number

Clocktime: Type is number

\(l, m, n, p, q, p_1, p_2, q_1, q_2, p_3, q_3\): Var process

\(i, j, k\): Var event

\(x, y, z, r, s, t\): Var time

\(X, Y, Z, R, S, T\): Var Clocktime

\(\gamma, \theta\): Var function[process \(\rightarrow\) Clocktime]

\(\delta, \mu, \rho, r_{min}, r_{max}, \beta, \Lambda\): number

\(PC_{\ast 1}(\ast 2), VC_{\ast 1}(\ast 2)\): function[process, time \(\rightarrow\) Clocktime]

\(t_{\ast 2}^{\ast 1}\): function[process, event \(\rightarrow\) time]

\(\Theta_{\ast 2}^{\ast 1}\): function[process, event \(\rightarrow\) function[process \(\rightarrow\) Clocktime]]

\(IC_{\ast 2}^{\ast 1}(\ast 3)\): function[process, event, time \(\rightarrow\) Clocktime]

correct: function[process, time \(\rightarrow\) bool]

cfn: function[process, function[process \(\rightarrow\) Clocktime] \(\rightarrow\) Clocktime]

\(\pi\): function[Clocktime, Clocktime \(\rightarrow\) Clocktime]

\(\alpha\): function[Clocktime \(\rightarrow\) Clocktime]

\(\text{delta.0: Axiom } \delta \geq 0\)

\(\text{mu.0: Axiom } \mu \geq 0\)

\(\text{rho.0: Axiom } \rho \geq 0\)

\(\text{rho.1: Axiom } \rho < 1\)

\(\text{rmin.0: Axiom } r_{min} > 0\)

\(\text{rmax.0: Axiom } r_{max} > 0\)

\(\text{beta.0: Axiom } \beta \geq 0\)

\(\text{lamb.0: Axiom } \Lambda \geq 0\)

\(\text{init: Axiom } \text{correct}(p, 0) \supset PC_p(0) \geq 0 \land PC_p(0) \leq \mu\)

\(\text{correct_closed: Axiom } s \geq t \land \text{correct}(p, s) \supset \text{correct}(p, t)\)

\(\text{rate.1: Axiom } \text{correct}(p, s) \land s \geq t \supset PC_p(s) - PC_p(t) \leq (s - t) \times (1 + \rho)\)
rate_2: **Axiom** \( \text{correct}(p, s) \land s \geq t \supset PC_p(s) - PC_p(t) \geq (s - t) \cdot (1 - \rho) \)

\( \text{rts0: Axiom} \text{ correct}(p, t) \land t \leq t_p^{t+1} \supset t - t_p^{t} \leq r_{\max} \)

\( \text{rts1: Axiom} \text{ correct}(p, t) \land t \geq t_p^{t+1} \supset t - t_p^{t} \geq r_{\min} \)

\( \text{rts0: Lemma} \text{ correct}(p, t_p^{t+1}) \supset t_p^{t+1} - t_p^{t} \leq r_{\max} \)

\( \text{rts1: Lemma} \text{ correct}(p, t_p^{t+1}) \supset t_p^{t+1} - t_p^{t} \geq r_{\min} \)

\( \text{rts2: Axiom} \text{ correct}(p, t) \land t \geq t_p^{t} + \beta \land \text{correct}(q, t) \supset t \geq t_p^{q} \)

\( \text{rts2: Axiom} \text{ correct}(p, t_p^{t}) \land \text{correct}(q, t_p^{q}) \supset t_p^{t} - t_p^{q} \leq \beta \)

\( \text{synctime.0: Axiom} \ t_p^{0} = 0 \)

\( \text{VClock_defn: Axiom} \) \( \text{correct}(p, t) \land t < t_p^{t+1} \supset VC_p(t) = IC_p^t(t) \)

\( \text{Adj: function}\[\text{process, event \rightarrow Clocktime}] = \\
(\lambda p, i: \text{if } i > 0 \text{ then } cfn(p, \Theta_p^t) - PC_p(t_p^{t}) \text{ else } 0 \text{ end if}) \)

\( \text{IClock_defn: Axiom} \text{ correct}(p, t) \supset IC_p^t(t) = PC_p(t) + \text{Adj}(p, i) \)

\( \text{Readerror: Axiom} \text{ correct}(p, t) \supset IC_p^t(t) = PC_p(t) + \text{Adj}(p, i) \)

\( \text{translation_invariance: Axiom} \)

\( X \geq 0 \supset cfn(p, (\lambda p_1 \rightarrow \text{Clocktime}: \gamma(p_1) + X)) = cfn(p, \gamma) + X \)

\( \text{ppred: Var function}\[\text{process \rightarrow bool}] \)

\( \text{maxfaults: process} \)

\( \text{okay_Readpred: function}\[\text{function}\[\text{process \rightarrow Clocktime}], \text{Clocktime}, \text{function}\[\text{process \rightarrow bool}] \rightarrow \text{bool}] = \\
(\lambda \gamma, Y, \text{ppred}: (\forall l, m: \text{ppred}(l) \land \text{ppred}(m) \supset |\gamma(l) - \gamma(m)| \leq Y)) \)

\( \text{okay_pairs: function}\[\text{function}\[\text{process \rightarrow Clocktime}], \text{Clocktime}, \text{function}\[\text{process \rightarrow bool}] \rightarrow \text{bool}] = \\
(\lambda \gamma, \theta, X, \text{ppred}: (\forall p_3: \text{ppred}(p_3) \supset |\gamma(p_3) - \theta(p_3)| \leq X)) \)

\( N: \text{process} \)

\( \text{N.0: Axiom} N > 0 \)

\( \text{N_maxfaults: Axiom maxfaults} \leq N \)
precision_enhancement_ax: Axiom
\[\text{count}(\text{ppred}, N) \geq N - \text{maxfaults}\]
\[\wedge \text{okay_Readpred}(\gamma, Y, \text{ppred})\]
\[\wedge \text{okay_Readpred}(\theta, Y, \text{ppred})\]
\[\wedge \text{okay_pairs}(\gamma, \theta, X, \text{ppred}) \wedge \text{ppred}(p) \wedge \text{ppred}(q)\]
\[\supset |\text{cf}(p, \gamma) - \text{cf}(q, \theta)| \leq \pi(X, Y)\]
correct_count: Axiom \[\text{count}((\lambda p: \text{correct}(p, t)), N) \geq N - \text{maxfaults}\]
okay_Reading: function\[\text{function}[\text{process} \rightarrow \text{Clocktime}], \text{Clocktime}, \text{time} \rightarrow \text{bool}\] =
\[\lambda \gamma, Y, t: (\forall p_1, q_1: \text{correct}(p_1, t) \wedge \text{correct}(q_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \leq Y)\]
okay_Readvars: function\[\text{function}[\text{process} \rightarrow \text{Clocktime}], \text{Clocktime}, \text{Clocktime} \rightarrow \text{bool}\] =
\[\lambda \gamma, \theta, X, t: (\forall p_3: \text{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \leq X)\]
okay_Readpred_Reading: Lemma
\[\text{okay_Reading}(\gamma, Y, t) \supset \text{okay_Readpred}(\gamma, Y, (\lambda p: \text{correct}(p, t)))\]
okay_pairs_Readvars: Lemma
\[\text{okay_Readvars}(\gamma, \theta, X, t) \supset \text{okay_pairs}(\gamma, \theta, X, (\lambda p: \text{correct}(p, t)))\]
precision_enhancement: Lemma
\[\text{okay_Reading}(\gamma, Y, t_p^{t+1})\]
\[\wedge \text{okay_Reading}(\theta, Y, t_p^{t+1})\]
\[\wedge \text{okay_Readvars}(\gamma, \theta, X, t_p^{t+1})\]
\[\wedge \text{correct}(p, t_p^{t+1}) \wedge \text{correct}(q, t_p^{t+1})\]
\[\supset |\text{cf}(p, \gamma) - \text{cf}(q, \theta)| \leq \pi(X, Y)\]
okay_Reading_defn_l_r: Lemma
\[\text{okay_Reading}(\gamma, Y, t)\]
\[\supset (\forall p_1, q_1: \text{correct}(p_1, t) \wedge \text{correct}(q_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \leq Y)\]
okay_Reading_defn_r_l: Lemma
\[\forall p_1, q_1: \text{correct}(p_1, t) \supset |\gamma(p_1) - \gamma(q_1)| \leq Y\]
\[\supset \text{okay_Reading}(\gamma, Y, t)\]
okay_Readvars_defn_l_r: Lemma
\[\text{okay_Readvars}(\gamma, \theta, X, t) \supset (\forall p_3: \text{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \leq X)\]
okay_Readvars_defn_r_l: Lemma
\[\forall p_3: \text{correct}(p_3, t) \supset |\gamma(p_3) - \theta(p_3)| \leq X\]
\[\supset \text{okay_Readvars}(\gamma, \theta, X, t)\]
accuracy_preservation_ax: Axiom

okay_Readpred(\gamma, X, ppred)
\wedge \text{count}(ppred, N) \geq N - \text{maxfaults} \wedge \text{ppred}(p) \wedge \text{ppred}(q)
\implies |\text{cfn}(p, \gamma) - \gamma(q)| \leq \alpha(X)

Proof

okay_Reading_defn_rl_pr: Prove
okay_Reading_defn_rl \{p_1 \leftarrow p_1 \oplus P1S, q_1 \leftarrow q_1 \oplus P1S\} \text{ from okay_Reading}

okay_Reading_defn_lr_pr: Prove okay_Reading_defn_lr from
okay_Reading \{p_1 \leftarrow p_1 \oplus CS, q_1 \leftarrow q_1 \oplus CS\}

okay_Readvars_defn_rl_pr: Prove okay_Readvars_defn_rl \{p_3 \leftarrow p_3 \oplus P1S\} \text{ from okay_Readvars}

okay_Readvars_defn_lr_pr: Prove okay_Readvars_defn_lr from
okay_Readvars \{p_3 \leftarrow p_3 \oplus CS\}

precision_enhancement_pr: Prove precision_enhancement from
precision_enhancement_ax \{ppred \leftarrow (\lambda q: \text{correct}(q, t_p^{i+1}))\},
okay_Readpred_Reading \{t \leftarrow t_p^{i+1}\},
okay_Readpred_Reading \{t \leftarrow t_p^{i+1}, \gamma \leftarrow \theta\},
okay_pairs_Readvars \{t \leftarrow t_p^{i+1}\},
correct_count \{t \leftarrow t_p^{i+1}\}

okay_Readpred_Reading_pr: Prove okay_Readpred_Reading from
okay_Readpred \{ppred \leftarrow (\lambda p: \text{correct}(p, t))\},
okay_Reading \{p_1 \leftarrow p_1 \oplus P1S, q_1 \leftarrow q_1 \oplus P1S\}

okay_pairs_Readvars_pr: Prove okay_pairs_Readvars from
okay_pairs \{ppred \leftarrow (\lambda p: \text{correct}(p, t))\}, okay_Readvars \{p_3 \leftarrow p_3 \oplus P1S\}

rts_0_proof: Prove rts_0 from rts0 \{t \leftarrow t_p^{i+1}\}

rts_1_proof: Prove rts_1 from rts1 \{t \leftarrow t_p^{i+1}\}

End clockassumptions
basics: Module

Using clockassumptions, arith

Exporting all with clockassumptions

Theory

\[ p, q, p_1, p_2, q_1, q_2, l, m, n: \text{Var process} \]
\[ t, j, k: \text{Var event} \]
\[ x, y, z: \text{Var number} \]
\[ r, s, t, t_1, t_2: \text{Var time} \]
\[ X, Y, Z, R, S, T, T_1, T_2: \text{Var Clocktime} \]
\[ \gamma, \beta: \text{Var function}[\text{process} \rightarrow \text{time}] \]

(\ast 1 \ast 2) \ast 3: Definition function[process, process, event \rightarrow process] =

\[ (\lambda p, q, i: (\text{if } t'_p \geq t'_q \text{ then } p \text{ else } q \text{ end if})) \]

maxsync_correct: Lemma correct(p, s) \land correct(q, s) \supset correct((p \uparrow q)[i], s)

minsync: Definition function[process, process, event \rightarrow process] =

\[ (\lambda p, q, i: (\text{if } t'_p \geq t'_q \text{ then } q \text{ else } p \text{ end if})) \]

minsync_correct: Lemma correct(p, s) \land correct(q, s) \supset correct((p \downarrow q)[i], s)

minsync_maxsync: Lemma \[ t_{(p \uparrow q)[i]} \leq t_{(p \downarrow q)[i]} \]

(\ast 1 \ast 2 \ast 3): Definition function[process, process, event \rightarrow time] =

\[ (\lambda p, q, i: t'_{(p \uparrow q)[i]} ) \]

lemma.1: Lemma correct(p, t'p) \land correct(q, t'i_q) \land \beta \leq r_{min}

\[ \supset t'_p \leq t'_{i_q} + 1 \]

lemma.1.1: Lemma correct(p, t'i_q) \land correct(q, t'i_q) \land \beta \leq r_{min}

\[ \supset t'_p \leq t'_{i_q} + 1 \]

lemma.1.2: Lemma correct(p, t'p) \land correct(q, t'q)

\[ \supset t'_{i_q} + 1 \leq t'_q + r_{max} + \beta \]

lemma.2.0: Lemma correct(p, 0) \land correct(q, 0) \supset |IC_p(0) - IC_q(0)| \leq \mu

lemma.2.1: Lemma correct(q, t'i_q)

\[ \supset IC_{i_q + 1}(t'_{i_q + 1}) = cfn(q, \Theta_{i_q + 1}) \]

lemma.2.2a: Lemma

\[ \text{correct}(q, s) \land s \geq t \supset IC_{i_q}(s) \leq IC_{i_q}(t) + (s - t) \ast (1 + \rho) \]
lemma 2.2b: Lemma
\[ \text{correct}(q, s) \land s \geq t \Rightarrow IC^i_q(s) \geq IC^i_q(t) + (s - t) \times (1 - \rho) \]

abs.shift: Lemma \[ |r - s| \leq x \]
\[ \land t_1 \leq r + y + z \land t_1 \geq r + y - z \land t_2 \leq s + y + z \land t_2 \geq s + y - z \]
\[ \Rightarrow |t_1 - t_2| \leq x + 2 \times z \]

ReadClock.bnd1: Lemma
\[ \text{correct}(p, t^i_p) \land \text{correct}(q, t^i_q) \]
\[ \Rightarrow \Theta^i_{p+1} |q| \leq IC^i_q(t^i_p) + \Lambda \]

ReadClock.bnd2: Lemma
\[ \text{correct}(p, t^i_p) \land \text{correct}(q, t^i_q) \]
\[ \Rightarrow \Theta^i_{p+1} |q| \geq IC^i_q(t^i_p) - \Lambda \]

ReadClock.bnd11: Lemma
\[ \text{correct}(p, t^i_p) \land \text{correct}(q, t^i_q) \land \text{correct}(p_1, t^i_{p_1}) \land \beta \leq r_{\min} \]
\[ \Rightarrow \Theta^i_{p+1} |q| \leq IC^i_q(t^i_{p_1}) + (t^i_{p_1} - t^i_p) + (r_{\max} + \beta) \times \rho + \Lambda \]

ReadClock.bnd12: Lemma
\[ \text{correct}(p, t^i_p) \land \text{correct}(q, t^i_q) \land \text{correct}(p_1, t^i_{p_1}) \land \beta \leq r_{\min} \]
\[ \Rightarrow \Theta^i_{p+1} |q| \geq IC^i_q(t^i_{p_1}) + (t^i_{p_1} - t^i_p) - (r_{\max} + \beta) \times \rho - \Lambda \]

ReadClock.bnd: Lemma
\[ \text{correct}(p, t^i_p) \]
\[ \land \text{correct}(q, t^i_q) \]
\[ \land \text{correct}(q_1, t^i_{q_1}) \]
\[ \land |IC^i_q(t^i_{q_1}) - IC^i_q(t^i_{q_1})| \leq X \land \beta \leq r_{\min} \]
\[ \Rightarrow |\Theta^i_{p+1} |q| - \Theta^i_{p+1} |q_1|| \leq X + 2 \times ((r_{\max} + \beta) \times \rho + \Lambda) \]

okay.Reading.shift1: Lemma
\[ \text{correct}(p_1, s) \land s \geq t^i_{p_1} \]
\[ \land \beta \leq r_{\min} \]
\[ \land (\forall p, q: \]
\[ \text{correct}(p, t^i_p) \land \text{correct}(q, t^i_q) \]
\[ \Rightarrow |IC^i_p(t^i_p) - IC^i_q(t^i_q)| \leq X) \]
\[ \Rightarrow \text{okay.Reading}(\Theta^i_{p+1}, X + 2 \times ((r_{\max} + \beta) \times \rho + \Lambda), s) \]

okay.Readingvars.shift.step: Lemma
\[ s \geq t_1 - y \land s \leq t_1 + y \]
\[ \land t \geq t_2 - y \land t \leq t_2 + y \land 0 \leq t_2 - t_1 \land t_2 - t_1 \leq x \]
\[ \Rightarrow |s + x - t| \leq 2 \times y + x \]
okay.Readvars.shift_stepb: Lemma
\[ s \geq t_1 - y \land s \leq t_1 + y \]
\[ \land t \geq t_2 - y \land t \leq t_2 + y \land 0 \leq t_2 - t_1 \land t_2 - t_1 \leq x \]
\[ \Rightarrow |s - t| \leq 2 * y + x \]

okay.Readvars.shift_step1: Lemma
\[ |s - t_1| \leq y \land |t - t_2| \leq y \land 0 \leq t_2 - t_1 \land t_2 - t_1 \leq x \]
\[ \Rightarrow |s + x - t| \leq 2 * y + x \]

okay.Readvars.shift_step2: Lemma
\[ |s - t_1| \leq y \land |t - t_2| \leq y \land 0 \leq t_2 - t_1 \land t_2 - t_1 \leq x \]
\[ \Rightarrow |s - t| \leq 2 * y + x \]

okay.Readvars.shift11: Lemma
\[
\begin{align*}
\text{correct}(p, t_p^{i+1}) & \land \text{correct}(q, t_p^{i+1}) \land \text{correct}(p_1, t_p^{i+1}) \land t_p^{i+1} \geq t_q^{i+1} \\
& \Rightarrow \Theta_p^{i+1} p_1 + (PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})) - \Theta_p^{i+1} p_1 \\
& \leq 2 * \Lambda + 2 * \beta * \rho
\end{align*}
\]

okay.Readvars.shift12: Lemma
\[
\begin{align*}
\text{correct}(p, t_p^{i+1}) & \land \text{correct}(q, t_p^{i+1}) \land \text{correct}(p_1, t_p^{i+1}) \land t_p^{i+1} \geq t_q^{i+1} \\
& \Rightarrow \Theta_p^{i+1} p_1 - (\Theta_q^{i+1} p_1 + (PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}))) \\
& \leq 2 * \Lambda + 2 * \beta * \rho
\end{align*}
\]

okay.Readvars.shift1: Lemma
\[
\begin{align*}
\text{correct}(p, t_p^{i+1}) & \land \text{correct}(q, t_p^{i+1}) \land \text{correct}(p_1, t_p^{i+1}) \land t_p^{i+1} \geq t_q^{i+1} \\
& \Rightarrow |\Theta_p^{i+1} p_1 - (\Theta_q^{i+1} p_1 + (PC_q(t_p^{i+1}) - PC_q(t_q^{i+1})))| \\
& \leq 2 * \Lambda + 2 * \beta * \rho
\end{align*}
\]

okay.Readvars.shift2: Lemma
\[
\begin{align*}
\text{correct}(p, t_p^{i+1}) & \land \text{correct}(q, t_p^{i+1}) \land t_p^{i+1} \geq t_q^{i+1} \\
& \Rightarrow \text{okay.Readvars}(\Theta_p^{i+1}, \Theta_q^{i+1}, 2 * \Lambda + 2 * \beta * \rho, t)
\end{align*}
\]

okay.Readvars.shift: Lemma
\[ t \geq t_p^{i+1} \land \text{correct}(p, t) \land \text{correct}(q, t) \land t_p^{i+1} \geq t_q^{i+1} \]
\[ \Rightarrow \text{okay.Readvars}(\Theta_p^{i+1}, \Theta_q^{i+1}, 2 * \Lambda + 2 * \beta * \rho, t) \]

Proof
maxsync_correct.pr: Prove maxsync_correct from \((\ast_1 \uparrow \ast_2)[\ast 3]\)

minsinc_correct.pr: Prove minsync_correct from minsync

minsinc_maxsync.pr: Prove minsync_maxsync from minsync, \((\ast_1 \uparrow \ast_2)[\ast 3]\)

okay_Rading_shift1.proof: Prove

okay_Rading_shift1 \(\{p \leftarrow p_1@P1S, \ q \leftarrow q_1@P1S\}\) from

okay_Rading_defn.rl

\{\gamma = \Theta_{p_1}^{i+1}, \ Y = X + 2 * ((r_{max} + \beta) * \rho + \Lambda), \ t = s\},

ReadClock_bnd \(\{p \leftarrow p_1, \ q - p_1@P1S, \ q_1 - q_1@P1S\}\),

\(t_{p_1, q_1}^3\) \(\{p \leftarrow p_1@P1S, \ q - q_1@P1S\}\),

maxsync_correct \(\{p \leftarrow p_1@P1S, \ q - q_1@P1S, \ s \leftarrow t_{p_1}^{i+1}\}\),

correct_closed \(\{p \leftarrow p_1@P1S, \ t \leftarrow t_{p_1}^{i+1}\}\),

correct_closed \(\{p \leftarrow q_1@P1S, \ t \leftarrow t_{p_1}^{i+1}\}\),

correct_closed \(\{p \leftarrow p_1@P1S, \ t \leftarrow t_{p, q_1}^{i+1}, \ s \leftarrow t_{p_1}^{i+1}\}\),

correct_closed \(\{p \leftarrow q_1@P1S, \ t \leftarrow t_{p, q_1}^{i+1}, \ s \leftarrow t_{p_1}^{i+1}\}\),

correct_closed \(\{p \leftarrow p_1, \ t \leftarrow t_{p_1}^{i+1}\}\),

lemma_1.1 \(\{q \leftarrow p_1, \ p \leftarrow (p \uparrow q)[i]\}\)

ReadClock_bnd.proof: Prove ReadClock_bnd from

ReadClock_bnd11 \(\{p_1 \leftarrow (q \uparrow q_1)[i]\}\),

ReadClock_bnd12 \(\{p_1 \leftarrow (q \uparrow q_1)[i]\}\),

ReadClock_bnd11 \(\{q \leftarrow q_1, \ p_1 \leftarrow (q \uparrow q_1)[i]\}\),

ReadClock_bnd12 \(\{q \leftarrow q_1, \ p_1 \leftarrow (q \uparrow q_1)[i]\}\),

lemma_1.1 \(\{p \leftarrow (q \uparrow q_1)[i], \ q \leftarrow p\}\),

correct_closed

\(\{p \leftarrow (q \uparrow q_1)[i], \ s \leftarrow t_{p}^{i+1}, \ t \leftarrow t_{(p \uparrow q_1)[i]}\}\),

abs_shift

\(\{r = IC_{q_1}^i(t_{q, q_1}^i), \ s = IC_{q_1}^i(t_{q, q_1}^i), \ t_1 = \Theta_{p}^{i+1}(q_1), \ t_2 = \Theta_{p}^{i+1}(q_1), \ y = (t_{p}^{i+1} - t_{q, q_1}^i), \ z = (r_{max} + \beta) * \rho + \Lambda, \ x = X\}\),

\(t_{p_1, q_1}^3\) \(\{p \leftarrow q, \ q \leftarrow q_1\},

maxsync_correct \(\{p \leftarrow q, \ q \leftarrow q_1, \ s \leftarrow t_{p_1}^{i+1}\}\)
ReadClock_bnd11_proof: Prove ReadClock_bnd11 from ReadClock_bnd1,
lemma_2.2a \( s \leftarrow t_p^{i+1}, t \leftarrow t_{p_1}^i \),
lemma_1.2 \( q \leftarrow p_1 \),
lemma_1 \( q \leftarrow p, p \leftarrow p_1 \),
multJdistrib \( x \leftarrow t_p^{i+1} - t_{p_1}^i, y \leftarrow 1, z \leftarrow \rho \),
multJeq \( x \leftarrow r_{max} + \beta, y \leftarrow t_p^{i+1} - t_{p_1}^i, z \leftarrow \rho \),
mult_rident \( x \leftarrow t_p^{i+1} - t_{p_1}^i \),
\rho_0

ReadClock_bnd12_proof: Prove ReadClock_bnd12 from ReadClock_bnd2,
lemma_2.2b \( s \leftarrow t_p^{i+1}, t \leftarrow t_{p_1}^i \),
lemma_1.2 \( q \leftarrow p_1 \),
lemma_1 \( q \leftarrow p, p \leftarrow p_1 \),
multJdistrib \( x \leftarrow t_p^{i+1} - t_{p_1}^i, y \leftarrow 1, z \leftarrow \rho \),
multJeq \( x \leftarrow r_{max} + \beta, y \leftarrow t_p^{i+1} - t_{p_1}^i, z \leftarrow \rho \),
mult_rident \( x \leftarrow t_p^{i+1} - t_{p_1}^i \),
\rho_0

ReadClock_bnd1_proof: Prove ReadClock_bnd1 from Readerror,
\(|1*1| \{ x \leftarrow \Theta_p^{i+1}q \} - IC_q^i(t_p^{i+1}) \}

ReadClock_bnd2_proof: Prove ReadClock_bnd2 from Readerror,
\(|1*1| \{ x \leftarrow \Theta_p^{i+1}q \} - IC_q^i(t_p^{i+1}) \}

okay_Readvars_shift_step1_proof: Prove okay_Readvars_shift_step1 from okay_Readvars_shift_step,
\(|1*1| \{ x \leftarrow s - t_1 \} \), \(|1*1| \{ x \leftarrow t - t_2 \} \)

okay_Readvars_shift_step2_proof: Prove okay_Readvars_shift_step2 from okay_Readvars_shift_stepb,
\(|1*1| \{ x \leftarrow s - t_1 \} \), \(|1*1| \{ x \leftarrow t - t_2 \} \)

okay_Readvars_shift11_proof: Prove okay_Readvars_shift11 from ReadClock_bnd2,
\( q \leftarrow p_1 \),
ReadClock_bnd1 \( p \leftarrow q, q \leftarrow p_1 \),
correct_closed \( s \leftarrow t_p^{i+1}, t \leftarrow t_{p_1}^i, p \leftarrow p_1 \),
correct_closed \( s \leftarrow t_p^{i+1}, t \leftarrow t_q^{i+1}, p \leftarrow p \),
lemma_2.2b \( q \leftarrow p_1, s \leftarrow t_p^{i+1}, t \leftarrow t_q^{i+1} \),
rates \( s \leftarrow t_p^{i+1}, t \leftarrow t_q^{i+1}, p \leftarrow p \),
multJdistrib \( x \leftarrow t_p^{i+1} - t_{p_1}^i, y \leftarrow 1, z \leftarrow \rho \),
multJdistrib \( x \leftarrow t_p^{i+1} - t_q^{i+1}, y \leftarrow 1, z \leftarrow \rho \),
multJeq \( x \leftarrow \beta, y \leftarrow t_p^{i+1} - t_q^{i+1}, z \leftarrow \rho \),
rts \( i \leftarrow i + 1 \),
\rho_0
okay_Readvars_shift12_proof: Prove okay_Readvars_shift12 from
ReadClock_bnd1 \{ q \equiv p_1 \},
ReadClock_bnd2 \{ p \equiv q, q \equiv p_1 \},
correct_closed \{ s \equiv t_{p+1}, t \equiv t_{q+1}, p \equiv p_1 \},
correct_closed \{ s \equiv t_{p+1}, t \equiv t_{q+1}, p \equiv q \},
lemma.2.2a \{ q \equiv p_1, s \equiv t_{p+1}, t \equiv t_{q+1} \},
rate_2 \{ s \equiv t_{p+1}, t \equiv t_{q+1}, p \equiv q \},
multAdistrib_minus \{ x \equiv t_{p+1} - t_{q+1}, y \equiv 1, z \equiv \rho \},
multAdistrib \{ x \equiv t_{p+1} - t_{q+1}, y \equiv 1, z \equiv \rho \},
mult_leq \{ x \equiv \beta, y \equiv t_{p+1} - t_{q+1}, z \equiv \rho \},
rts.2 \{ i \equiv i + 1 \},
rho_0

okay_Readvars_shift1_proof: Prove okay_Readvars_shift1 from
okay_Readvars_shift11,
okay_Readvars_shift12,
abs_diff.3
\{ y \equiv \Theta(t_{p+1}) + (PC_q(t_{p+1}) - PC_q(t_{q+1})),
x \equiv \Theta(t_{p+1}),
z \equiv 2 * \Lambda + 2 * \beta * \rho \}

okay_Readvars_shift_step_proof: Prove okay_Readvars_shift_step from
\{ x \equiv s + x - t \}

okay_Readvars_shift_stepb_proof: Prove okay_Readvars_shift_stepb from
\{ x \equiv s - t, \}, \{ x \equiv t_2 - t_1 \}

okay_Readvars_shift_proof: Prove okay_Readvars_shift from
okay_Readvars_shift1 \{ p_1 \equiv p_0@P2S \},
okay_Readvars_defn.rl
\{ \theta \equiv (\lambda p_1 \equiv \text{time} \Theta(t_{p+1}) + PC_q(t_{p+1}) - PC_q(t_{q+1})),
\gamma \equiv \Theta(t_{p+1}),
X \equiv 2 * \Lambda + 2 * \beta * \rho \},
correct_closed \{ s \equiv t, t \equiv t_{p+1} \},
correct_closed \{ p \equiv q, s \equiv t, t \equiv t_{p+1} \},
correct_closed \{ p \equiv p_3@P2S, s \equiv t, t \equiv t_{p+1} \}

lemma.1_proof: Prove lemma.1 from
rts.1 \{ p \equiv q \},
rts.2,
rmin.0,
correct_closed \{ p \equiv q, s \equiv t_{q+1}, t \equiv t_{q+1} \}
lemma_1.2_proof: Prove lemma_1.2 from
rts_0,
rts_1,
rts_2,
rmin_0,
correct_closed \{ s \rightarrow t_p^{t+1}, t \rightarrow t_p^t \}

lemma_2.0_proof: Prove lemma_2.0 from
synctime_0,
synctime_0 \{ p \rightarrow q \},
IClock_defn \{ p \rightarrow q, i \rightarrow 0, t \rightarrow 0 \},
IClock_defn \{ i \rightarrow 0, t \rightarrow 0 \},
Adj \{ i \rightarrow -0, p \rightarrow q \},
Adj \{ i \rightarrow 0 \},
initial \{ p \rightarrow q \},
initial,

\hspace{0.5cm} rts_1 \{ p \rightarrow q, i \rightarrow 0 \},
rts_1 \{ i \rightarrow 0 \},
rmin_0,
mu_0,
abs_bnd \{ x \rightarrow IC_p^0(t_p^0), y \rightarrow IC_q^0(t_q^0), z \rightarrow \mu \}

lemma_2.1_proof: Prove lemma_2.1 from
IClock_defn \{ p \rightarrow q, i \rightarrow i+1, t \rightarrow t_q^{i+1} \},
Adj \{ i \rightarrow i+1, p \rightarrow q \}

lemma_2.2a_proof: Prove lemma_2.2a from
IClock_defn \{ p \rightarrow q, t \rightarrow s \},
IClock_defn \{ p \rightarrow q \},
rate_1 \{ p \rightarrow q \},
correct_closed \{ p \rightarrow q \}

lemma_2.2b_proof: Prove lemma_2.2b from
IClock_defn \{ p \rightarrow q, t \rightarrow s \},
IClock_defn \{ p \rightarrow q \},
rate_2 \{ p \rightarrow q \},
correct_closed \{ p \rightarrow q \}

abs_shift_proof: Prove abs_shift from |*1| \{ x \rightarrow r - s \}, |*1| \{ x \rightarrow t_1 - t_2 \}
lemmaA_l-proof: Prove lemmaA_l from
rts.1 \{ p \leftarrow q \},
rts2 \{ t \leftarrow t_{+1}' \},
\betaa_0,
\text{rmin}_0,
correct\_closed \{ p \leftarrow q, s \leftarrow t_{+1}', t \leftarrow t' \}

\textbf{End basics}
readbounds: Module

Using basics, clockassumptions, arith

Exporting all with basics

Theory

\( p, q, p_1, p_2, q_1, q_2, i, m, n \): Var process
\( i, j, k \): Var event
\( X, Y, Z, R, S, T, T_1, T_2 \): Var Clocktime
\( x, y, z, r, s, t, t_1, t_2 \): Var number
\( \gamma, \delta \): Var function[process \( \rightarrow \) Clocktime]
prop: Var function[nat \( \rightarrow \) bool]
okaymaxsync: function[nat, Clocktime \( \rightarrow \) bool] =

\[ \lambda i, X : (\forall p, q:
  \text{correct}(p, t_{\mathit{p,q}}^i) \land \text{correct}(q, t_{\mathit{p,q}}^i)
  \implies I_C^i(t_{\mathit{p,q}}^i) - I_C^i(t_{\mathit{p,q}}^i) \leq X) \]

okaymaxsync.defn.lr: Lemma
okaymaxsync(i, X)
\( \therefore (\forall p, q:
  \text{correct}(p, t_{\mathit{p,q}}^i) \land \text{correct}(q, t_{\mathit{p,q}}^i)
  \implies I_C^i(t_{\mathit{p,q}}^i) - I_C^i(t_{\mathit{p,q}}^i) \leq X) \)

okaymaxsync.defn.rl: Lemma
\( (\forall p, q: \text{correct}(p, t_{\mathit{p,q}}^i) \land \text{correct}(q, t_{\mathit{p,q}}^i)
  \implies I_C^i(t_{\mathit{p,q}}^i) - I_C^i(t_{\mathit{p,q}}^i) \leq X) \)
\( \therefore \) okaymaxsync(i, X)

lemma.2_base: Lemma \( \mu \leq X \therefore \) okaymaxsync(0, X)

okay.Reading.shift2: Lemma
\( \text{correct}(p_1, s) \land s \geq t_{\mathit{p,q}}^{i+1} \land \beta \leq r_{\min} \land \) okaymaxsync(i, X)
\( \therefore \) okay.Reading(\( \Theta_p^{i+1}, X + 2 \ast ((r_{\max} + \beta) \ast p + \Lambda), s \))

Cfn.IClock1: Lemma
\( \text{correct}(q, t_{\mathit{p,q}}^{i+1}) \land \text{correct}(p, t_{\mathit{p,q}}^{i+1}) \land t_{\mathit{p,q}}^{i+1} \geq t_{\mathit{q}}^{i+1}
  \implies I_C^{i+1}(t_{\mathit{q}}^{i+1})
  = cfn(q, (\lambda p_1 \rightarrow \text{time: } \Theta_p^{i+1}p_1) + PC_q(t_{\mathit{p,q}}^{i+1}) - PC_q(t_{\mathit{q}}^{i+1})) \)

okay.Reading.plus: Lemma
okay.Reading(\( \gamma, Y, t \)) \( \therefore \) okay.Reading((\( \lambda p_1 \rightarrow \text{time: } \gamma(p_1) + X \), Y, t)
lemma 2.ind1: Lemma
\[ \beta \leq r_{\min} \land \pi(2 \cdot \Lambda + 2 \cdot \beta \cdot \rho, X + 2 \cdot ((r_{\max} + \beta) \cdot \rho + \Lambda)) \leq X \]
\[ \land \text{okaymaxsync}(i, X) \]
\[ \land t_p^{i+1} \geq t_q^{i+1} \land \text{correct}(p, t_p^{i+1}) \land \text{correct}(q, t_q^{i+1}) \]
\[ \supset |\text{cfn}(p, \Theta_p^{i+1}) - \text{cfn}(q, \Theta_q^{i+1})| \]
\[ (\lambda p_1 \rightarrow \text{time:} \Theta_q^{i+1}(p_1) + PC_q(t_q^{i+1}) - PC_q(t_q^{i+1})) \]
\[ \leq X \]

lemma 2.abs_fact: Lemma
\[ t_1 \leq t \land t \leq t_2 \land |s - t_1| \leq X \land |s - t_2| \leq X \lor |s - t| \leq X \]

lemma 2.ind3: Lemma
\[ \beta \leq r_{\min} \land \pi(2 \cdot \Lambda + 2 \cdot \beta \cdot \rho, X + 2 \cdot ((r_{\max} + \beta) \cdot \rho + \Lambda)) \leq X \]
\[ \land \text{okaymaxsync}(i, X) \]
\[ \land t_p^{i+1} \geq t_q^{i+1} \land \text{correct}(p, t_p^{i+1}) \land \text{correct}(q, t_q^{i+1}) \]
\[ \supset |IC_p^{i+1}(t_p^{i+1}) - IC_q^{i+1}(t_q^{i+1})| \leq X \]

lemma 2.ind.step: Lemma
\[ |IC_{p^{i+1}}^{i}(t) - IC_{q^{i+1}}^{i}(t)| \leq X \supset |IC_p^{i}(t) - IC_q^{i}(t)| \leq X \]

lemma 2.ind: Lemma
\[ \beta \leq r_{\min} \land \pi(2 \cdot \Lambda + 2 \cdot \beta \cdot \rho, X + 2 \cdot ((r_{\max} + \beta) \cdot \rho + \Lambda)) \leq X \]
\[ \land \text{okaymaxsync}(i, X) \]
\[ \supset \text{okaymaxsync}(i + 1, X) \]

lemma 2: Lemma \[ \beta \leq r_{\min} \land \mu \leq X \land \pi(2 \cdot \Lambda + 2 \cdot \beta \cdot \rho, X + 2 \cdot ((r_{\max} + \beta) \cdot \rho + \Lambda)) \leq X \]
\[ \supset \text{okaymaxsync}(i, X) \]

induction: Axiom \[ \text{prop}(0) \land (\forall j: \text{prop}(j) \supset \text{prop}(j + 1)) \supset \text{prop}(i) \]

Proof

okaymaxsync.defn.lr.pr: Prove okaymaxsync.defn.lr from
\[ \text{okaymaxsync} \{ p \rightarrow p@CS, q \rightarrow q@CS \} \]

okaymaxsync.defn.rl.pr: Prove
\[ \text{okaymaxsync.defn.rl} \{ p \rightarrow p@P1S, q \rightarrow q@P1S \} \text{ from okaymaxsync} \]

lemma 2.base_proof: Prove lemma 2.base from
\[ t_2^{i+2} \{ i \leftarrow 0, p \rightarrow p@P4S, q \rightarrow q@P4S \}, \]
\[ \text{synctime}_0 \{ p \leftarrow (p@P4S \uparrow q@P4S)[0] \}, \]
\[ \text{lemma 2.0} \{ p \rightarrow p@P4S, q \rightarrow q@P4S \}, \]
\[ \text{okaymaxsync.defn.rl} \{ i \leftarrow 0 \} \]
okay_Reading.shift2.proof: Prove okay_Reading.shift2 from
okay_Reading.shift1, okaymaxsync.defn.lr {p ← p@P1S, q ← q@P1S}

Cfn._Clock1.proof: Prove Cfn._Clock1 from
IClock.defn {p ← q, t ← t_p^{i+1}, i ← i + 1},
Adj {p ← q, i ← i + 1},
translalation_invariance
{p ← q),
γ ← T^i+1_q,
X ← PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}),
rate_2 {p ← q, s ← t_p^{i+1}, t ← t_q^{i+1}},
rho_i,
pos.product {x ← t_p^{i+1} - t_q^{i+1}, y ← 1 - ρ}

okay_Reading.plus.proof: Prove okay_Reading.plus from
okay_Reading.defn.lr {p_1 ← p_1@P2S, q_1 ← q_1@P2S},
okay_Reading.defn.rl {γ ← (λ p_1 → time: γ(p_1) + X)}

lemma_2.ind1.proof: Prove lemma_2.ind1 from
precision_enhancement
{θ ← (λ p_1 → time: T^{i+1}_q p_1) + PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}),
γ ← T^i+1_q, X ← 2 * Λ + 2 * β * p, Y ← X + 2 * ((r_max + β) * ρ + Λ)},
okay_Reader.shift {t ← t_p^{i+1}},
okay_Reading.shift2 {p_1 ← p, s ← t_p^{i+1}},
okay_Reading.shift2 {p_1 ← q, s ← t_p^{i+1}},
okay.Reading.plus
{γ ← T^i+1_q, t ← t_p^{i+1}, X ← PC_q(t_p^{i+1}) - PC_q(t_q^{i+1}), Y ← X + 2 * ((r_max + β) * ρ + Λ)},
correct.closed {p ← q, s ← t_p^{i+1}, t ← t_q^{i+1}}

lemma2.abs.fact.proof: Prove lemma2.abs.fact from
|*1| {x ← s - t_1}, |*1| {x ← s - t_2}, |*1| {x ← s - t}
lemma_2.ind3.proof: Prove lemma_2.ind3 from
lemma_2.ind1,
lemma2.abs_fact
{s ← IC_p^{i+1}(t_p^{i+1}),
t ← IC_q^{i+1}(t_p^{i+1}),
t_1 ← cf\(q, \Theta_q^{i+1}\),
t_2 ← cf\(q, (\lambda p_1 → \text{time: } \Theta_q^{i+1}p_1) + \beta*(1+\rho))\),
X = X},
lemma_2.1 {q ← p},
Cfn.IClock1

lemma_2.ind.step.proof: Prove lemma_2.ind.step from
(*1) (*2) (*3), minsync, abs.com \(x ← IC_p(t), y ← IC_q(t)\)
End readbounds
lemma3: Module

Using readbounds, basics, clockassumptions, arith

Exporting all with readbounds

Theory

prop: Var function[nat → bool]
l, m, n, p0, q0, p, q, p1, p2, q1, q2: Var process
i, j, k: Var event
x, y, z, r, s, t, t1, t2, x1, x2, y1, y2: Var time
X, Y, Z, R, S, T, T1, T2, X1, X2, Y1, Y2: Var Clocktime
γ, θ: Var function[process → Clocktime]
abs_IClock_diff: function[nat, Clocktime → bool]
IClock_Reading: function[nat, time → function[process → Clocktime]]
δ: time

maxmax_gap: Lemma
  correct(p, s) ∧ correct(q, s)
  ∧ s ≥ t ∧ s ≤ t_{p+1}^i ∧ t ≥ t_{p+1}^i
  ⇒ s - t ≤ r_{max}

minmax_gap: Lemma
  correct(p, s) ∧ correct(q, s)
  ∧ s ≥ t ∧ s ≤ t_{p+1}^i ∧ t ≥ t_{p+1}^i
  ⇒ s - t ≤ r_{max}

drift_bnd: Lemma t ≤ s
  ∧ correct(p, s) ∧ correct(q, s) ∧ |IC^i_p(t) - IC^i_q(t)| ≤ X
  ⇒ |IC^i_p(s) - IC^i_q(s)| ≤ X + 2 * (s - t) * ρ

maxsync_max: Lemma t_{p+1}^i ≥ t_{p}^i ∧ t_{p+1}^i ≥ t_{p}^i

minsync_min: Lemma t_{p+1}^i ≤ t_{p}^i ∧ t_{p+1}^i ≤ t_{p}^i

accuracy_preservation: Lemma
  correct(p, t_{p+1}^i)
  ∧ correct(q, t_{p+1}^i)
  ∧ (∀ l, m:
    correct(l, t_{p+1}^i) ∧ correct(m, t_{p+1}^i)
    ⇒ |IC^i_p(t_{p+1}^i) - IC^i_m(t_{p+1}^i)| ≤ X)
  ⇒ |IC^i_p(t_{p+1}^i) - IC^i_q(t_{p+1}^i)| ≤ α(X + 2 * Λ) + Λ

accuracy_pres_step0: Lemma
  |s - t_1| ≤ y ∧ |t - t_2| ≤ y ∧ |t_1 - t_2| ≤ x ⇒ |s - t| ≤ 2 * y + x

68
accuracy_pres_step1: Lemma
\[ \text{correct}(p, t_p^{i+1}) \land \text{correct}(l, t_l^{i+1}) \land \text{correct}(m, t_m^{i+1}) \]
\[ \supset |\Theta_p^{i+1} - \Theta_p^{i+1}m| \]
\[ \leq |IC_p^i(t_p^{i+1}) - IC_p^i(t_p^{i+1})| + 2 \cdot \Lambda \]

lemma3.1.1: Lemma
\[ \text{correct}(p, t) \land \text{correct}(q, t) \]
\[ \wedge \beta \leq r_{\text{min}} \]
\[ \wedge \mu \leq X \]
\[ \wedge \pi(2 \cdot \Lambda + 2 \cdot \beta \cdot \rho, X + 2 \cdot ((\max + \beta) \cdot \rho + \Lambda)) \leq X \]
\[ \wedge t \geq t_p^i[p^q[i] \land t < t_p^i[p^q[i+1] \]
\[ \supset |IC_p^i(t) - IC_p^i(t)| \leq X + 2 \cdot (t - t_p^i[p^q[i]) \cdot \rho \]

lemma3.1: Lemma
\[ \text{correct}(p, t) \]
\[ \land \text{correct}(q, t) \]
\[ \wedge \beta \leq r_{\text{min}} \]
\[ \wedge \mu \leq X \]
\[ \wedge \pi(2 \cdot \Lambda + 2 \cdot \beta \cdot \rho, X + 2 \cdot ((\max + \beta) \cdot \rho + \Lambda)) \leq X \]
\[ \wedge t \geq t_p^i[p^q[i] \land t < t_p^i[p^q[i+1] \]
\[ \supset |VC_p(t) - VC_q(t)| \leq X + 2 \cdot r_{\text{max}} \cdot \rho \]

lemma3.2.0: Lemma
\[ \text{correct}(p, t_p^{i+1}) \]
\[ \land \text{correct}(q, t_q^{i+1}) \]
\[ \wedge \beta \leq r_{\text{min}} \]
\[ \wedge \mu \leq X \]
\[ \wedge \pi(2 \cdot \Lambda + 2 \cdot \beta \cdot \rho, X + 2 \cdot ((\max + \beta) \cdot \rho + \Lambda)) \leq X \]
\[ \wedge t \geq t_p^i[p^q[i] \land t < t_p^i[p^q[i+1] \]
\[ \supset |IC_p^i(p^q[i+1] - IC_q^i(p^q[i+1] \leq X + 2 \cdot (r_{\text{max}} + \beta) \cdot \rho + 2 \cdot \Lambda) + \Lambda \]

lemma3.2.1: Lemma
\[ \text{correct}(p, t) \land \text{correct}(q, t) \]
\[ \wedge \beta \leq r_{\text{min}} \]
\[ \wedge \mu \leq X \]
\[ \wedge \pi(2 \cdot \Lambda + 2 \cdot \beta \cdot \rho, X + 2 \cdot ((\max + \beta) \cdot \rho + \Lambda)) \leq X \]
\[ \wedge \alpha(X + 2 \cdot (r_{\text{max}} + \beta) \cdot \rho + 2 \cdot \Lambda + \Lambda) + 2 \cdot \beta \cdot \rho \leq \delta \]
\[ \wedge t \geq t_p^i[p^q[i+1] \land t < t_p^i[p^q[i+1] \]
\[ \supset |IC_p^i(p^q[i+1] - IC_q^i(p^q[i+1] \leq \delta \]

lemma3.2 step: Lemma
\[ \text{correct}(p, t) \land \text{correct}(q, t) \land \beta \leq r_{\text{min}} \land t \geq t_p^i[p^q[i] \land t < t_p^i[p^q[i] \]
\[ \supset t < t_p^i[p^q[i] \]

69
lemma3.2_step1: Lemma
\[
correct(p, t) \land correct(q, t) \land \beta \leq r_{\text{min}} \land t \geq t_{(p \neq q)[i+1]} \implies t \geq t_{(p \neq q)[i+1]}
\]

lemma3.2_step2: Lemma
\[
correct(p, t) \land correct(q, t) \\
\land \beta \leq r_{\text{min}} \land t \geq t_{(p \neq q)[i+1]} \land t < t_{(p \neq q)[i+1]} \\
\implies |IC_{(p \neq q)[i+1]}(t) - IC_{(p \neq q)[i+1]}(t)| \\
= |VC_{(p \neq q)[i+1]}(t) - VC_{(p \neq q)[i+1]}(t)|
\]

lemma3.2_step3: Lemma
\[
|VC_{(p \neq q)[i+1]}(t) - VC_{(p \neq q)[i+1]}(t)| = |VC_p(t) - VC_q(t)|
\]

lemma3.2: Lemma
\[
correct(p, t) \\
\land correct(q, t) \\
\land \beta \leq r_{\text{min}} \\
\land \mu \leq X \\
\land \pi(2 * A + 2 * \beta * \rho, X + 2 * ((r_{\text{max}} + \beta) * \rho + \Lambda)) \leq X \\
\land \alpha(X + 2 * (r_{\text{max}} + \beta) * \rho + 2 * \Lambda + 2 * \beta * \rho \leq \delta \\
\land X + 2 * r_{\text{max}} * \rho \leq \delta \\
\land t \geq t_{(p \neq q)[i]} \land t < t_{(p \neq q)[i+1]} \\
\implies |VC_p(t) - VC_q(t)| \leq \delta
\]

okayClocks: function[process, process, nat → bool] =
\[
(\lambda p, q, i: (\forall t: t \geq 0 \land t < t_{(p \neq q)[i]} \land correct(p, t) \land correct(q, t) \\
\implies |VC_p(t) - VC_q(t)| \leq \delta))
\]

okayClocks_defn_lr: Lemma
\[
\text{okayClocks}(p, q, i) \\
\implies (\forall t: t \geq 0 \land t < t_{(p \neq q)[i]} \land correct(p, t) \land correct(q, t) \\
\implies |VC_p(t) - VC_q(t)| \leq \delta)
\]

okayClocks_defn_rl: Lemma
\[
(\forall t: t \geq 0 \land t < t_{(p \neq q)[i]} \land correct(p, t) \land correct(q, t) \\
\implies |VC_p(t) - VC_q(t)| \leq \delta) \\
\implies \text{okayClocks}(p, q, i)
\]

lemma3.3.0: Lemma
\[
\mu \leq \delta \implies \text{okayClocks}(p, q, 0)
\]
lemma3.3.ind: Lemma
\[ \beta \leq r_{\min} \land \mu \leq \delta_s \]
\[ \land \pi(2 \star \Lambda + 2 \star \beta \star \rho, \delta_s + 2 \star ((r_{\text{max}} + \beta) \star \rho + \Lambda)) \leq \delta_s \]
\[ \land \delta_s + 2 \star \alpha(\delta_s + 2 \star (r_{\text{max}} + \beta) \star \rho + 2 \star \Lambda) + \Lambda + 2 \star \beta \star \rho \leq \delta \]
\[ \cup \text{okayClocks}(p, q, i) \]

\[ \cup \text{okayClocks}(p, q, i + 1) \]

\[ \cup \text{okayClocks}(p, q, i) \]

\[ \cup \text{okayClocks}(p, q, i) \]

Proof

okayClocks.defn.lr.pr: Prove okayClocks.defn.lr from okayClocks \{t \leftarrow t@CS\}

okayClocks.defn.rl.pr: Prove okayClocks.defn.rl \{t \leftarrow t@P1S\} from okayClocks

accuracy.pres.step2: Lemma
\[ \varepsilon \geq 0 \land y_1 - z \leq y \land y_1 + z \leq y \land |x - y| \leq |x - y_1| + z \]

accuracy.pres.step2.pr: Prove accuracy.pres.step2 from
\[ |*1| \{x \leftarrow x - y\}, |*1| \{x \leftarrow x - y_1\} \]

71
accuracy_preservation_pr: Prove

accuracy_preservation \{l \rightarrow l @ P2S, m \leftarrow m @ P2S\} from
accuracy_preservation_ax

\( \{ \text{ppred} \leftarrow (\lambda q: \text{correct}(q, t^{i+1})) \}, \)
\( \gamma \leftarrow \Theta^{i+1} \),
\( X \leftarrow X + 2 * \Lambda \}, \)

okay_Readpred
\( \{ Y \leftarrow X + 2 * \Lambda, \)
\( \text{ppred} \leftarrow (\lambda q: \text{correct}(q, t^{i+1})), \)
\( \gamma \leftarrow \Theta^{i+1} \}, \)

accuracy_pres_step1 \{l \rightarrow l @ P2S, m \leftarrow m @ P2S\},
accuracy_pres_step2
\( \{ z \leftarrow \Lambda, \)
\( y_1 \leftarrow \Theta^{i+1} q), \)
\( y \leftarrow IC_q(t^{i+1}), \)
\( x \leftarrow IC_q(t^{i+1}), \)
ReadClock_bnd1,
ReadClock_bnd2,
correct_count \{ t \leftarrow t^{i+1} \},
IClock_defn \{ i \leftarrow i + 1, t \leftarrow t^{i+1} \},
Adj \{ i \leftarrow i + 1 \}

abs_diff_2: Lemma \(|x - y| \leq z \land z - y \leq z \land y - x \leq z\)

abs_diff_2_pr: Prove abs_diff_2 from \(|x - y| \leq z \land z - y \leq z \land y - x \leq z\)

accuracy_pres_step0_pr: Prove accuracy_pres_step0 from
okay_Readvars_shift_step2,
okay_Readvars_shift_step2
\( \{ t_1 \leftarrow t_2, \)
\( t_2 \leftarrow t_1, \)
\( s \leftarrow t, \)
\( t \leftarrow s \}, \)
abs_diff_2 \{ x \leftarrow t_1, y \leftarrow t_2, z \leftarrow x \},
abs_com \{ z \leftarrow s, y \leftarrow t \}

72
accuracy_pres_step1_pr: Prove accuracy_pres_step1 from accuracy_pres_step0

\{ y \leftarrow \Lambda, \\
x \leftarrow |IC_i(t_p^{t+1}) - IC_m(t_p^{t+1})|, \\
s \leftarrow \Theta_p^{t+1}, \\
t_1 \leftarrow IC_i(t_p^{t+1}), \\
t \leftarrow \Theta_p^{t+1}, \\
t_2 \leftarrow IC_m(t_p^{t+1}) \}.

Readerror \{ q \leftarrow l \}, 
Readerror \{ q \leftarrow m \}, 
abs_com \{ x \leftarrow IC_i(t_p^{t+1}), y \leftarrow \Theta_p^{t+1} \}, 
abs_com \{ x \leftarrow IC_m(t_p^{t+1}), y \leftarrow \Theta_p^{t+1} \}.

lemma3_3_proof: Prove lemma3_3 from lemma3_3ind \{ i \leftarrow j \}.

readbounds.induction \{ prop \leftarrow (\lambda i \rightarrow \text{bool}: \\
\beta \leq r_{\text{min}} \land \mu \leq \delta s \\
\land \pi(2 \ast \Lambda + 2 \ast \beta \ast \rho, \delta s + 2 \ast ((r_{\text{max}} + \beta) \ast \rho + \Lambda)) \leq \delta s \\
\land \delta s + 2 \ast r_{\text{max}} \ast \rho \leq \delta \\
\land \alpha(\delta s + 2 \ast (r_{\text{max}} + \beta) \ast \rho + 2 \ast \Lambda) + \Lambda + 2 \ast \beta \ast \rho \leq \delta \\
\lor \text{okayClocks}(p, q, i)) \},

\text{lemma3_3_0,}
\text{pos_product \{ x \leftarrow r_{\text{max}}, y \leftarrow \rho \},}
\text{rmax_0,}
\text{rho_0.}

lemma3_3_ind_proof: Prove lemma3_3_ind from

\text{lemma3_2 \{ t \leftarrow t \ast P3S, X \leftarrow \delta s \},}
\text{okayClocks.defn Jr \{ t \leftarrow t \ast P3S \},}
\text{okayClocks.defn rl \{ i \leftarrow i + 1 \}}

lemma3_3_0_proof: Prove lemma3_3_0 from

\text{okayClocks.defn rl \{ i \leftarrow 0 \},}
\text{synctime_0 \{ p \leftarrow (p \uparrow q)[0] \},}
\text{synctime_0,}
\text{synctime_0 \{ p \leftarrow q \},}
\text{VClock.defn \{ t \leftarrow t \ast P1S, i \leftarrow 0 \},}
\text{VClock.defn \{ p \leftarrow q, t \leftarrow t \ast P1S, i \leftarrow 0 \},}
\text{lemma2_0,}
\text{rts1 \{ t \leftarrow t \ast P1S, i \leftarrow 0 \},}
\text{rts1 \{ p \leftarrow q, t \leftarrow t \ast P1S, i \leftarrow 0 \},}
\text{rmin_0}.
lemma3A_lproof: Prove lemma3A_l from
lemma2,
okaymaxsync_defn_lr \{p \leftarrow p, q \leftarrow q\},
t^3_{1,2},
drift_bnd \{s \leftarrow t, t \leftarrow t'(p \circ q)[i], Y \leftarrow X, j \leftarrow i\},
rho_0,
correct_closed \{s \leftarrow t, t \leftarrow t'(p \circ q)[i]\},
correct_closed \{s \leftarrow t, t \leftarrow t'(p \circ q)[i], p \leftarrow q\},
mult_leq \{z \leftarrow \rho, y \leftarrow t - t'(p \circ q)[i], x \leftarrow r_{max}\},
maxsync_max,
minsync_min \{i \leftarrow i + 1\},
minmax_gap \{s \leftarrow t, t \leftarrow t^i_p,q\}.

lemma3A_lproof: Prove lemma3A_l from
lemma3A.l,
VClock_defn,
VClock_defn \{p \leftarrow q\},
rtso,
mult_leq \{z \leftarrow \rho, y \leftarrow t - t'(p \circ q)[i], x \leftarrow r_{max}\},
maxsync_max,
minsync_min \{i \leftarrow i + 1\},
rho_0.
lemma3.2.0_proof: Prove lemma3.2.0 from

lemma3.1.1 \{ p \leftarrow l@P2S, \; q \leftarrow m@P2S, \; t \leftarrow t_{(p \uplus q)[i+1]} \},

accuracy_perservation

\begin{align*}
\{ p \leftarrow (p \uplus q)[i+1], \\
q \leftarrow (p \uparrow q)[i+1], \\
X \leftarrow X + 2 \ast (r_{\text{max}} + \beta) \ast \rho \}
\end{align*}

lemma_1.2 \{ p \leftarrow (p \uplus q)[i+1], \; q \leftarrow (l@P2S \uparrow m@P2S)[i] \},

multi_eq

\begin{align*}
\{ x \leftarrow r_{\text{max}} + \beta, \\
y \leftarrow t_{(p \uplus q)[i+1]}^{t+1} - t_{(l@P2S \uparrow m@P2S)[i]}^{t+1}, \\
z \leftarrow \rho \}
\end{align*}

lemma_1.1 \{ q \leftarrow (p \uplus q)[i+1], \; p \leftarrow (l@P2S \uparrow m@P2S)[i] \},

\begin{align*}
\rho_0, \\
\text{minsync_correct} \{ i \leftarrow i + 1, \; s \leftarrow t_{(p \uplus q)[i+1]}^{t+1} \}, \\
\text{maxsync_correct} \{ i \leftarrow i + 1, \; t \leftarrow t_{(p \uplus q)[i+1]}^{t+1} \}, \\
\text{correct_closed} \{ s \leftarrow t_{(p \uplus q)[i+1]}^{t+1}, \\
t \leftarrow t_{(l@P2S \uparrow m@P2S)[i]}^{t+1}, \\
p \leftarrow (l@P2S \uparrow m@P2S)[i] \}
\end{align*}
lemma3.2.1_proof: Prove lemma3.2.1 from
lemma3.2.0,
VClock_defn \{p \leftarrow (p \vdash q)[i + 1], i \leftarrow i + 1\},
VClock_defn \{p \leftarrow (p \uplus q)[i + 1]\},
drift_bnd
\{s \leftarrow t,
   t \leftarrow t_{(p q)[i+1]}^{i+1},
   q \leftarrow (p \vDash q)[i + 1],
   p \leftarrow (p \vdash q)[i + 1],
   i \leftarrow i + 1,
   j \leftarrow i,
Y \leftarrow \alpha(X + 2 \ast (r_{\text{max}} + \beta) \ast \rho + 2 \ast \Lambda) + \Lambda\},
rho.0,
maxsync_correct \{s \leftarrow t, i \leftarrow i + 1\},
minsync_correct \{s \leftarrow t, i \leftarrow i + 1\},
correct_closed
\{p \leftarrow (p \uplus q)[i + 1],
   s \leftarrow t,
   t \leftarrow t_{(p \uplus q)[i+1]}^{i+1}\},
correct_closed \{p \leftarrow (p \vdash q)[i + 1],
   s \leftarrow t,
   t \leftarrow t_{(p \vdash q)[i+1]}^{i+1}\},
correct_closed \{s \leftarrow t, t \leftarrow t_{(p \uplus q)[i+1]}^{i+1}\},
correct_closed \{s \leftarrow t, t \leftarrow t_{(p \uplus q)[i+1]}^{i+1}\},
rt1 \{i \leftarrow i + 1, p \leftarrow (p \vdash q)[i + 1]\},
mult_leq \{z \leftarrow \rho, u \leftarrow t - t_{(p \uplus q)[i+1]}^{i+1}, x \leftarrow \beta\},
rt2 \{i \leftarrow i + 1, p \leftarrow (p \vdash q)[i + 1], q \leftarrow (p \vdash q)[i + 1]\}

lemma3.2_proof: Prove lemma3.2 from
lemma3.2.1, lemma3.1, lemma3.2_step2, lemma3.2_step3

lemma3.2_step2_proof: Prove lemma3.2_step2 from
rt2 \{p \leftarrow (p \uplus q)[i], q \leftarrow (p \vdash q)[i]\},
rt1 \{p \leftarrow (p \vdash q)[i]\},
minsync_correct \{s \leftarrow t\},
maxsync_correct \{s \leftarrow t\},
minsync_min,
correct_closed \{p \leftarrow (p \vdash q)[i], s \leftarrow t, t \leftarrow t_{(p \uplus q)[i]}^{i}\}
lemma3.2_step1_proof: Prove lemma3.2_step1 from
rts2 \{p \leftarrow (p \uparrow q)[i+1], q \leftarrow (p \downarrow q)[i+1]\},
rt1 \{p \leftarrow (p \downarrow q)[i+1]\},
minsync_correct \{s \leftarrow t, i \leftarrow i+1\},
maxsync_correct \{s \leftarrow t, i \leftarrow i+1\}

lemma3.2_step2_proof: Prove lemma3.2_step2 from
lemma3.2_step \{i \leftarrow i+1\},
lemma3.2_step1,
VClock_defn \{p \leftarrow (p \downarrow q)[i+1], i \leftarrow i+1\},
VClock_defn \{p \leftarrow (p \uparrow q)[i+1]\},
minsync_correct \{s \leftarrow t, i \leftarrow i+1\},
maxsync_correct \{s \leftarrow t, i \leftarrow i+1\}

lemma3.2_step3_proof: Prove lemma3.2_step3 from
abs_com \{x \leftarrow VC_p(t), y \leftarrow VC_q(t)\},
minsync \{p \leftarrow p, q \leftarrow q, i \leftarrow i+1\},
\{i \leftarrow i+1\} \{p \leftarrow p, q \leftarrow q, i \leftarrow i+1\}

maxmax_gap_proof: Prove maxmax_gap from
\{i \leftarrow i+1\}, \{i \leftarrow i+1\}, \{i \leftarrow i+1\}, \{i \leftarrow i+1\}

minmax_gap_proof: Prove minmax_gap from
minsync_maxsync \{i \leftarrow i+1\}, maxmax_gap

drift_bnd_proof: Prove drift_bnd from
lemma.2.2a \{i \leftarrow j\},
lemma.2.2a \{q \leftarrow p\},
lemma.2.2b \{i \leftarrow j\},
lemma.2.2b \{q \leftarrow p\},
mult_ldistrib {
\{x \leftarrow -\sum(t), y \leftarrow -1, z \leftarrow p\},
\{x \leftarrow -\sum(t), y \leftarrow -1, z \leftarrow p\},
\{x \leftarrow -\sum(t) \times l, y \leftarrow -1, z \leftarrow p\},
\{x \leftarrow -\sum(t) \times l, y \leftarrow -1, z \leftarrow p\},
\{r \leftarrow IC_i^p(t), s \leftarrow IC_i^p(t), t_1 \leftarrow IC_i^p(s), t_2 \leftarrow IC_i^p(s), y \leftarrow (s-t) \times 1, z \leftarrow (s-t) \times \rho, x \leftarrow Y\}

maxsync_max_proof: Prove maxsync_max from \{i \leftarrow j\} \cdot \{i \leftarrow j\} \cdot \{i \leftarrow j\} \cdot \{i \leftarrow j\}

minsync_min_proof: Prove minsync_min from minsync
End lemma3
lemma_final: Module

Using clockassumptions, lemma3, arith, basics

Exporting all with clockassumptions, lemma3

Theory

\( p, q, p_1, p_2, q_1, q_2, q_3, i, j, k: \) Var nat
\( l, m, n: \) Var int
\( x, y, z: \) Var number

posnumber: Type from number with ( \( \lambda x: x \geq 0 \))
\( r, s, t: \) Var posnumber

correct_synctime: Lemma correct(p, t) \( \land t < t_p^t + r_{min} \lor t < t_{p+1}^t \)

synctime_multiples: Lemma correct(p, t) \( \land t \geq 0 \land t < i \times r_{min} \lor t_p^t > t \)

synctime_multiples_bnd: Lemma correct(p, t) \( \land t \geq 0 \land t < t_p^t[1/r_{min}] + 1 \)

agreement: Lemma \( \beta \leq r_{min} \)
\( \wedge \mu \leq \delta \times (2 \times \Lambda + 2 \times \beta \times \rho, \delta \Lambda + 2 \times (r_{max} + \beta) \times \rho + \Lambda) \leq \delta \)
\( \wedge \delta \Lambda + 2 \times r_{max} \times \rho \leq \delta \)
\( \wedge \sum (\delta \Lambda + 2 \times r_{max} + \beta) \times \rho + 2 \times \Lambda) + \Lambda + 2 \times \beta \times \rho \leq \delta \)
\( \wedge t \geq 0 \land \text{correct}(p, t) \land \text{correct}(q, t) \)
\( \geq |VC_p(t) - VC_q(t)| \leq \delta \)

Proof

agreement_proof: Prove agreement from
lemma3.3 \{ i \leftarrow [t/r_{min}] + 1 \},
okayClocks_defn_lr \{ i \leftarrow [t/r_{min}] + 1, t \leftarrow t@CS \},
maxsync_correct \{ s \leftarrow t, i \leftarrow [t/r_{min}] + 1 \},
synctime_multiples_bnd \{ p \leftarrow (p \uparrow q)[[t/r_{min}] + 1] \},
rmin.0,
div_nonnegative \{ x \leftarrow t, y \leftarrow r_{min} \},
ceil_defn \{ x \leftarrow (t/r_{min}) \}

synctime_multiples_bnd_proof: Prove synctime_multiples_bnd from
ceil_plus_mult_div \{ x \leftarrow t, y \leftarrow r_{min} \},
synctime_multiples \{ i \leftarrow [t/r_{min}] + 1 \},
rmin.0,
div_nonnegative \{ x \leftarrow t, y \leftarrow r_{min} \},
ceil_defn \{ x \leftarrow (t/r_{min}) \}

correct_synctime_proof: Prove correct_synctime from rtsl \{ t \leftarrow t@CS \}
synctime_multiples_pred: function[nat, nat, posnumber → bool] ==
(λ i, p, t: correct(p, t) ∧ t ≥ 0 ∧ t < i * rmin ⊃ t_p^i > t)

synctime_multiples_step: Lemma
 correct(p, t) ∧ t ≥ t_p^i ∧ t ≥ 0 ⊃ t_p^i ≥ i * rmin

synctime_multiples_proof: Prove synctime_multiples from
 synctime_multiples_step

synctime_multiples_step_pred: function[nat, nat, posnumber → bool] ==
(λ i, p, t: correct(p, t) ∧ t ≥ 0 ∧ t > 0 ⊃ t_p^i > i * rmin)

synctime_multiples_step_proof: Prove synctime_multiples_step from
 readbounds.induction
 {prop ← (λ i: synctime_multiples_step_pred(i, p, t))},
 mult.10 {z ← rmin},
synctime_0,
 rts_1 {i ← j@P1},
 rmin_0,
 correct_closed {s ← t, t ← t_p^{j@P1+1}},
 distrib {x ← j@P1, y ← 1, z ← rmin},
 mult.lident {z ← rmin}

End lemma_final

80
lemma_final.tcc: Module

Using lemma_final

Exporting all with lemma_final

Theory

\( p: \text{Var naturalnumber} \)
\( x: \text{Var number} \)
\( j: \text{Var naturalnumber} \)
\( t: \text{Var posnumber} \)

posnumber.TCC1: Formula \((\exists x: x \geq 0)\)

synctime.multiples.bnd.TCC1: Formula \((\text{correct}(p, t) \land t \geq 0) \supset (r_{\text{min}} \neq 0)\)

synctime.multiples.bnd.TCC2: Formula
\((\text{correct}(p, t) \land t \geq 0) \supset ([t/r_{\text{min}}] + 1 \geq 0)\)

agreement.proof.TCC1: Formula \((r_{\text{min}} \neq 0)\)

agreement.proof.TCC2: Formula \([t/r_{\text{min}}] + 1 \geq 0\)

Proof

posnumber.TCC1.PROOF: Prove posnumber_TCC1

synctime.multiples.bnd.TCC1.PROOF: Prove synctime.multiples.bnd.TCC1

synctime.multiples.bnd.TCC2.PROOF: Prove synctime.multiples.bnd.TCC2

agreement.proof.TCC1.PROOF: Prove agreement.proof.TCC1

agreement.proof.TCC2.PROOF: Prove agreement.proof.TCC2

End lemma_final.tcc
ica: Module

Using arith, countmod, clockassumptions, readbounds

Exporting all with clockassumptions

Theory

process: Type is nat
event: Type is nat
time: Type is number
Clocktime: Type is number

l, m, n, p, q, p1, p2, q1, q2, p3, q3: Var process
i, j, k: Var event
x, y, z, r, s, t: Var time
X, Y, Z, R, S, T: Var Clocktime

fun, γ, θ: Var function[process → Clocktime]
ppred, ppred1, ppred2: Var function[process → bool]
sigma.size: function[function[process → Clocktime], process → process] =
( λ fun, i: i)
sigma: function[function[process → Clocktime], process → Clocktime] =
( λ fun, i: ( if i > 0 then fun(i - 1) + sigma(fun, i - 1) else 0 end if))
by sigma.size

fix: function[Clocktime, Clocktime, Clocktime → Clocktime] =
( λ X, Y, Z: ( if |Y - Z| ≤ X then Y else Z end if))
iconv: function[process, function[process → Clocktime], Clocktime → Clocktime] =
( λ p, fun, Y: sigma(( λ q: fix(Y, fun(q), fun(p))), N))
icalg: function[process, function[process → Clocktime], Clocktime → Clocktime] =
( λ p, fun, Y: icalg(p, fun(q), fun(p)), N))

ica.transl.invariance1: Lemma
iconv(p, ( λ q: fun(q) + X), Y) = iconv(p, fun, Y) + N * X

ica.transl.invariance: Lemma
N > 0 ⊃ icalg(p, ( λ q: fun(q) + X), Y) = icalg(p, fun, Y) + X

extensionality: Axiom (∀ l: ppred1(l) = ppred2(l)) ⊃ ppred1 = ppred2

fun1, fun2: Var function[process → time]

fun.extensionality: Axiom (∀ l: fun1(l) = fun2(l)) ⊃ fun1 = fun2

sigma.trans.inv: Lemma sigma(( λ q1: fun(q1) + X), n) = sigma(fun(n) + n * X

Proof
**fix_trans**: Lemma (\( \lambda q:\)
\( \text{fix}(Y, ((\lambda q_1: \text{fun}(q_1) + X)q), ((\lambda q_1: \text{fun}(q_1) + X)p)) \)
\( = (\lambda q: \text{fix}(Y, \text{fun}(q), \text{fun}(p)) + X) \)

**fix_trans_pr**: Prove fix_trans from

fun_extensionality
\{fun1 \( \leftarrow (\lambda q: \text{fix}(Y, ((\lambda q_1: \text{fun}(q_1) + X)q), ((\lambda q_1: \text{fun}(q_1) + X)p))) \},
fun2 \( \leftarrow (\lambda q: \text{fix}(Y, \text{fun}(q), \text{fun}(p)) + X) \},
fix
\{X \leftarrow Y, \)
\( Y \leftarrow ((\lambda q_1: \text{fun}(q_1) + X)l@P1S) \),
\( Z \leftarrow ((\lambda q_1: \text{fun}(q_1) + X)p) \},
fix \{X \leftarrow Y, Y \leftarrow \text{fun}(l@P1S), Z \leftarrow \text{fun}(p) \}

**sigma_trans_inv_base**: Lemma \( \sigma \text{ma}((\lambda q_1: \text{fun}(q_1) + X), 0) = \sigma \text{ma} (\text{fun}, 0) \)

**sigma_trans_inv_base_pr**: Prove sigma_trans_inv_base from
\sigma \{i \leftarrow 0\}, \sigma \{\text{fun} \leftarrow (\lambda q_1: \text{fun}(q_1) + X), i \leftarrow 0\} 

**sigma_trans_inv_ind**: Lemma
\( \sigma \text{ma}((\lambda q_1: \text{fun}(q_1) + X), j) = \sigma \text{ma} (\text{fun}, j) + j \times X \)
\( \sigma \text{ma}((\lambda q_1: \text{fun}(q_1) + X), j + 1) = \sigma \text{ma} (\text{fun}, j + 1) + (j + 1) \times X \)

**sigma_trans_inv_ind_pr**: Prove sigma_trans_inv_ind from
\sigma \{\text{fun} \leftarrow (\lambda q_1: \text{fun}(q_1) + X), i \leftarrow j + 1\},
\sigma \{i \leftarrow j + 1\},
distrib \{x \leftarrow j, y \leftarrow 1, z \leftarrow X\},
mult.\text{ident} \{x \leftarrow X\}

**sigma_trans_inv_pr**: Prove sigma_trans_inv from
induction
\{prop \leftarrow (\lambda n: \sigma \text{ma}((\lambda q_1: \text{fun}(q_1) + X), n) = \sigma \text{ma} (\text{fun}, n) + n \times X), i \leftarrow n\},
sigma_trans_inv_base,
sigma_trans_inv.ind \{j \leftarrow j@P1\},
mult.\text{l0} \{x \leftarrow X\}

**ica_translation_invariance1.pr**: Prove ica_translation_invariance1 from
iconv,
iconv \{\text{fun} \leftarrow (\lambda q: \text{fun}(q) + X)\},
fix_trans,
sigma_trans_inv \{\text{fun} \leftarrow (\lambda q: \text{fix}(Y, \text{fun}(q), \text{fun}(p))), n \leftarrow N\}

83
ica_translation_invariance.pr: Prove ica_translation_invariance from
ica_translation_invariance1,
icalg,
icalg {fun ← (λ q: fun(q) + X)},
div_distrib {z ← iconv(p, fun, Y), y ← N * X, z ← N},
div_cancel {z ← N, y ← X}

End ica
ica2: **Module**

Using arith, countmod, clockassumptions, readbounds, ica

Exporting all with ica

**Theory**

process: **Type is nat**
event: **Type is nat**
time: **Type is number**
Clocktime: **Type is number**
l, m, n, p, q, P1, P2, q1, q2, P3, q3: Var process
i, j, k: Var event
z, y, z, r, s, t: Var time
D, X, Y, Z, R, S, T: Var Clocktime
fun, fun1, fun2, γ, θ: Var function[process → Clocktime]
ppred, ppred1, ppred2: Var function[process → bool]

sigma_split: **Lemma**

\[
\sigma(f, i) = \sigma(\lambda q: (\text{if } \text{ppred}(q) \text{ then } f(q) \text{ else } 0 \text{ end if})), i)
+ \sigma(\lambda q: (\text{if } \neg\text{ppred}(q) \text{ then } f(q) \text{ else } 0 \text{ end if})), i)
\]

sigma_pos: **Lemma** okay_pairs(fun1, fun2, X, ppred)

\[\sigma((\lambda q: (\text{if } \text{ppred}(q) \text{ then } (f1(q) - f2(q)) \text{ else } 0 \text{ end if})), i)\]
\[\leq \text{count}(\text{ppred}, i) \times X\]

okay_pairs_fix: **Lemma**

\[Z \geq 0 \land \text{ppred}(p)\]
\[\land \text{ppred}(q)\]
\[\land \text{okay_pairs}(f1, f2, X, ppred)\]
\[\land \text{okay_readpred}(f1, Z, ppred) \land \text{okay_readpred}(f2, Z, ppred)\]
\[\forall \text{okay_pairs}((\lambda q1: \text{fix}(Y, f1(q1), f1(p))),\]
\[(\lambda q2: \text{fix}(Y, f2(q2), f2(q))),\]
\[(\text{if } Z \leq Y \text{ then } X \text{ else } X + Z \text{ end if}),\]
ppred)

sigma_diff: **Lemma**

\[
\sigma(f1, i) - \sigma(f2, i) = \sigma(\lambda q: f1(q) - f2(q)), i)
\]
\textbf{sigma_neg: Lemma} \( Y \geq 0 \land \text{fun1}(p) - \text{fun2}(q) \leq z \)

\[ \sigma \text{lemma}(\lambda q_1:\)
\]
\[ \begin{array}{l}
( \text{if } \neg \text{ppred}(q_1) \\
\text{then } (\text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))) \\
\text{else } 0 \\
\text{end if})
\end{array},
\]
\[ i \leq \text{count}((\lambda q_1:\neg \text{ppred}(q_1)), i) \ast (z + 2 \ast Y)
\]

\textbf{sigma_pos_neg: Lemma} \( Y > 0 \land Z > 0 \land \text{ppred}(p) \)

\[ \sigma \text{lemma}(\lambda q_1:\)
\]
\[ \begin{array}{l}
( \text{if } \neg \text{ppred}(q_1) \\
\text{then } (\text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))) \\
\text{else } 0 \\
\text{end if})
\end{array},
\]
\[ i \leq \text{count}((\lambda q_1:\neg \text{ppred}(q_1)), i) \ast (z + 2 \ast Y)
\]

\textbf{iconv_sigma_diff: Lemma} \( Y \geq 0 \land Z \geq 0 \land \text{ppred}(p) \)

\[ \sigma \text{lemma}(\lambda q_1:\)
\]
\[ \begin{array}{l}
( \text{if } \neg \text{ppred}(q_1) \\
\text{then } (\text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))) \\
\text{else } 0 \\
\text{end if})
\end{array},
\]
\[ i \leq \text{count}((\lambda q_1:\neg \text{ppred}(q_1)), i) \ast (z + 2 \ast Y)
\]

\textbf{okay_Readpred_pairs: Lemma} \( ppred(p) \land ppred(q) \)

\[ \sigma \text{lemma}(\lambda q_1:\)
\]
\[ \begin{array}{l}
( \text{if } \neg \text{ppred}(q_1) \\
\text{then } (\text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))) \\
\text{else } 0 \\
\text{end if})
\end{array},
\]
\[ i \leq \text{count}((\lambda q_1:\neg \text{ppred}(q_1)), i) \ast (z + 2 \ast Y)
\]

\textbf{okay_Readpred_lr: Lemma} \( ppred(p) \land ppred(q) \land \text{okay_Readpred}(\text{fun1}, Z, ppred) \)

\[ \sigma \text{lemma}(\lambda q_1:\)
\]
\[ \begin{array}{l}
( \text{if } \neg \text{ppred}(q_1) \\
\text{then } (\text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))) \\
\text{else } 0 \\
\text{end if})
\end{array},
\]
\[ i \leq \text{count}((\lambda q_1:\neg \text{ppred}(q_1)), i) \ast (z + 2 \ast Y)
\]

\textbf{okay_pairs_lr: Lemma} \( ppred(p) \land \text{okay_pairs}(\text{fun1}, \text{fun2}, X, ppred) \)

\[ \sigma \text{lemma}(\lambda q_1:\)
\]
\[ \begin{array}{l}
( \text{if } \neg \text{ppred}(q_1) \\
\text{then } (\text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))) \\
\text{else } 0 \\
\text{end if})
\end{array},
\]
\[ i \leq \text{count}((\lambda q_1:\neg \text{ppred}(q_1)), i) \ast (z + 2 \ast Y)
\]

\textbf{Proof}

\textbf{okay_Readpred_pairs_pr: Prove okay_Readpred_pairs from}

\textbf{okay_pairs} \{ \gamma \leftarrow \text{fun1}, \theta \leftarrow \text{fun2}, p_3 \leftarrow q \},

\textbf{abs_leq_0} \{ x \leftarrow \text{fun1}(q), y \leftarrow \text{fun2}(q), z \leftarrow X \},

\textbf{okay_Readpred} \{ \gamma \leftarrow \text{fun1}, Y \leftarrow Z, l \leftarrow p, m \leftarrow q \},

\textbf{abs_leq_0} \{ x \leftarrow \text{fun1}(p), y \leftarrow \text{fun1}(q), z \leftarrow Z \}
iconv_sigma_diff_pr: \textbf{Prove} iconv\_sigma\_diff \textbf{from} \\
\text{sigma\_pos\_neg} \\{i \leftarrow N\}, \\
\text{sigma\_diff} \\
\{\text{fun1} \leftarrow (\lambda q_1: \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p))), \\
\text{fun2} \leftarrow (\lambda q_1: \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))), \\
i \leftarrow N\}, \\
\text{iconv} \{\text{fun} \leftarrow \text{fun1}\}, \\
\text{iconv} \{p \leftarrow \text{q}, \text{fun} \leftarrow \text{fun2}\}

sigma\_pos\_neg\_pr: \textbf{Prove} sigma\_pos\_neg \textbf{from} \\
\text{sigma\_pos} \\
\{\text{fun1} \leftarrow (\lambda q_1: \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p))), \\
\text{fun2} \leftarrow (\lambda q_1: \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))), \\
X \leftarrow (\text{if } Z \leq Y \text{ then } X \text{ else } X + Z \text{ end if})\}, \\
\text{sigma\_neg} \{z \leftarrow X + Z\}, \\
\text{okay\_pairs\_fix}, \\
\text{okay\_Readpred\_pairs}, \\
\text{sigma\_split} \\
\{\text{fun} \leftarrow (\lambda q_1: \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q)))\}

fix\_diff1\_pr: \textbf{Prove} fix\_diff1 \textbf{from} \\
\text{fix} \{X \leftarrow Y, Y \leftarrow \text{fun1}(p_3), Z \leftarrow \text{fun1}(p)\}, \\
\text{abs\_drift} \\
\{z_1 \leftarrow \text{fun1}(p), \\
y \leftarrow \text{fun2}(p_3), \\
z \leftarrow \text{fun1}(p_3), \\
z \leftarrow X, \\
z_1 \leftarrow Z\}, \\
\text{abs\_com} \{z \leftarrow \text{fun1}(p), y \leftarrow \text{fun1}(p_3)\}

fix\_diff2\_pr: \textbf{Prove} fix\_diff2 \textbf{from} \\
\text{abs\_drift} \\
\{z_1 \leftarrow \text{fun1}(p_3), \\
y \leftarrow \text{fun2}(q), \\
z \leftarrow \text{fun2}(p_3), \\
z_1 \leftarrow X, \\
z \leftarrow Z\}
\textbf{fix_diff3: Lemma} \(|\text{fun1}(q) - \text{fun2}(q)| \leq X \land |\text{fun1}(p) - \text{fun1}(q)| \leq Z \land |\text{fun1}(p) - \text{fun2}(q)| \leq X + Z\)

\textbf{fix_diff3_pr: Prove} \text{fix_diff3} \text{ from}  \\
\textbf{abs_drift}  \\
\{z_1 \leftarrow \text{fun1}(p), \}  \\
y \leftarrow \text{fun2}(q), \}  \\
x \leftarrow \text{fun1}(q), \}  \\
z_1 \leftarrow Z, \}  \\
z \leftarrow X\)

\textbf{fix_diff: Lemma} \(Z \geq 0\)  \\
\(\land |\text{fun1}(p_3) - \text{fun2}(p_3)| \leq X\)  \\
\(\land |\text{fun1}(q) - \text{fun2}(q)| \leq X\)  \\
\(\land |\text{fun1}(p_3) - \text{fun1}(p)| \leq Z\)  \\
\(\land |\text{fun2}(p_3) - \text{fun2}(q)| \leq Z \land |\text{fun1}(p) - \text{fun1}(q)| \leq Z\)  \\
\(\supset |\text{fix}(Y, \text{fun1}(p_3), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(p_3), \text{fun2}(q))| \leq (\text{if } Z \leq Y \text{ then } X \text{ else } X + Z \text{ end if})\)

\textbf{fix_diff_pr: Prove} \text{fix_diff} \text{ from}  \\
f \{X \leftarrow Y, Y \leftarrow \text{fun1}(p_3), Z \leftarrow \text{fun1}(p)\}, \}  \\
f \{X \leftarrow Y, Y \leftarrow \text{fun2}(p_3), Z \leftarrow \text{fun2}(q)\}, \}  \\
\text{fix_diff}\_1, \}  \\
\text{fix_diff}\_2, \}  \\
\text{fix_diff}\_3

\textbf{okay_pairs_lr_pr: Prove} \text{okay_pairs}_\text{lr} \text{ from}  \\
\text{okay_pairs} \{\gamma \leftarrow \text{fun1}, \theta \leftarrow \text{fun2}, p_3 \leftarrow p\}, \}

\textbf{okay_Readpred_lr_pr: Prove} \text{okay_Readpred}_\text{lr} \text{ from}  \\
\text{okay_Readpred} \{\gamma \leftarrow \text{fun1}, Y \leftarrow Z, l \leftarrow p, m \leftarrow q\}, \}

\textbf{fix_diff_corr: Lemma}  \\
\(Z \geq 0 \land \text{ppred}(p)\)  \\
\(\land \text{ppred}(q)\)  \\
\(\land \text{ppred}(p_3)\)  \\
\(\land \text{okay_pairs}(\text{fun1}, \text{fun2}, X, \text{ppred})\)  \\
\(\land \text{okay_Readpred}(\text{fun1}, Z, \text{ppred}) \land \text{okay_Readpred}(\text{fun2}, Z, \text{ppred})\)  \\
\(\supset |\text{fix}(Y, \text{fun1}(p_3), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(p_3), \text{fun2}(q))| \leq (\text{if } Z \leq Y \text{ then } X \text{ else } X + Z \text{ end if})\)

88
fix_diff_corr_pr: Prove fix_diff_corr from
fix_diff,
okay_pairs_lr \{p \leftarrow p_3\},
okay_pairs_lr \{p \leftarrow q\},
okay_Readpred_lr \{p \leftarrow p_3, q \leftarrow p\},
okay_Readpred_lr \{\text{fun1} \leftarrow \text{fun2}, p \leftarrow p_3\},
okay_Readpred_lr

okay_pairs_fix_pr: Prove okay_pairs_fix from
okay_pairs
\{\gamma \leftarrow (\lambda q_1: \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p))),
\theta \leftarrow (\lambda q_1: \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q))),
X \leftarrow (\text{if } Z \leq Y \text{ then } X \text{ else } X + Z \text{ end if})\},
fix_diff_corr \{p_3 \leftarrow p_3@PS\}

sigma_neg_ind_step: Lemma
\begin{align*}
\gamma &\geq 0 \land \text{fun1}(p) - \text{fun2}(q) \leq z \\
\gamma &\geq \text{fix}(Y, \text{fun1}(i), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(i), \text{fun2}(q)) \leq z + 2 \ast Y
\end{align*}

sigma_neg_ind_step_pr: Prove sigma_neg_ind_step from
fix \{X \leftarrow Y, Y \leftarrow \text{fun1}(i), Z \leftarrow \text{fun1}(p)\},
fix \{X \leftarrow Y, Y \leftarrow \text{fun2}(i), Z \leftarrow \text{fun2}(q)\},
abs_leq_0 \{x \leftarrow \text{fun1}(i), y \leftarrow \text{fun1}(p), z \leftarrow Y\},
abs_com \{x \leftarrow \text{fun2}(i), y \leftarrow \text{fun2}(q)\},
abs_leq_0 \{x \leftarrow \text{fun2}(q), y \leftarrow \text{fun2}(i), z \leftarrow Y\}

sigma_neg_ind: Lemma
\begin{align*}
\gamma &\geq 0 \land \text{fun1}(p) - \text{fun2}(q) \leq z \\
\forall \sigma((\lambda q_1:\ (\text{if } \neg \text{ppred}(q_1) \text{ then } \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) \text{ else } 0) \end{align*}
\begin{align*}
&\neg \text{ppred}(q_1)) \text{ i) } \ast (z + 2 \ast Y) \\
&\forall \sigma((\lambda q_1:\ (\text{if } \neg \text{ppred}(q_1) \text{ then } \text{fix}(Y, \text{fun1}(q_1), \text{fun1}(p)) - \text{fix}(Y, \text{fun2}(q_1), \text{fun2}(q)) \text{ else } 0) \end{align*}
\begin{align*}
&\neg \text{ppred}(q_1)) \text{ i) + 1} \ast (z + 2 \ast Y)
\end{align*}

\leq \text{count}((\lambda q_1:\ \neg \text{ppred}(q_1)), i + 1) \ast (z + 2 \ast Y)
**sigma_neg_ind.pr**  
Prove `sigma_neg_ind` from

```plaintext
sigma
{fun ← (λ q1:
  (if ~ppred(q1)
      then fix(Y, fun1(q1), fun1(p)) − fix(Y, fun2(q1), fun2(q))
      else 0
      end if),
i ← i + 1),
count {ppred ← (λ q1: ~ppred(q1)), i ← i + 1},
sigma_neg_ind_step,
distrib
{z ← 1,
y ← count((λ q1: ~ppred(q1)), i),
z ← z + 2 * Y},
mult.10 {x ← z + 2 * Y}
```

**sigma_neg.pr**  
Prove `sigma_neg` from induction

```plaintext
{prop ← (λ i:
  Y ≥ 0 ∧ fun1(p) − fun2(q) ≤ z
  ⊃ sigma((λ q1:
    if ~ppred(q1)
      then fix(Y, fun1(q1), fun1(p)) − fix(Y, fun2(q1), fun2(q))
      else 0
      end if),
i) ≤ count((λ q1: ~ppred(q1)), i) * (z + 2 * Y))},
sigma
{fun ← (λ q1:
  (if ~ppred(q1)
      then fix(Y, fun1(q1), fun1(p)) − fix(Y, fun2(q1), fun2(q))
      else 0
      end if),
i ← 0),
count {i ← 0, ppred ← (λ q1: ~ppred(q1))},
mult.10 {x ← z + 2 * Y},
sigma_neg_ind {i ← j@P1S}
```

**sigma_diff.ind**  
Lemma

```plaintext
sigma(fun1, i) − sigma(fun2, i) = sigma((λ q: fun1(q) − fun2(q)), i)
⊃ sigma(fun1, i + 1) − sigma(fun2, i + 1)
= sigma((λ q: fun1(q) − fun2(q)), i + 1)
```
Prove \( \sigma_{\text{diff\_ind}} \) from

\[ \sigma \{ \text{fun} \leftarrow \text{fun1}, \ i \leftarrow i + 1 \}, \]
\[ \sigma \{ \text{fun} \leftarrow \text{fun2}, i \leftarrow i + 1 \}, \]
\[ \sigma \{ \text{fun} \leftarrow (\lambda q:\text{fun1}(q) - \text{fun2}(q)), i \leftarrow i + 1 \} \]

Prove \( \sigma_{\text{diff}} \) from

induction

\[ \text{prop} \leftarrow (\lambda i:\]
\[ \sigma(\text{fun1}, i) - \sigma(\text{fun2}, i)
\[ = \sigma((\lambda q:\text{fun1}(q) - \text{fun2}(q)), i)) \}, \]
\[ \sigma \{ \text{fun} \leftarrow \text{fun1}, i \leftarrow 0 \}, \]
\[ \sigma \{ \text{fun} \leftarrow \text{fun2}, i \leftarrow 0 \}, \]
\[ \sigma \{ \text{fun} \leftarrow (\lambda q:\text{fun1}(q) - \text{fun2}(q)), i \leftarrow 0 \}, \]
\[ \sigma_{\text{diff\_ind}} \{ i \leftarrow j@P1S \} \]

sigma_pos_ind: Lemma

\begin{align*}
\text{okay\_pairs}(\text{fun1}, \text{fun2}, X, \text{ppred}) \\
\wedge \sigma((\lambda q:\text{if ppred(q) then (fun1(q) - fun2(q)) else 0 end if)), i) \\
\leq \text{count(ppred, i)} * X \\
\supset \sigma((\lambda q:\text{if ppred(q) then (fun1(q) - fun2(q)) else 0 end if)), i + 1) \\
\leq \text{count(ppred, i + 1)} * X
\end{align*}

Prove \( \sigma_{\text{pos\_ind}} \) from

\[ \text{sigma} \{ \text{fun} \leftarrow (\lambda q:\text{if ppred(q) then (fun1(q) - fun2(q)) else 0 end if)}, i \leftarrow i + 1 \}, \]
\[ \text{okay\_pairs} \{ \gamma \leftarrow \text{fun1}, \ \theta \leftarrow \text{fun2}, p_3 \leftarrow i \}, \]
\[ \text{count} \{ i \leftarrow i + 1 \}, \]
\[ \text{distrib} \{ x \leftarrow 1, y \leftarrow \text{count(ppred, i)}, z \leftarrow X \}, \]
\[ \text{mul\_lident} \{ x \leftarrow X \}, \]
\[ \text{abs\_leq\_0} \{ x \leftarrow \text{fun1}(i), y \leftarrow \text{fun2}(i), z \leftarrow X \} \]
sigma.pos.pr: Prove sigma.pos from induction

{prop ← (λ i:
       okay_pairs(fun1, fun2, X, ppred)
       ⊃ sigma((λ q:
                   (if ppred(q) then (fun1(q) - fun2(q)) else 0 end if)),
                   i) ≤ count(ppred, i) * X)},

sigma

{fun ← (λ q:( if ppred(q) then (fun1(q) - fun2(q)) else 0 end if)),
       i ← 0},

count {i ← 0},

mult 10 {x ← X},

sigma.pos.ind {i ← j@P1S}

sigma.split.ind: Lemma

sigma(fun, i) = sigma((λ q:( if ppred(q) then fun(q) else 0 end if)), i)
       + sigma((λ q:( if ~ppred(q) then fun(q) else 0 end if)), i)
       ⊃ sigma(fun, i + 1)
       = sigma((λ q:( if ppred(q) then fun(q) else 0 end if)), i + 1)
       + sigma((λ q:( if ~ppred(q) then fun(q) else 0 end if)), i + 1)

sigma.split.ind.pr: Prove sigma.split.ind from sigma

{fun ← (λ q:( if ppred(q) then fun(q) else 0 end if)),
       i ← i + 1},

sigma

{fun ← (λ q:( if ppred(q) then fun(q) else 0 end if)),
       i ← i + 1}

sigma.split.pr: Prove sigma.split from induction

{prop ← (λ i:
         sigma(fun, i)
         = sigma((λ q:( if ppred(q) then fun(q) else 0 end if)), i)
         + sigma((λ q:( if ~ppred(q) then fun(q) else 0 end if)), i))},

sigma {i ← 0},

sigma

{fun ← (λ q:( if ppred(q) then fun(q) else 0 end if)),
       i ← 0},

sigma.split.ind {i ← j@P1S}

92
End ica2
ica3: Module

Using arith, countmod, clockassumptions, readbounds, ica, ica2

Exporting all with clockassumptions, ica2

Theory

process: Type is nat
event: Type is nat
time: Type is number
Clocktime: Type is number
l, m, n, p, q, p1, p2, q1, q2, p3, q3: Var process
i, j, k: Var event
x, y, z, r, s, t: Var time
D, X, Y, Z, R, S, T: Var Clocktime
fun, fun1, fun2, γ, θ: Var function[process → Clocktime]
ppred, ppred1, ppred2: Var function[process → bool]
Δ: Clocktime

Delta_0: Axiom Δ ≥ 0

mult_sum_ineq: Lemma
m + n = p + q ∧ n ≤ q ∧ x ≤ y ∨ m * x + n * y ≤ p * x + q * y

count_complement: Lemma count((λ q: ¬ppred(q)), n) = n - count(ppred, n)

prec_enh_step3: Lemma
count(ppred, N) ≥ N - maxfaults ∧ X ≥ 0 ∧ Y ≥ 0 ∧ Z ≥ 0
∨ count(ppred, N) * ( if Z ≤ Y then X else X + Z end if)
+ count((λ q1: ¬ppred(q1)), N) * (X + Z + 2 * Y)
≤ N - maxfaults * ( if Z ≤ Y then X else X + Z end if)
+ maxfaults * (X + Z + 2 * Y)

icalg.Pi: function[Clocktime, Clocktime → Clocktime] =
(λ X, Z: (N - maxfaults * ( if Z ≤ Δ then X else X + Z end if)
+ maxfaults * (X + Z + 2 * Δ))
/N)

prec_enh_step: Lemma
ppred(p) ∧ ppred(q) ∧ okay_Readpred(fun1, Z, ppred) ∨ Z ≥ 0

prec_enh_step2: Lemma ppred(p) ∧ okay_pairs(fun1, fun2, X, ppred) ∨ X ≥ 0

94
icalg_precision_enhancement_step: Lemma
ppred(p) ∧ ppred(q)
   ∧ count(ppred, N) ≥ N – maxfaults
   ∧ okay_pairs(fun1, fun2, X, ppred)
   ∧ okay_Readpred(fun1, Z, ppred) ∧ okay_Readpred(fun2, Z, ppred)
   ⊢ icalg(p, fun1, Δ) – icalg(q, fun2, Δ)
   ≤ (count(ppred, N) * (if Z ≤ Δ then X else X + Z end if) + count((λ q1: ¬ppred(q1)), N) * (X + Z + 2 * Δ)) / N

icalg_Mu: function[Clocktime, Clocktime, function[process → bool]
   → Clocktime] =
   (λ X, Z, ppred:
     (count(ppred, N) * (if Z ≤ Δ then X else X + Z end if) + count((λ q1: ¬ppred(q1)), N) * (X + Z + 2 * Δ)) / N
   )

icalg_precision_enhancement: Lemma
ppred(p) ∧ ppred(q)
   ∧ count(ppred, N) ≥ N – maxfaults
   ∧ okay_pairs(fun1, fun2, X, ppred)
   ∧ okay_Readpred(fun1, Z, ppred) ∧ okay_Readpred(fun2, Z, ppred)
   ⊢ icalg(p, fun1, Δ) – icalg(q, fun2, Δ) ≤ icalg_Pi(X, Z)

Proof
prec_enh_step4: Lemma
N > 0 ∧ ppred(p)
   ∧ ppred(q)
   ∧ count(ppred, N) ≥ N – maxfaults
   ∧ okay_pairs(fun1, fun2, X, ppred)
   ∧ okay_Readpred(fun1, Z, ppred) ∧ okay_Readpred(fun2, Z, ppred)
   ⊢ icalg_Mu(X, Z, ppred) ≤ icalg_Pi(X, Z)
prec_enh_step4_pr: Prove prec_enh_step4 from
prec_enh_step,
prec_enh_step2,
prec_enh_step3 \{Y \leftarrow \Delta \},
\Delta_0,
icalg_Pi,
icalg_Mu,
div_ineq
\{x \leftarrow \text{count}(p, N) \ast (\text{if } Z \leq \Delta \text{ then } X \text{ else } X + Z \text{ end if})
+ \text{count}(\lambda q_1: \neg \text{pred}(q_1), N) \ast (X + Z + 2 \ast \Delta),
  \quad y \leftarrow (N - \text{maxfaults}) \ast (\text{if } Z \leq \Delta \text{ then } X \text{ else } X + Z \text{ end if})
+ \text{maxfaults} \ast (X + Z + 2 \ast \Delta),
  \quad z \leftarrow N \}\}

icalg_precision_enhancement_pr: Prove icalg_precision_enhancement from
prec_enh_step4, N_0, icalg.precision_enhancement_step, icalg_Mu

icalg_precision_enhancement_step_pr: Prove icalg_precision_enhancement_step from
prec_enh_step,
prec_enh_step2,
iconv_sigma.diff \{Y \leftarrow \Delta \},
N_0,
icalg \{\text{fun} \leftarrow \text{fun1}, Y \leftarrow \Delta \},
icalg \{p \leftarrow q, \text{fun} \leftarrow \text{fun2}, Y \leftarrow \Delta \},
div_minus_distrib
\{x \leftarrow \text{iconv}(p, \text{fun1}, \Delta),
  \quad y \leftarrow \text{iconv}(q, \text{fun2}, \Delta),
  \quad z \leftarrow N \}\},
\Delta_0,
div_ineq
\{x \leftarrow \text{iconv}(p, \text{fun1}, \Delta) - \text{iconv}(q, \text{fun2}, \Delta),
  \quad y \leftarrow \text{count}(p, N) \ast (\text{if } Z \leq \Delta \text{ then } X \text{ else } X + Z \text{ end if})
+ \text{count}(\lambda q_1: \neg \text{pred}(q_1), N) \ast (X + Z + 2 \ast \Delta),
  \quad z \leftarrow N \}\}

prec_enh_step3_pr: Prove prec_enh_step3 from
count_complement \{n \leftarrow N \},
mult_sum.ineq
\{m \leftarrow \text{count}(p, N),
  \quad n \leftarrow \text{count}(\lambda q: \neg \text{pred}(q), N),
  \quad p \leftarrow N - \text{maxfaults},
  \quad q \leftarrow \text{maxfaults},
  \quad x \leftarrow (\text{if } Z \leq Y \text{ then } X \text{ else } X + Z \text{ end if}),
  \quad y \leftarrow X + Z + 2 \ast Y \}\}
prec_enh_step2_pr: Prove prec_enh_step2 from
   okay_pairs_lr, |∗1| {x ← fun1(p) − fun2(p)}

count_complement_pr: Prove count_complement from
   induction
   {prop ← (λ n: count(λ q: ¬ppred(q)), n) = n − count(ppred, n)),
    i ← n},
   count {ppred ← (λ q: ¬ppred(q)), i ← 0},
   count {i ← 0},
   count {ppred ← (λ q: ¬ppred(q)), i ← j@P1S + 1},
   count {i ← j@P1S + 1}

mult_sum_ineq_pr: Prove mult_sum_ineq from
   distrib {z ← n, y ← q − n, z ← y},
   distrib {x ← p, y ← m − p, z ← x},
   mult_leq_2 {z ← q − n, x ← y, y ← x}

prec_enh_step_pr: Prove prec_enh_step from
   okay_Readpred_lr, |∗1| {x ← fun1(p) − fun1(q)}

End ica3
ica4: Module

Using arith, countmod, clockassumptions, readbounds, ica, ica2, ica3

Exporting all with clockassumptions, ica3

Theory

process: Type is nat
event: Type is nat
time: Type is number
Clocktime: Type is number
l, m, n, p, q, p1, p2, q1, q2, p3, q3: Var process
i, j, k: Var event
x, y, z, r, s: Var time
D, X, Y, Z, R, S, T: Var Clocktime
fun, fun1, fun2: Var function[process → Clocktime]
ppred, ppred1, ppred2: Var function[process → bool]

sigma_duplicate: Lemma sigma(( λ i: x), i) = i * x

okay_Readpred_fix_diff: Lemma
ppred(p) A ppred(q) A ppred(p1) A okay_Readpred(fun, X, ppred)
⇒ |fix(Y, fun(p1), fun(p)) - fun(q)| ≤ X

okay_Readpred_fix_diff2: Lemma
ppred(p) A ppred(q) A okay_Readpred(fun, X, ppred) ∧ Y ≥ 0
⇒ |fix(Y, fun(p1), fun(p)) - fun(q)| ≤ X + Y

acc_pres_sigma_pos: Lemma
ppred(p) A ppred(q) A okay_Readpred(fun, X, ppred)
⇒ sigma(( λ p1:
          ( if ppred(p1)
              then |fix(Y, fun(p1), fun(p)) - fun(q)|
              else 0
          end if)),
          N) ≤ count(ppred, N) * X

acc_pres_sigma_neg: Lemma
ppred(p) A ppred(q) A okay_Readpred(fun, X, ppred) ∧ Y ≥ 0
⇒ sigma(( λ p1:
          ( if ¬ppred(p1)
              then |fix(Y, fun(p1), fun(p)) - fun(q)|
              else 0
          end if)),
          N) ≤ count(( λ p1: ¬ppred(p1)), N) * (X + Y)
sigma_abs: **Lemma** \(|\sigma_a (fun, i)| \leq \sigma_a (\lambda p: [fun(p)], i)\)

acc_pres_step: **Lemma**

\[
ppred(p) \land ppred(q) \land \text{okay_Readpred}(fun, X, ppred) \\
\supset |\text{iconv}(p, fun, \Delta) - N \times \text{fun}(q)| \\
\leq \text{count}(ppred, N) \times X + \text{count}((\lambda p: \neg ppred(p)), N) \times (X + \Delta)
\]

icalg_accuracy_preservation: **Lemma**

\[
ppred(p) \land ppred(q) \\
\land \text{count}(ppred, N) \geq N - \text{maxfaults} \land \text{okay_Readpred}(fun, X, ppred) \\
\supset |\text{icalg}(p, fun, \Delta) - \text{fun}(q)| \\
\leq ((N - \text{maxfaults}) \times X + \text{maxfaults} \times (X + \Delta))/N
\]

**Proof**

icalg_accuracy_preservation_pr: **Prove** icalg_accuracy_preservation from acc_pres_step,
N_0,
abs_div \{z \leftarrow \text{iconv}(p, fun, \Delta) - N \times \text{fun}(q), y \leftarrow N\},
icalg \{Y \leftarrow \Delta\},
div_cancel \{x \leftarrow N, y \leftarrow \text{fun}(q)\},
mult_sum_ineq
\{m \leftarrow \text{count}(ppred, N), \\
n \leftarrow \text{count}((\lambda p: \neg ppred(p)), N), \\
p \leftarrow N - \text{maxfaults}, \\
g \leftarrow \text{maxfaults}, \\
x \leftarrow X, \\
y \leftarrow X + \Delta\},
Delta_0,
count_complement \{n \leftarrow N\},
div_minus_distrib \{z \leftarrow N, x \leftarrow \text{iconv}(p, fun, \Delta), y \leftarrow N \times \text{fun}(q)\},
div_ineq
\{z \leftarrow N, \\
x \leftarrow |\text{iconv}(p, fun, \Delta) - N \times \text{fun}(q)|, \\
y \leftarrow (N - \text{maxfaults}) \times X + \text{maxfaults} \times (X + \Delta)\}
acc_pres_step_pr: Prove acc_pres_step from
sigma_split
{fun ← (λ p₁: fix(Δ, fun(p₁), fun(p)) − fun(q)),
i ← N},
sigma_abs {fun ← (λ p₁: fix(Δ, fun(p₁), fun(p)) − fun(q)), i ← N},
sigma_diff
{fun₁ ← (λ p₁: fix(Δ, fun(p₁), fun(p))),
fun₂ ← (λ p₁: fun(q)),
i ← N},
acc_pres_sigma_neg {Y ← Δ},
acc_pres_sigma_pos {Y ← Δ},
iconv {Y ← Δ},
sigma_duplicate {z ← fun(q), i ← N},
Delta_0

sigma_abs_pr: Prove sigma_abs from
induction {prop ← (λ i: [sigma(fun, i)] ≤ sigma((λ p: [fun(p)]), i))},
sigma {i ← 0},
|·| {z ← 0},
sigma {i ← 0, fun ← (λ p: [fun(p)])},
sigma {i ← j@P1S + 1},
sigma {i ← j@P1S + 1, fun ← (λ p: [fun(p)])},
abs_plus {x ← sigma(fun, j@P1S), y ← fun(j@P1S)}

acc_pres_sigma_neg_pr: Prove acc_pres_sigma_neg from
sigma_pos
{i ← N,
fun₁ ← (λ p₁: [fix(Y, fun(p₁), fun(p)) − fun(q))},
fun₂ ← (λ p₁ → number: 0),
ppred ← (λ p₁: ¬ppred(p₁)),
X ← X + Y},
okay_pairs
{γ ← (λ p₁: [fix(Y, fun(p₁), fun(p)) − fun(q))},
θ ← (λ p₁ → number: 0),
X ← X + Y,
ppred ← (λ p₁: ¬ppred(p₁))},
okay_Readpred_fix_diff2 {p₁ ← p₃@P2S},
|·| {z ← [fix(Y, fun(p₃@P2S), fun(p)) − fun(q)]},
|·| {z ← fix(Y, fun(p₃@P2S), fun(p)) − fun(q)}
acc_pres_sigma_pos_pr: Prove acc_pres_sigma_pos from sigma_pos

{\{i \leftarrow N, \\
    fun1 \leftarrow (\lambda p_1: fix(Y, fun(p_1), fun(p)) - fun(q)), \\
    fun2 \leftarrow (\lambda p_1 \rightarrow number: 0)\},

okay_pairs

{\{y \leftarrow (\lambda p_1: fix(Y, fun(p_1), fun(p)) - fun(q)), \\
    \theta \leftarrow (\lambda p_1 \rightarrow number: 0)\},

okay_Readpred_fix_diff \{ p_1 \leftarrow p_3 @ P2S \},

\{ x \leftarrow \{x \leftarrow fix(Y, fun(p_3 @ P2S), fun(p)) - fun(q)\}, \\
    \star 1 | \{ x \leftarrow fix(Y, fun(p_3 @ P2S), fun(p)) - fun(q)\}\}

okay_Readpred_fix_diff2_pr: Prove okay_Readpred_fix_diff2 from

okay_Readpred_lr \{ fun1 \leftarrow fun, Z \leftarrow X \},

{\{x \leftarrow Y, Y \leftarrow fun(p_1), Z \leftarrow fun(p)\},

abs_drift

\{z_1 \leftarrow fun(p_1), \\
    y \leftarrow fun(q), \\
    x \leftarrow fun(p), \\
    z \leftarrow X, \\
    z_1 \leftarrow Y\}

okay_Readpred_fix_diff_pr: Prove okay_Readpred_fix_diff from

okay_Readpred_lr \{ fun1 \leftarrow fun, Z \leftarrow X \},

okay_Readpred_lr \{ fun1 \leftarrow fun, p \leftarrow p_1, Z \leftarrow X \},

{\{x \leftarrow Y, Y \leftarrow fun(p_1), Z \leftarrow fun(p)\}

sigma_duplicate_pr: Prove sigma_duplicate from

induction \{ prop \leftarrow (\lambda i: sigma((\lambda i: z), i = i * x))\},

sigma \{i \leftarrow 0, fun \leftarrow (\lambda i: x)\},

\star 1 \star 2 \{x \leftarrow 0, y \leftarrow x\},

sigma \{i \leftarrow j @ P1S, fun \leftarrow (\lambda i: x)\},

sigma \{i \leftarrow j @ P1S + 1, fun \leftarrow (\lambda i: x)\},

distrib \{x \leftarrow j @ P1S, y \leftarrow 1, z \leftarrow z\},

\star 1 \star 2 \{x \leftarrow 1, y \leftarrow x\}

End ica4
ica_tcc: Module

Using ica

Exporting all with ica

Theory

\( i: \text{Var} \text{ naturalnumber} \)
\( \text{fun}: \text{Var} \text{ function[naturalnumber} \rightarrow \text{number]} \)
\( j: \text{Var} \text{ naturalnumber} \)
\( l: \text{Var} \text{ naturalnumber} \)

\( \sigma_{\text{TCC1}}: \text{Formula} (i > 0) \supset (i - 1 \geq 0) \)

\( \sigma_{\text{TCC2}}: \text{Formula} (i > 0) \supset \sigma_{\text{size}}(\text{fun}, i) > \sigma_{\text{size}}(\text{fun}, i - 1) \)

\( \sigma_{\text{calg.TCC1}}: \text{Formula} (N \neq 0) \)

Proof

\( \sigma_{\text{TCC1.PROOF}}: \text{Prove } \sigma_{\text{TCC1}} \)

\( \sigma_{\text{TCC2.PROOF}}: \text{Prove } \sigma_{\text{TCC2}} \)

\( \sigma_{\text{calg.TCC1.PROOF}}: \text{Prove } \sigma_{\text{calg.TCC1}} \)

End ica_tcc
ica4_tcc: Module

Using ica4

Exporting all with ica4

Theory

\( p \): Var naturalnumber  
\( q \): Var naturalnumber  
\( X \): Var number  
fun: Var function[naturalnumber \to number]  
ppred: Var function[naturalnumber \to boolean]  
\( p_3 \): Var naturalnumber  
\( j \): Var naturalnumber

icalg_accuracy_preservation_TCC1: Formula

\[
(ppred(p) \land ppred(q) \land \text{count}(ppred, N) \geq N - \text{maxfaults} \land \text{okay}\_\text{Readpred}(\text{fun}, X, ppred)) \\
\lor (N \neq 0)
\]

icalg_accuracy_preservation_pr_TCC1: Formula \( (N - \text{maxfaults} \geq 0) \)

Proof

icalg_accuracy_preservation_TCC1_PROOF: Prove  
icalg_accuracy_preservation_TCC1

icalg_accuracy_preservation_pr_TCC1_PROOF: Prove  
icalg_accuracy_preservation_pr_TCC1

End ica4_tcc
ica3.tcc: Module

Using ica3

Exporting all with ica3

Theory

\[ \begin{align*}
  p &: \text{Var naturalnumber} \\
  q &: \text{Var naturalnumber} \\
  X &: \text{Var number} \\
  Z &: \text{Var number} \\
  \text{fun1} &: \text{Var function[naturalnumber \to number]} \\
  \text{fun2} &: \text{Var function[naturalnumber \to number]} \\
  \text{ppred} &: \text{Var function[naturalnumber \to boolean]} \\
  j &: \text{Var naturalnumber}
\end{align*} \]

icalg_Pi.TCC1: Formula \((N \neq 0)\)

icalg_precision_enhancement_step.TCC1: Formula
\[ \begin{align*}
  \text{ppred}(p) \land \text{ppred}(q) \\
  \land \text{count(ppred, N)} \geq N - \text{maxfaults} \\
  \land \text{okay_pairs}(\text{fun1}, \text{fun2}, X, \text{ppred}) \\
  \land \text{okay_Readpred}(\text{fun1}, Z, \text{ppred}) \\
  \land \text{okay_Readpred}(\text{fun2}, Z, \text{ppred})
\end{align*} \]
\[ \Rightarrow (N \neq 0) \]

prec_enh_step3_pr.TCC1: Formula \((N - \text{maxfaults} \geq 0)\)

Proof

icalg_Pi.TCC1.PROOF: Prove icalg_Pi.TCC1

icalg_precision_enhancement_step.TCC1.PROOF: Prove
icalg_precision_enhancement_step.TCC1

prec_enh_step3_pr.TCC1.PROOF: Prove prec_enh_step3_pr.TCC1

End ica3.tcc
tcc_proofs: Module

Using countmod_tcc, lemma_final_tcc, division, clockassumptions, ica_tcc,
ica4_tcc, ica3_tcc

Exporting all
with countmod_tcc, lemma_final_tcc, division, clockassumptions, ica_tcc,
ica4_tcc, ica3_tcc

Proof

countmod_TCC4.pr: Prove count_TCC4 from
  countsize, countsize \{i \leftarrow (\text{if } i > 0 \text{ then } i - 1 \text{ else } i \text{ end if})\}

countmod_TCC5.pr: Prove count_TCC5 from
  countsize, countsize \{i \leftarrow (\text{if } i > 0 \text{ then } i - 1 \text{ else } i \text{ end if})\}

posnumber_TCC1.PROOF: Prove posnumber_TCC1 \{x \leftarrow 0\}

syntime_multiples_bnd_TCC1.PROOF: Prove syntime_multiples_bnd_TCC1 from
  rmin_0

syntime_multiples_bnd_TCC2.PROOF: Prove syntime_multiples_bnd_TCC2 from
  div_nonnegative \{x \leftarrow t, y \leftarrow r_{\text{min}}\}, rmin_0, ceil_defn \{x \leftarrow t/r_{\text{min}}\}

agreement_proof_TCC1.PROOF: Prove agreement_proof_TCC1 from rmin_0

agreement_proof_TCC2.PROOF: Prove agreement_proof_TCC2 from
  div_nonnegative \{x \leftarrow t, y \leftarrow r_{\text{min}}\}, rmin_0, ceil_defn \{x \leftarrow t/r_{\text{min}}\}

sigma_TCC2.PROOF: Prove sigma_TCC2 from
  sigma_size, sigma_size \{i \leftarrow (\text{if } i > 0 \text{ then } i - 1 \text{ else } 0 \text{ end if})\}

icalg_TCC1.PROOF: Prove icalg_TCC1 from N_0

icalg_Pi_TCC1.PROOF: Prove icalg_Pi_TCC1 from N_0

icalg_precision_enhancement_step_TCC1.PROOF: Prove
  icalg_precision_enhancement_step_TCC1 from N_0

prec_enh_step3_pr_TCC1.PROOF: Prove prec_enh_step3_pr_TCC1 from N_maxfaults

icalg_accuracy_preservation_TCC1.PROOF: Prove
  icalg_accuracy_preservation_TCC1 from N_0

icalg_accuracy_preservation_pr_TCC1.PROOF: Prove
  icalg_accuracy_preservation_pr_TCC1 from N_maxfaults

End tcc_proofs

105
tcc_proofs.tcc: Module

Using tcc_proofs

Exporting all with tcc_proofs

Theory

t: Var lemma_final.posnumber
i: Var naturalnumber

countmod.TCC4.pr.TCC1: Formula (( if i > 0 then i - 1 else i end if) > 0)
synctime_multiples.bnd.TCC2.PROOF.TCC1: Formula (r_min ≠ 0)
sigma.TCC2.PROOF.TCC1: Formula (( if i > 0 then i - 1 else 0 end if) ≥ 0)

Proof

countmod.TCC4.pr.TCC1.PROOF: Prove countmod.TCC4.pr.TCC1

synctime_multiples.bnd.TCC2.PROOF.TCC1.PROOF: Prove
synctime_multiples.bnd.TCC2.PROOF.TCC1

sigma.TCC2.PROOF.TCC1.PROOF: Prove sigma.TCC2.PROOF.TCC1

End tcc_proofs.tcc
top: Module

Using arith, lemma_final, ica4, tcc_proofs, tcc_proofs_tcc, division_tcc

Theory

Proof

synctime_multiples_bnd.TCC2.PROOF.TCC1: Prove
  synctime_multiples_bnd.TCC2.PROOF.TCC1 from rmin.0

End top
Appendix C

Proof Chain Analysis

The dependency analysis automatically establishes that there are no unproved statements in the proof that are not axioms or definitions.

C.1 Proof Chain for Agreement

Terse proof chain for proof agreement_proof in module lemma_final

Use of the formula
\texttt{lemma\_final.synctime\_multiples\_bnd}
requires the following TCCs to be proven
\texttt{lemma\_final\_tcc.posnumber\_TCC1}
\texttt{lemma\_final\_tcc.synctime\_multiples\_bnd\_TCC1}
\texttt{lemma\_final\_tcc.synctime\_multiples\_bnd\_TCC2}
\texttt{lemma\_final\_tcc.agreement\_proof\_TCC1}
\texttt{lemma\_final\_tcc.agreement\_proof\_TCC2}

Use of the formula
\texttt{division.div\_nonnegative}
requires the following TCCs to be proven
\texttt{division\_tcc.mult\_div\_i\_TCC1}
\texttt{division\_tcc.mult\_div\_TCC1}
\texttt{division\_tcc.div\_cancel\_TCC1}
\texttt{division\_tcc.ceil\_mult\_div\_TCC1}
\texttt{division\_tcc.div\_nonnegative\_TCC1}
\texttt{division\_tcc.div\_ineq\_TCC1}
\texttt{division\_tcc.div\_minus\_1\_TCC1}

=============== SUMMARY ===============
The proof chain is complete

The axioms and assumptions at the base are:
  clockassumptions.IClock_defn
  clockassumptions.Readererror
  clockassumptions.VClock_defn
  clockassumptions.accuracy_preservation_ax
  clockassumptions.beta_0
  clockassumptions.correct_closed
  clockassumptions.correct_count
  clockassumptions.init
  clockassumptions.mu_0
  clockassumptions.precision_enhancement_ax
  clockassumptions.rate_1
  clockassumptions.rate_2
  clockassumptions.rho_0
  clockassumptions.rho_1
  clockassumptions.rmax_0
  clockassumptions.rmin_0
  clockassumptions.rts0
  clockassumptions.rts1
  clockassumptions.rts2
  clockassumptions.rts_2
  clockassumptions.synctime_0
  clockassumptions.translation_invariance
  division.ceil_defn
  division.mult_div_1
  division.mult_div_2
  division.mult_div_3
  multiplication.mult_10
  multiplication.mult_non_neg
  readbounds.induction
Total: 29

The definitions and type-constraints are:
  absmod.abs
  basics.maxsync
  basics.maxsynctime
  basics.minsync
  clockassumptions.Adj
  clockassumptions.okay_Reading
  clockassumptions.okay_Readpred
  clockassumptions.okay_Readvars
clockassumptions.okay_pairs
lemma3.okayClocks
multiplication.mult
readbounds.okaymaxsync
Total: 12

The formulae used are:
absmod.abs_bnd
absmod.abs_com
absmod.abs_diff_3
basics.ReadClock_bnd
basics.ReadClock_bnd1
basics.ReadClock_bnd11
basics.ReadClock_bnd12
basics.ReadClock_bnd2
basics.abs_shift
basics.lemma_1
basics.lemma_1_1
basics.lemma_1_2
basics.lemma_2_0
basics.lemma_2_1
basics.lemma_2_2a
basics.lemma_2_2b
basics.maxsync_correct
basics.minsync_correct
basics.minsync_maxsync
basics.okay_Reading_shift1
basics.okay_Readvars_shift
basics.okay_Readvars_shift1
basics.okay_Readvars_shift11
basics.okay_Readvars_shift12
basics.okay_Readvars_shift_step2
basics.okay_Readvars_shift_stepb
clockassumptions.okay_Reading_defn_lr
clockassumptions.okay_Reading_defn_rl
clockassumptions.okay_Readpred_Reading
clockassumptions.okay_Readvars_defn_rl
clockassumptions.okay_pairs_Readvars
clockassumptions.precision_enhancement
clockassumptions.rts_0
clockassumptions.rts_1
division.ceil_mult_div
division.ceil_plus_mult_div
division.div_nonnegative
division.mult_div
division_tcc.ceil_mult_div_TCC1
division_tcc.div_cancel_TCC1
division_tcc.div_ineq_TCC1
division_tcc.div_minus_1_TCC1
division_tcc.div_nonnegative_TCC1
division_tcc.mult_div_1_TCC1
division_tcc.mult_div_TCC1
lemma3.abs_diff_2
lemma3.accuracy_pres_step0
lemma3.accuracy_pres_step1
lemma3.accuracy_pres_step2
lemma3.accuracy_preservation
lemma3.drift_bnd
lemma3.lemma3_1
lemma3.lemma3_1_1
lemma3.lemma3_2
lemma3.lemma3_2_0
lemma3.lemma3_2_1
lemma3.lemma3_2_step
lemma3.lemma3_2_step1
lemma3.lemma3_2_step2
lemma3.lemma3_2_step3
lemma3.lemma3_3
lemma3.lemma3_3_0
lemma3.lemma3_3_ind
lemma3.maxmax_gap
lemma3.maxsync_max
lemma3.minmax_gap
lemma3.minsync_min
lemma3.okayClocks_defn_lr
lemma3.okayClocks_defn_rl
lemma_final.synctime_multiples
lemma_final.synctime_multiples_bnd
lemma_final.synctime_multiples_step
lemma_final_tcc.agreement_proof_TCC1
lemma_final_tcc.agreement_proof_TCC2
lemma_final_tcc.posnumber_TCC1
lemma_final_tcc.synctime_multiples_bnd_TCC1
lemma_final_tcc.synctime_multiples_bnd_TCC2
multiplication.distrib
multiplication.distrib_minus
multiplication.mult_com
multiplication.mult_ldistrib
The completed proofs are:
  absmod.abs_bnd_proof
  absmod.abs_com_proof
  absmod.abs_diff_3_pr
  basics.ReadClock_bnd11_proof
  basics.ReadClock_bnd12_proof
  basics.ReadClock_bnd1_proof
  basics.ReadClock_bnd2_proof
  basics.ReadClock_bnd_proof
  basics.abs_shift_proof
  basics.lemma_1_1_proof
  basics.lemma_1_2_proof
  basics.lemma_1_proof
  basics.lemma_2_0_proof
  basics.lemma_2_1_proof
  basics.lemma_2_2a_proof
  basics.lemma_2_2b_proof
  basics.maxsync_correct_pr
  basics.minsync_correct_pr
  basics.minsync_maxsync_pr
  basics.okay_Reading_shift1_proof
  basics.okay_Readvars_shift11_proof
  basics.okay_Readvars_shift12_proof
  basics.okay_Readvars_shift1_proof
  basics.okay_Readvars_shift_proof
C.2 Proof Chain for ICA Translation Invariance

Terse proof chain for proof ica_translation_invariance_pr in module ica

Use of the formula

ica.ica_translation_invariance1

requires the following TCCs to be proven

ica_tcc.sigma_TCC1
Formula ica_tcc.sigma_TCC2 is a termination TCC for ica.sigma
Proof of
ica_tcc.sigma_TCC2
must not use
ica.sigma

Use of the formula
division.div_distrib
requires the following TCCs to be proven
division_tcc.mult_div_l_TCC1
division_tcc.mult_div_TCC1
division_tcc.div_cancel_TCC1
division_tcc.ceil_mult_div_TCC1
division_tcc.div_nonnegative_TCC1
division_tcc.div_ineq_TCC1
division_tcc.div_minus_1_TCC1

============== SUMMARY ===============

The proof chain is complete

The axioms and assumptions at the base are:
clockassumptions.N_0
division.mult_div_1
division.mult_div_2
division.mult_div_3
ica.fun_extensionality
multiplication.mult_10
readbounds.induction
Total: 7

The definitions and type-constraints are:
ica.fix
ica.icalg
ica.iconv
ica.sigma
ica.sigma_size
multiplication.mult
Total: 6

The formulae used are:
division.div_cancel
division.div_distrib
division_tcc.ceil_mult_div_TCC1
division_tcc.div_cancel_TCC1
division_tcc.div_ineq_TCC1
division_tcc.div_minus_1_TCC1
division_tcc.div_nonnegative_TCC1
division_tcc.mult_div_1_TCC1
division_tcc.mult_div_TCC1
ica.fix_trans
ica.ica_translation_invariance1
ica.sigma_trans_inv
ica.sigma_trans_inv_base
ica.sigma_trans_inv_ind
ica_tcc.icalg_TCC1
ica_tcc.sigma_TCC1
ica_tcc.sigma_TCC2
multiplication.distrib
multiplication.mult_lident
multiplication.mult_rident

Total: 20

The completed proofs are:
division.div_cancel_pr
division.div_distrib_pr
division_tcc.ceil_mult_div_TCC1_PROOF
division_tcc.div_cancel_TCC1_PROOF
division_tcc.div_ineq_TCC1_PROOF
division_tcc.div_minus_1_TCC1_PROOF
division_tcc.div_nonnegative_TCC1_PROOF
division_tcc.mult_div_1_TCC1_PROOF
division_tcc.mult_div_TCC1_PROOF
ica.fix_trans_pr
ica.ica_translation_invariance1_pr
ica.ica_translation_invariance_pr
ica.sigma_trans_inv_base_pr
ica.sigma_trans_inv_ind_pr
ica.sigma_trans_inv_pr
ica_tcc.sigma_TCC1_PROOF
multiplication.distrib_proof
multiplication.mult_lident_proof
multiplication.mult_rident_proof
tcc_proofs.icalg_TCC1_PROOF
tcc_proofs.sigma_TCC2_PROOF
C.3 Proof Chain for ICA Precision Enhancement

Terse proof chain for proof icalg_precision_enhancement_pr in module ica3

Use of the formula
ica3.prec_enh_step4
requires the following TCCs to be proven
ica3_tcc.icalg_Fi_TCC1
ica3_tcc.icalg_precision_enhancement_step_TCC1
ica3_tcc.prec_enh_step3_pr_TCC1

Use of the formula
countmod.count
requires the following TCCs to be proven
countmod_tcc.count_TCC1
countmod_tcc.count_TCC2
countmod_tcc.count_TCC3
countmod_tcc.count_TCC4
countmod_tcc.count_TCC5

Formula countmod_tcc.count_TCC4 is a termination TCC for countmod.count
Proof of
countmod_tcc.count_TCC4
must not use
countmod.count

Formula countmod_tcc.count_TCC5 is a termination TCC for countmod.count
Proof of
countmod_tcc.count_TCC5
must not use
countmod.count

Use of the formula
division.div_ineq
requires the following TCCs to be proven
division_tcc.mult_div_1_TCC1
division_tcc.mult_div_TCC1
division_tcc.div_cancel_TCC1
division_tcc.ceil_mult_div_TCC1
division_tcc.div_nonnegative_TCC1
division_tcc.div_ineq_TCC1
division_tcc.div_minus_1_TCC1

Use of the formula
  ica.sigma
requires the following TCCs to be proven
  ica_tcc.sigma_TCC1
  ica_tcc.sigma_TCC2
  ica_tcc.icalg_TCC1

Formula ica_tcc.sigma_TCC2 is a termination TCC for ica.sigma
Proof of
  ica_tcc.sigma_TCC2
must not use
  ica.sigma

=============== SUMMARY ===============

The proof chain is complete

The axioms and assumptions at the base are:
  clockassumptions.N_0
  clockassumptions.N_maxfaults
  division.mult_div_1
  division.mult_div_2
  division.mult_div_3
  ica3.Delta_0
  multiplication.mult_10
  multiplication.mult_non_neg
  multiplication.mult_pos
  readbounds.induction
Total: 10

The definitions and type-constraints are:
  absmod.abs
  clockassumptions.okay_Readpred
  clockassumptions.okay_pairs
  countmod.count
  countmod.countsize
  ica.fix
  ica.icalg
  ica.iconv
  ica.sigma
ica.sigma_size
ica3.icalg_Mu
ica3.icalg_Pi
multiplication.mult

Total: 13

The formulae used are:
absmod.abs_1_bnd
absmod.abs_2_bnd
absmod.abs_3_bnd
absmod.abs_com
absmod.abs_drift
absmod.abs_leq_0
countmod_tcc.count_TCC1
countmod_tcc.count_TCC2
countmod_tcc.count_TCC3
countmod_tcc.count_TCC4
countmod_tcc.count_TCC5
division.div_distrib
division.div_ineq
division.div_minus_distrib
division.mult_div
division.mult_minus
division_tcc.ceil_mult_div_TCC1
division_tcc.div_cancel_TCC1
division_tcc.div_ineq_TCC1
division_tcc.div_minus_1_TCC1
division_tcc.div_nonnegative_TCC1
division_tcc.mult_div_1_TCC1
division_tcc.mult_div_TCC1
ica2.fix_diff
ica2.fix_diff1
ica2.fix_diff2
ica2.fix_diff3
ica2.fix_diff_corr
ica2.iconv_sigma_diff
ica2.okay_Readpred_lr
ica2.okay_Readpred_pairs
ica2.okay_pairs_fix
ica2.okay_pairs_lr
ica2.sigma_diff
ica2.sigma_diff_ind
ica2.sigma_neg
ica2.sigma_neg_ind
ica2.sigma_neg_ind_step
ica2.sigma_pos
ica2.sigma_pos_ind
ica2.sigma_pos_neg
ica2.sigma_split
ica2.sigma_split_ind
ica3.count_complement
ica3.icalg_precision_enhancement_step
ica3.multip_sum_ineq
ica3.prec_enh_step
ica3.prec_enh_step2
ica3.prec_enh_step3
ica3.prec_enh_step4
ica3.tcc.icalg_PI_TCC1
ica3.tcc.icalg_precision_enhancement_step_TCC1
ica3.tcc.prec_enh_step3_pr_TCC1
ica3.tcc.icalg_TCC1
ica3.tcc.sigma_TCC1
ica3.tcc.sigma_TCC2
multiplication.distrib
multiplication.distrib_minus
multiplication.mult_com
multiplication.mult_gt
multiplication.mult_l_distrib_minus
multiplication.mult_leq_2
multiplication.mult_l_ident
multiplication.mult_r_distrib_minus
multiplication.mult_r_ident
Total: 64

The completed proofs are:
absmod.abs_1_bnd_proof
absmod.abs_2_bnd_proof
absmod.abs_3_bnd_proof
absmod.abs_com_proof
absmod.abs_drift_proof
absmod.abs_leq_0_proof
countmod_tcc.count_TCC1_PROOF
countmod_tcc.count_TCC2_PROOF
countmod_tcc.count_TCC3_PROOF
division.div_distrib_pr
division.div_ineq_pr
division.div_minus_distrib_pr
division.mult_div_pr
division.mult_minus_pr
C.4 Proof Chain for ICA Accuracy Preservation

Terse proof chain for proof icalg_accuracy_preservation_pr in module ica4

Use of the formula
    ica4.acc_pres_step
requires the following TCCs to be proven
    ica4_tcc.icalg_accuracy Preservation_TCC1
    ica4_tcc.icalg_accuracy_preservation_pr_TCC1

Use of the formula
    ica.sigma
requires the following TCCs to be proven
    ica_tcc.sigma_TCC1
    ica_tcc.sigma_TCC2
    ica_tcc.icalg_TCC1

Formula ica_tcc.sigma_TCC2 is a termination TCC for ica.sigma
Proof of
    ica_tcc.sigma_TCC2
must not use
    ica.sigma

Use of the formula
    countmod.count
requires the following TCCs to be proven
    countmod_tcc.count_TCC1
    countmod_tcc.count_TCC2
    countmod_tcc.count_TCC3
    countmod_tcc.count_TCC4
    countmod_tcc.count_TCC5
Formula `countmod_tcc.count_TCC4` is a termination TCC for `countmod.count`

Proof of `countmod_tcc.count_TCC4`

must not use `countmod.count`

Formula `countmod_tcc.count_TCC5` is a termination TCC for `countmod.count`

Proof of `countmod_tcc.count_TCC5`

must not use `countmod.count`

Use of the formula `ica3.Delta_0` requires the following TCCs to be proven

- `ica3_tcc.icalg_Pi_TCC1`
- `ica3_tcc.icalg_precision_enhancement_step_TCC1`
- `ica3_tcc.prec_enh_step3_pr_TCC1`

Use of the formula `division.abs_div` requires the following TCCs to be proven

- `division_tcc.mult_div_1_TCC1`
- `division_tcc.mult_div_TCC1`
- `division_tcc.div_cancel_TCC1`
- `division_tcc.ceil_mult_div_TCC1`
- `division_tcc.div_nonnegative_TCC1`
- `division_tcc.div_ineq_TCC1`
- `division_tcc.div_minus_1_TCC1`

The proof chain is complete

The axioms and assumptions at the base are:

- `clockassumptions.N_0`
- `clockassumptions.N_maxfaults`
- `division.mult_div_1`
- `division.mult_div_2`
- `division.mult_div_3`
- `ica3.Delta_0`
- `multiplication.mult_10`
- `multiplication.mult_non_neg`
- `multiplication.mult_pos`
readbounds.induction
Total: 10

The definitions and type-constraints are:
  absmod.abs
  clockassumptions.okay_Readpred
  clockassumptions.okay_pairs
  countmod.count
  countmod.countsize
  ica.fix
  ica.icalg
  ica.iconv
  ica.sigma
  ica.sigma_size
  multiplication.mult
Total: 11

The formulae used are:
  absmod.abs_1_bnd
  absmod.abs_2_bnd
  absmod.abs_3_bnd
  absmod.abs_drift
  absmod.abs_leq_0
  absmod.abs_plus
  countmod_tcc.count_TCC1
  countmod_tcc.count_TCC2
  countmod_tcc.count_TCC3
  countmod_tcc.count_TCC4
  countmod_tcc.count_TCC5
  division.abs_div
  division.div_cancel
  division.div_distrib
  division.div_ineq
  division.div_minus_1
  division.div_minus_distrib
  division.div_nonnegative
  division.mult_div
  division.mult_minus
  division_tcc.ceil_mult_div_TCC1
  division_tcc.div_cancel_TCC1
  division_tcc.div_ineq_TCC1
  division_tcc.div_minus_1_TCC1
  division_tcc.div_nonnegative_TCC1
  division_tcc.mult_div_1_TCC1
The completed proofs are:
absmod.abs_1_bnd_proof
absmod.abs_2_bnd_proof
absmod.abs_3_bnd_proof
absmod.abs_drift_proof
absmod.abs_leq_0_proof
absmod.abs_plus_pr
countmod_tcc.count_TCC1_PROOF

Total: 60
countmod_tcc.count_TCC2_PROOF
countmod_tcc.count_TCC3_PROOF
division.abs_div_pr
division.div_cancel_pr
division.div_distrib_pr
division.div_ineq_pr
division.div_minus_1_pr
division.div_minus_distrib_pr
division.div_nonnegative_pr
division.mult_div_pr
division.mult_minus_pr
division_tcc.ceil_mult_div_TCC1_PROOF
division_tcc.div_cancel_TCC1_PROOF
division_tcc.div_ineq_TCC1_PROOF
division_tcc.div_minus_1_TCC1_PROOF
division_tcc.div_nonnegative_TCC1_PROOF
division_tcc.mult_div_1_TCC1_PROOF
division_tcc.mult_div_TCC1_PROOF
ica2.okay_Readpred_lr_pr
ica2.sigma_diff_ind_pr
ica2.sigma_diff_pr
ica2.sigma_pos_ind_pr
ica2.sigma_pos_pr
ica2.sigma_split_ind_pr
ica2.sigma_split_pr
ica3.count_complement_pr
ica3.mult_sum_ineq_pr
ica4.acc_pres_sigma_neg_pr
ica4.acc_pres_sigma_pos_pr
ica4.acc_pres_step_pr
ica4.icalg_accuracy_preservation_pr
ica4.okay_Readpred_fix_diff2_pr
ica4.okay_Readpred_fix_diff_pr
ica4.sigma_abs_pr
ica4.sigma_duplicate_pr
ica_tcc.sigma_TCC1_PROOF
multiplication.distrib_minus_pr
multiplication.distrib_proof
multiplication.mult_com_pr
multiplication.mult_gt_pr
multiplication.mult_idistrib_minus_proof
multiplication.mult_leq_2_pr
multiplication.mult_lident_proof
multiplication.mult_rident_proof
multiplication.pos_product_pr
tcc_proofs.countmod_TCC4_pr
tcc_proofs.countmod_TCC5_pr
tcc_proofs.icalg_Pi_TCC1_PROOF
tcc_proofs.icalg_TCC1_PROOF
tcc_proofs.icalg_accuracy_preservation_TCC1_PROOF
tcc_proofs.icalg_accuracy_preservation_pr_TCC1_PROOF
tcc_proofs.icalg_precision_enhancement_step_TCC1_PROOF
tcc_proofs.prec_enh_step3_pr_TCC1_PROOF
tcc_proofs.sigma_TCC2_PROOF
Total: 61
Schneider [1] generalizes a number of protocols for Byzantine fault tolerant clock synchronization and presents a uniform proof for their correctness. We present a machine checked proof of this schematic protocol that revises some of the details in Schneider's original analysis. The verification was carried out with the EHDM system [2] developed at the SRI Computer Science Laboratory. The mechanically checked proofs include the verification that the egocentric mean function used in Lamport and Melliar-Smith's Interactive Convergence Algorithm [3] satisfies the requirements of Schneider's protocol.