A Case Study in Analytic Protocol Analysis in ACL2

Max von Hippel*
Northeastern University
Boston, Massachusetts
vonhippel.m@northeastern.edu

Panagiotis Manolios
Northeastern University
Boston, Massachusetts
p.manolios@northeastern.edu

Kenneth L. McMillan
University of Texas at Austin
Austin, Texas
kenmcm@cs.utexas.edu

Cristina Nita-Rotaru
Northeastern University
Boston, Massachusetts
c.nitarot@northeastern.edu

Lenore Zuck
University of Illinois Chicago
Chicago, Illinois
zuck@uic.edu

When verifying computer systems we sometimes want to study their asymptotic behaviors, i.e., how they behave in the long run. In such cases, we need *real analysis*, the area of mathematics that deals with limits and the foundations of calculus. In a prior work, we used real analysis in ACL2s to study the asymptotic behavior of the RTO computation, commonly used in congestion control algorithms across the Internet. One key component in our RTO computation analysis was proving in ACL2s that for all $\alpha \in [0,1)$, the limit as $n \to \infty$ of α^n is zero. Whereas the most obvious proof strategy involves the logarithm, whose codomain includes irrationals, by default ACL2 only supports rationals, which forced us to take a non-standard approach. In this paper, we explore different approaches to proving the above result in ACL2(r) and ACL2s, from the perspective of a relatively new user to each. We also contextualize the theorem by showing how it allowed us to prove important asymptotic properties of the RTO computation. Finally, we discuss tradeoffs between the various proof strategies and directions for future research.

1 Introduction

In contrast to purely mathematically oriented theorem provers, ACL2 is designed specifically with the verification of computer systems in mind. This focus manifests in a variety of places across the ACL2 software landscape, such as the automated proofs of termination based on context-calling graphs provided by ACL2s [2, 1, 7], the integration of QuickLisp, or the development of advanced string-solving capabilities [6]. It also manifests in what ACL2 lacks: e.g., ACL2 does not support irrationals such as e, π, or $\sqrt{2}$, meaning it cannot be used to reason about the reals in full generality. Although this limitation is often immaterial, it shows up when we want to study the asymptotic behaviors of computer systems, meaning how they behave in the long run. In such cases we require *real analysis*, the area of mathematics that deals with limits and the foundations of calculus. However, in general real analysis proofs are riddled with *real* numbers; e.g., the logarithm is often used in these proofs, and it is often the case that the logarithm of a rational number will be irrational. If all we want is to prove a real analysis result, we can use ACL2(r) [3], the variant of ACL2 that supports real numbers. Unfortunately, ACL2(r) proofs cannot be imported into a normal ACL2 environment because the two systems have conflicting underlying theories, e.g., in ACL2 it is a theorem that $\forall x :: x^2 \neq 2$, while in ACL2(r), $\sqrt{2}$ is a number. So, what if we have formalized and studied a computer system in ACL2, and now we want to look at its asymptotic behaviors, without needing to port our model to ACL2(r)?

*Authors are listed alphabetically by last name.

A. Coglio, S. Swords (Eds.): 18th ACL2 Workshop (ACL2-2023)
EPTCS 393, 2023, pp. 50–66, doi:10.4204/EPTCS.393.6

In this work we focus on one such scenario, drawn from our recent work [4] studying Karn's algorithm [5] and the related Retransmission TimeOut (RTO) computation [9]. This work can be viewed as a companion paper to our work in [4], with a special focus on the proof strategy for the RTO observations. Next, we briefly summarize our work in [4] as it provides context for this study.

1.1 Motivating Example

In order to understand the RTO computation and its motivation, we first need to understand the context in which it is used. In the real world this context is an Internet connection between two computers, such as might occur between a sender and receiver using TCP. In the most general sense, the context consists of two endpoints communicating over a *channel*. The endpoints send and receive *datagrams* partitioned into *packets* (that the sender transmits to the receiver) and *acknowledgments*, or ACKs (that the receiver transmits to the sender), and each datagram is uniquely identifiable by its natural id and type (packet or ACK). The channel is responsible for delivering messages transmitted from one endpoint to the other. However, it might not do so reliably. It cannot create new messages, but may reorder, drop, delay, or duplicate transmitted ones.[1]

We assume the sender does not transmit a packet $p > 1$ unless it previously transmitted all packets in $[1, p-1]$, although it may transmit $p = 1$ at any time. We additionally assume that ACKs are cumulative in the sense that, the receiver does not transmit an ACK $a > 1$ unless it was previously delivered packets $1, 2, \ldots, a-1$ but not a. In the event that the receiver cannot cumulatively acknowledge anything, for instance, if it was delivered the packet 2 but never 1, then it may transmit the trivial acknowledgment of 1 (which does not acknowledge any packets). We also assume the receiver transmits an ACK whenever it is delivered a packet. When the sender receives an acknowledgment a, it considers a to be *new* iff a exceeds all the ACKs it received previously. In other words, a is new iff it acknowledges at least one previously un-acknowledged packet.

In the real world, the channel may have limited bandwidth, and will start to lose datagrams when its queues become full[2]. This bandwidth is unknowable to either endpoint, so the sender is forced to use ACKs to assess the instantaneous state of the channel and react accordingly. It does so in two ways. First, the sender measures the round-trip time (RTT) using its local clock between when it first transmits a packet p and when it first receives any acknowledgment $a > p$, as an indication of the pace at which the channel is delivering datagrams. It can use this information to moderate its transmission rate. Second, if no new ACKs arrive for some amount of time, the sender can assume the channel has been overwhelmed and is dropping data. In this case, it can slow its pace of transmission, and begin retransmitting unacknowledged data accordingly. The time the sender will wait before timing out, slowing its pace, and retransmitting, is called the RTO and defined in RFC6298 [9].

The conjunction of these two mechanisms creates a problem: suppose the sender retransmits an unacknowledged packet p, then receives some new ACK $a > p$. How does it know which transmission of p triggered the ACK? The estimated RTT will differ depending on the answer. We illustrate this situation in Figure 1[3]. One solution, known as Karn's algorithm, is to only sample RTTs for packets that were transmitted precisely once, since ACKs for these packets are unambiguous [5].

We formally modeled Karn's algorithm in Ivy [8] with the network model outlined above. We proved various inductive invariants about the algorithm, including that it samples a real RTT, that this RTT is in some sense pessimistic, and that when the receiver-to-sender channel path is FIFO, this RTT is for the

[1] This model is less strict than the typical IP one where duplication is disallowed.

[2] Such lossy communication is commonly modeled using a so-called "token bucket filter".

[3] Icons are from https://openmoji.org/

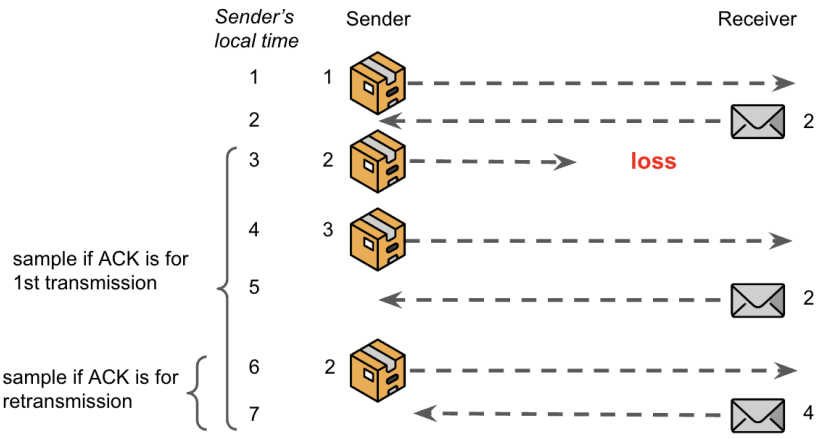

Figure 1: Message sequence chart illustrating an ambiguous ACK, with the sender's local clock shown on the left. Sender's packets are illustrated as packets, while receiver's ACKs are shown as envelopes. The first time the sender transmits 2 the packet is lost in-transit. Later, upon receiving a cumulative ACK of 2, the sender determines the receiver had not yet received the 2 packet and thus the packet might be lost in transit. It thus retransmits 2. Ultimately the receiver receives the retransmission and responds with a cumulative ACK of 4. When the sender receives this ACK it cannot determine which 2 packet delivery triggered the ACK transmission and thus, it does not know whether to measure an RTT of 7-3=4 or 7-6=1. Hence, the ACK is ambiguous, so any sampled RTT would be as well.

packet whose id is equal to the previously highest-received acknowledgment. Then, we formally modeled the RTO computation in ACL2s. We chose ACL2s over Ivy because the computation is defined over real numbers, which we chose to model as rationals, and Ivy only supports integers.[4] And in particular, we chose ACL2s over ACL2 because we made frequent use of its features. For example, we use the built-in counterexample generation to show that a variable referred to as the "RTT variance" is not actually a statistical variance; and in one of our proofs, we use an automated proof of function termination to prove the existence of a particular value (namely, the value returned by the function in question).

The RTO computation is recursively defined over the RTT samples $\mathsf{S}_1, \mathsf{S}_2, \ldots$ output by Karn's algorithm and parameterized by three positive constants ($\alpha < 1$, $\beta < 1$, and G) as follows.

$$\begin{aligned} \mathsf{rto}_i &= \mathsf{srtt}_i + \max(G, 4 \cdot \mathsf{rttvar}_i) \\ \mathsf{rttvar}_i &= \begin{cases} \mathsf{S}_i/2 & \text{if } i = 1 \\ (1-\beta)\mathsf{rttvar}_{i-1} + \beta|\mathsf{srtt}_{i-1} - \mathsf{S}_i| & \text{if } i > 1 \end{cases} \\ \mathsf{srtt}_i &= \begin{cases} \mathsf{S}_i & \text{if } i = 1 \\ (1-\alpha)\mathsf{srtt}_{i-1} + \alpha\mathsf{S}_i & \text{if } i > 1 \end{cases} \end{aligned} \tag{1}$$

Note, we use "RTO" when discussing the calculation generally, and "rto" when discussing its actual implementation (given in the equation above).

We looked at what we called the *steady-state* where for some rational *center* c and *radius* r, the samples $\mathsf{S}_i, \mathsf{S}_{i+1}, \ldots, \mathsf{S}_{i+n}$ all fall within the bounds $[c-r, c+r]$, and we proved the following.

[4] In practice, the implementations we are aware of use integers, however, when Karn and Partridge wrote the algorithm down on paper, they did so using reals.

I. The srtt_{i+n} is bounded by the interval $[L,H]$ defined as follows.

$$\begin{aligned} L &= (1-\alpha)^{n+1}\mathsf{srtt}_{i-1} + (1-(1-\alpha)^{n+1})(c-r) \\ H &= (1-\alpha)^{n+1}\mathsf{srtt}_{i-1} + (1-(1-\alpha)^{n+1})(c+r) \end{aligned} \tag{2}$$

II. For all $0 \leq m < n$, rttvar_{i+n} is upper-bounded by the following expression.

$$\begin{aligned} &(1-\beta)^{n+1-m}\mathsf{rttvar}_{i+m-1} + (1-(1-\beta)^{n+1-m})\Delta_m \text{ , where} \\ &\Delta_m = (1-\alpha)^{m+1}\mathsf{srtt}_{i-1} + 2r - (1-\alpha)^{m+1}(c+r) \end{aligned} \tag{3}$$

Then we reanalyzed the results in the asymptotic case. In other words, we asked what these bounds converge to as the number n of consecutively bounded samples, as well as the cutoff $m < n$ for the bound Δ_m defined above, grow toward infinity. By *limit*, we are referring to the standard definition[5] from real analysis, which we give below using the 1D Euclidean metric $d(x,y) = |x-y|$.

Definition 1 (Limit to ∞)**.** *Let* $(a_i)_{i=0}^{\infty} \in \mathbb{R}^{\omega}$ *and* $\ell \in \mathbb{R}$*. Then* $\lim_{i\to\infty} a_i = \ell$ iff *the following holds:*

$$\forall \varepsilon > 0 :: \exists \delta > 0 :: \forall n > \delta :: |a_n - \ell| < \varepsilon$$

We first proved the following theorem, which is the focus of this paper.

Theorem 1. $\forall \alpha \in [0,1) :: \lim_{n\to\infty} \alpha^n = 0$

Theorem 1 can be manually proven as follows.

Proof. Let $\varepsilon > 0$ and $0 \leq \alpha < 1$ arbitrarily. If $\alpha = 0$ the result is immediate; suppose $\alpha > 0$. Suppose $\varepsilon < 1$, noting that if the theorem holds for $\varepsilon < 1$ then it holds for $\varepsilon \geq 1$. Let $\delta = \ln(\varepsilon)/\ln(\alpha)$. Note that $\ln(\varepsilon)$ and $\ln(\alpha)$ are negative. Let n be some natural number and observe that $n\ln(\alpha) = \ln(\alpha^n)$. Thus:

$$\begin{aligned} n > \delta &\iff n > \ln(\varepsilon)/\ln(\alpha) && \text{by definition of } \delta \\ &\iff n\ln(\alpha) < \ln(\varepsilon) && \text{multiplying each side by } \ln(a) \\ &\iff e^{n\ln(\alpha)} < e^{\ln(\varepsilon)} && \text{raising each side above } e \\ &\iff e^{\ln(\alpha^n)} < \varepsilon && \text{because } e^{\ln(x)} = x \text{ for all } x, \text{ and } n\ln(\alpha) = \ln(\alpha^n) \\ &\iff \alpha^n < \varepsilon && \text{because } e^{\ln(x)} = x \text{ for all } x \end{aligned}$$

□

We then used this theorem to show that $\lim_{n\to\infty} L = c - r$, $\lim_{n\to\infty} H = c + r$, and the limit as n and $m < n$ both grow toward infinity of the upper bound in Eqn. 3 is precisely $2r$.

[5] Limit proofs that assume an $\varepsilon > 0$ and then prove the existence of a corresponding $\delta > 0$ satisfying this definition are commonly reffered to as ε/δ-proofs.

Outline. The rest of this paper is organized as follows. We study Theorem 1 in Sections 2 and 3. Specifically, in Section 2 we formalize the English-language proof given above in ACL2(r). But recall, we cannot use an ACL2(r) proof to study the RTO system we defined in ACl2s, without totally remodeling it, as the two proof systems are incompatible. Thus in order to have our asymptotic proofs in the same model as our other preexisting proofs about the RTO system, we need a rational proof. We give two such proofs in Section 3. The first uses the ceiling function to define a δ directly. The second begins by proving that $\lim_{n\to\infty} 1/2^n = 0$, then uses the binomial theorem to construct a δ such that $n > \delta \implies \alpha^\delta < 1/2$. For the second proof strategy, we show two different ways to prove $\lim_{n\to\infty} 1/2^n = 0$, one of which is more automatic than the other. With those proofs out of the way, we show how to derive the limits for the bounds on srtt and rttvar in Section 4, which concludes our analytic study of the RTO. We discuss trade-offs between the various proofs and lessons learned in Section 5 and conclude in Section 6.

2 Real Proof

In this Section we overview the most obvious proof strategy for Theorem 1 – the one we gave in the introduction – and its formalization in ACL2(r), the variant of ACL2 that supports real numbers. Because the rationals form a dense subset of the reals, this proof implies the desired result over the rationals as well. However, since ACl2 and ACL2(r) are theoretically incompatible, we cannot just import the proof into our preexisting ACL2s model to cohabitate with our other theorems about the RTO calculation.

The theorem we aim to prove uses an existential quantifier, so we define it via `defun-sk`. Note that our theorem statement will be the same in Section 3, except that since we will be using ACL2s in those proofs, we will also have type declarations and guards there. Notice how we can drop the absolute value signs from Definition 1 because α is assumed to be positive, implying that α^n is also positive.

```
(defun-sk lim-0 (a e n)
  (exists (d)
    (=> (^ (realp e) (< 0 e) (< d n)) (< (raise a n) e))))

(defthm lim-a^n->0
  (=> (^ (realp a) (< 0 a) (< a 1) (realp e) (< 0 e) (natp n))
      (lim-0 a e n)) :instructions ...) ;; proof will go here
```

The most important step in an ε/δ proof is defining the δ. We do so by defining a witness function $d_a : \varepsilon \to \delta$, so that the proof obligation reduces to showing $\forall \varepsilon > 0 :: \forall n > d_a(\varepsilon) :: \alpha^n < \varepsilon$.

```
(defun d (eps a) (/ (acl2-ln eps) (acl2-ln a)))
```

The remainder of the proof consists of two important steps. First, we define a number of arithmetic lemmas which ACL2s proves automatically. Second, because ACL2(r) lacks a generic real logarithm or exponent (having only the natural variants), we prove a translational lemma (R1) saying that $e^{n\ln(\alpha)} = \alpha^n$. Then we rephrase the proof from §1 in terms of e, at which point it goes through easily.

2.1 Arithmetic Lemmas

We began by proving some basic arithmetic lemmas, which we needed for the more complicated proofs. We proved that if $e^y < 1$ then $y < 0$, and that if $y \in (0,1)$, then $\ln(y) < 0$. Then we proved two facts about fractions of logarithms. First, if $\alpha \in (0,1)$ then $(\ln(\varepsilon)/\ln(\alpha))\ln(\alpha) = \ln(\varepsilon)$. Second, if ε and α both fall within $(0,1)$ and $n > \ln(\varepsilon)/\ln(\alpha)$, then $n\ln(\alpha) < \ln(\varepsilon)$, and thus, since the natural exponent is

monotonic (i.e., $x < y \implies e^x < e^y$), it follows that $e^{n\ln(\alpha)} < e^{\ln(\varepsilon)} = \varepsilon$. Finally, combining these results gave us that if α and $n \in \mathbb{R}_+$ then $\ln(\alpha^n) = \ln(e^{n\ln(\alpha)}) = n\ln(\alpha)$.

2.2 Translational Lemma

Our arithmetic lemmas allow us to prove the following translational result.

Lemma R1. *For all positive reals α and n, $e^{n\ln(\alpha)} = \alpha^n$.*

2.3 Proof of Theorem 1

Next we prove the desired result without loss of generality, and without explicitly using quantifiers. Our proof uses Lemma R1 as a hint.

Lemma R2. *Let α and ε be reals in $(0,1)$ and let $n > d_\alpha(\varepsilon)$ be a natural. Then $\alpha^n < \varepsilon$.*

Notice how this immediately gives us our desired result, because if $\varepsilon < 1$ then we can just use Lemma R2 directly, and if $\varepsilon \geq 1$, we can use it with a new "epsilon" $< \varepsilon$. And this is precisely the strategy we take in the `instructions` to our proof of Theorem 1, which go as follows:

i. Promote all variables then case-split on $\varepsilon < 1$.

ii. Case 1: $\varepsilon < 1$. Use Lemma R2. Then instantiate `lim-0-suff` with $\delta = d_\alpha(\varepsilon)$ and prove.

iii. Case 2: $\varepsilon > 1$. Use Lemma R2 with a new "epsilon" value of $\varepsilon' = 1/2$ and claim that all its preconditions are satisfied. Further claim that if $d_\alpha(1/2) < n$ then $\alpha^n < 1/2 < \varepsilon$. Then instantiate `lim-0-suff` with $\delta = d_\alpha(1/2)$, promote, and prove.

On the one hand, our proof is straightforward in the sense that it mostly follows the English-language proof we outlined in the introduction. On the other hand, we can easily see some places where more machinery and theorems in the nonstandard arithmetic library would drastically simplify things. The biggest ommision is that the nonstandard arithmetic library only supports natural exponent and logarithm, which forced us to use the translational lemma. Most interestingly, this is not the shortest proof in this paper! There is actually a more concise[6], rational proof which we outline in the next Section.

3 Rational Proofs

In this Section, we reprove Theorem 1 in ACL2s, using only rationals. We present two proofs. The first is the one we used in our motivating work [4], where we explicitly construct the δ using the ceiling function. To do so, we have to prove various properties of that function. The second proof is our most concise, and proceeds in two steps. First we show that $\lim_{n\to\infty} 1/2^n = 0$. Then using the binomial theorem, we show how, for any $0 < \alpha < 1$, we can construct an n such that $\alpha^n < 1/2$. The result follows. For convenience, we refer to the first proof (given in §3.1) as the *ceiling proof* and the second (§3.2) as the *binomial proof*. We recap in §3.3.

3.1 Ceiling Proof

In prose, the ceiling proof of Theorem 1 proceeds as follows.

[6](as measured by lines of code and number of imported books)

Proof. Let $0 \leq \alpha < 1$ and $\varepsilon > 0$, arbitrarily. Let $k = \lceil a/(1-a) \rceil$ and observe that $a \leq k/(k+1)$. Let $f(n) = k\alpha^k/n$. As an intermediary lemma, we claim that for all $n \geq k$, $\alpha^n \leq f(n)$.

Base Case: $n = k$ thus $f(n) = \alpha^k \geq \alpha^n$ and we are done.

Inductive Step: By inductive hypothesis, we have

$$a^n \leq k\alpha^k/n \tag{4}$$

and $k \leq n$. This gives us $k/(k+1) \leq n/(n+1)$ and thus:

$$\alpha \leq n/(n+1) \tag{5}$$

Multiplying Eqn. 4 through by α, we get $\alpha^{n+1} \leq k\alpha^{k+1}/n$. Combining this with Eqn. 5:

$$\alpha^{n+1} \leq (k\alpha^k/n)\frac{n}{n+1} = k\alpha^k/(n+1) \tag{6}$$

and we are done.

Hence induction: $\forall n \geq k$, $\alpha^n \leq f(n)$. Now, let $\delta = \lceil k\alpha^k/\varepsilon \rceil$. It follows that $\forall n \geq \delta$, $f(n) \leq \varepsilon$, and thus by the above result, $\alpha^n \leq \varepsilon$. We get $\alpha^n < \varepsilon$ by repeating this process for $\varepsilon/2$, and we are done. □

Although the proof is relatively straightforward on paper, as we will see, it is much more challenging in ACL2s. The primary issue is that ACL2/ACL2s does not by default know very much about the ceiling function, so we will be forced to prove many obvious lemmas before making the essential argument. Since we are in ACL2s now, we first restate Theorem 1 with types.

```
(defun-sk lim-0 (a e n)
  (declare (xargs :guard (and (posratp a) (< a 1) (posratp e) (natp n))
                  :verify-guards t))
  (exists (d) (and (natp d) (implies (< d n) (< (expt a n) e)))))

(property lim-a^n->0 (a e :pos-rational n :nat)
  :hyps (< a 1)
  (lim-0 a e n) :instructions ...) ;; proof will go here
```

The rest of the subsection is organized in follows. We prove arithmetic lemmas in §3.1.1. We use these lemmas to prove the "intermediary lemma" in §3.1.2, which we use to prove prove Thm. 1 in §3.1.3.

3.1.1 Arithmetic Lemmas

In order to fill out the `instructions`, we first need some lemmas, primarily about the ceiling function.

Lemma C1. *For all $x, y \in \mathbb{Q}_+$, if $\lceil x \rceil < \lceil y \rceil$ then (i) $x \leq \lceil x \rceil$ and (ii) $\lceil x \rceil < y$.*

For the next Lemma we write a manual proof, adapted from [10], and utilizing Lemma C1, which we give immediately below.

Lemma C2. *Let $m, n \in \mathbb{N}_+$ and $x \in \mathbb{Q}_+$. Then $\lceil x/mn \rceil = \lceil \lceil x/m \rceil / n \rceil$.*

Proof. Observe:

$$\lceil x/m \rceil - 1 < x/m \leq \lceil x/m \rceil \tag{7}$$

Dividing Eqn. 7 by n, we get $(\lceil x/m\rceil - 1)/n < x/mn \leq \lceil x/m\rceil/n$. We also observe that $\lceil x/mn\rceil \leq \lceil\lceil x/m\rceil/n\rceil$. Thus by Lemma C1:

$$x/mn \leq \lceil x/mn\rceil < \lceil x/m\rceil/n \tag{8}$$

Suppose (for a contradiction) that $\lceil x/mn\rceil < \lceil\lceil x/m\rceil/n\rceil$. Multiplying Eqn. 8 by n, we get $x/m \leq n\lceil x/mn\rceil < \lceil x/m\rceil$, which is sufficient information for ACL2s to automatically find the contradiction:

$$\lceil x/m\rceil \leq n\lceil x/mn\rceil < \lceil x/m\rceil \tag{9}$$

sufficing to show that $\lceil x/mn\rceil \not< \lceil\lceil x/m\rceil/n\rceil$. But then $\lceil x/mn\rceil = \lceil\lceil x/m\rceil/n\rceil$ and we are done. □

Next we observe that for all $x,y,z \in \mathbb{Q}_+$, if $y \leq z$ then $y/x \leq z/x$ and moreover, $yx \leq zx$. The following property of the ceiling function automatically follows. Next we make an important observation about the ceiling function.

Lemma C3. *Let $\alpha \in \mathbb{Q}$ such that $0 < \alpha < 1$. Let $k = \lceil \alpha/(1-\alpha)\rceil$. Then $\alpha \leq k/(1+k)$.*

Proof. First note that $\alpha/(1-\alpha) \leq k$. Observe that for all $x,y,z \in \mathbb{Q}_+$, if $y \leq z$, then $yx \leq zx$. It follows that $\alpha = (\alpha/(1-\alpha))(1-\alpha) \leq k(1-\alpha)$. Next observe that $\alpha \leq k(1-\alpha) = k - k\alpha$. Adding $k\alpha$ to each side, we get $\alpha + k\alpha = \alpha(1+k) \leq k$. Again consider $x,y,z \in \mathbb{Q}_+$ such that $y \leq z$, but this time, observe that $yx \leq zx$. Thus, $\alpha(1+k)/(1+k) = \alpha \leq k/(1+k)$, and we are done. □

Next we prove two lemmas about fractions.

Lemma C4. *For all $k \in \mathbb{N}_+$ and $\alpha \in \mathbb{Q}_+$, $k\alpha^k/k = \alpha^k$.*

Lemma C5. *For all $k \leq n \in \mathbb{N}, k/(1+k) \leq n/(1+n)$.*

Finally, we make some obvious arithmetic observations, leading to the following result.

Lemma C6. *For all $x,y \in \mathbb{Q}_+$, we have $x/\lceil x/y\rceil \leq y$.*

Proof. Note $x/y \leq \lceil x/y\rceil$, thus $1/\lceil x/y\rceil \leq 1/(x/y)$. Multiply both sides by x, and we are done. □

3.1.2 Inductive Proof of Intermediary Lemma

With these arithmetic lemmas completed we can move on to the actual proof. For convenience, we will define a parameterized function $f_\alpha : \mathbb{N}_+ \to \mathbb{Q}_+$ such that $f_\alpha(n) = k\alpha^k/n$ for $k = \lceil \alpha/(1-\alpha)\rceil$. As an intermediary lemma, we claim that for all $n \geq k$, $a^n \leq f(n)$. Assuming the lemma holds, we can let $\delta = \lceil ka^k/\varepsilon\rceil$, and we immediately get that for all $n \geq \delta$, $a^n \leq f_\alpha(n) \leq \varepsilon$. Theorem 1 immediately follows. This sub-subsection is spent proving the intermediary lemma.

Lemma C7 (Base Case)**.** *Let $\alpha \in \mathbb{Q}_+$ and let $k = \lceil \alpha/(1-\alpha)\rceil$. Then $f_\alpha(k) = a^k$.*

Proof. Follows directly from Lemma C4. □

Before the inductive step, we need one more helper lemma, the proof of which follows from our prior arithmetic observations.

Lemma C8. *For all $n,k \in \mathbb{N}_+$ and $\alpha \in \mathbb{Q}_+$, if $\alpha^{n+1} \leq \alpha k\alpha^k/n$, then $\alpha^{n+1} \leq ka^k/(1+n)$.*

Lemma C9 (Inductive Step)**.** *Let $\alpha \in \mathbb{Q}_+$ and $n \in \mathbb{N}$. Suppose that $\alpha < 1$, $k = \lceil \alpha/(1-\alpha)\rceil \leq n$, and $\alpha^n \leq f_\alpha(n)$. Then $\alpha^{1+n} \leq f_\alpha(1+n)$.*

Proof. By Lemma C7, $f_\alpha(k) = a^k$. By Lemma C5, $\alpha \leq k/(1+k) \leq n/(1+n)$. Then by our arithmetic observations, $\alpha^{n+1} \leq f_\alpha(k)\alpha$, and thus, $\alpha^{n+1} \leq ka^k/(1+n) = f_\alpha(1+n)$. □

Although at this point we've laid out our inductive argument, we still need to implement it in ACL2s. To begin with, this means defining an inductive scheme.

```
(definec ikn (a :pos-rational n :nat) :nat
  :ic (< a 1)
  (if (> (ceiling (/ a (- 1 a)) 1) n) 0 (1+ (ikn a (- n 1)))))
```

We use this scheme in the proof of the next lemma.

Lemma C10 (Intermediary Lemma)**.** *Let $\alpha < 1$ be in $\mathbb{Q}_+$ and $n \geq \lceil \alpha/(1-\alpha) \rceil$ in $\mathbb{N}$. Then $\alpha^n \leq f_\alpha(n)$.*

Proof. Induct on `ikn`. Use Lemma C7 for the base case and Lemma C9 for the inductive step. □

3.1.3 Proof of Theorem 1

Our proof strategy is as follows. First, we introduce a function $\delta_\alpha : \mathbb{R}_+ \to \mathbb{N}_+$ defined by $\varepsilon \mapsto \max\{k,d\}$, where $k = \lceil \alpha/(1-\alpha) \rceil$ as before, and $d = \lceil k\alpha^k/\varepsilon \rceil$. Then we prove three lemmas about this function (given immediately below) which together suffice to imply Theorem 1. We use k and d as defined above.

Lemma C11. *Let $\alpha < 1$ and ε be in $\mathbb{Q}_+$ and $n \in \mathbb{N}$. Suppose $\delta_\alpha(\varepsilon) \leq n$. Then $k \leq n$.*

Proof. By definition of δ_α, we have $\max\{k,d\} \leq n$. Since $k \leq \max\{k,d\}$, we are done. □

Lemma C12. *Let $\alpha < 1$ and ε be in $\mathbb{Q}_+$ and $n \in \mathbb{N}$. Suppose $\delta_\alpha(\varepsilon) \leq n$. Then $\alpha^n \leq f_\alpha(n)$.*

Proof. Follows automatically from Lemmas C10 and C11 with the definitions of f_α and δ_α. □

Lemma C13. *Let $\alpha < 1$ and ε be in $\mathbb{Q}_+$ and $n \in \mathbb{N}$. Suppose $\delta_\alpha(\varepsilon) \leq n$. Then $f_\alpha(n) \leq \varepsilon$.*

Proof. By the definition of δ_α, we have $\max\{k,d\} \leq n$. Our prior arithmetic observations give us that $k\alpha^k/n \leq k\alpha^k/d$. Since $d = \lceil k\alpha^k/\varepsilon \rceil$, by Lemma C6, clearly $k\alpha^k/d \leq \varepsilon$. The result follows. □

Armed with these Lemmas, we can prove a "helper lemma" like we did previously. But in this case, we use $\leq$ instead of $<$ because of the way we structured our argument based on the intermediary result.

Lemma C14. *For all $\alpha < 1$ and ε in $\mathbb{Q}_+$ and $n \in \mathbb{N}_+$, if $\delta_\alpha(\varepsilon) \leq n$ then $\alpha^n \leq \varepsilon$.*

Proof. Follows from Lemmas C12 and C13 after observing that all their preconditions are met. □

Finally, we can provide the `instructions` to prove Theorem 1. Note how we divide ε by 2 in order to transform $\leq$ into $<$, to fit Definition 1.[7]

```
((:use (:instance lim-0-suff (d (delta a (/ e 2)))))
 (:use (:instance a^n->0 (a a) (e (/ e 2)) (n n)))
 :pro :prove)
```

[7] N.b., this trick suffices to show that the alternative definition with $\leq$ is equivalent.

3.2 Binomial Proof

In this subsection, we propose an alternative proof. The strategy can be split into two steps. First, we prove that $0 \leq \alpha \leq 1/2 \implies \lim_{n\to\infty} \alpha^n = 0$. Second, we prove that for all $\alpha \in [0,1)$, there exists some $\delta \in \mathbb{N}$ such that $n > \delta \implies \alpha^n \leq 1/2$. We rely on the binomial theorem to find this δ, hence the name of the proof. These results suffice to prove Theorem 1.

We found two ways to approach the first step. The first way was to attack the problem directly, with an ε/δ proof. The second was to leverage the termination analysis in ACL2s to find a δ semi-automatically. We cover the first approach in §3.2.1 and the second in §3.2.2. Then in §3.2.3 we show how, given either approach, we can prove Thm. 1 by completing the "second step" described above.

3.2.1 Manual Proof of $0 \leq \alpha \leq 1/2 \implies \lim_{n\to\infty} \alpha^n = 0$.

We begin by importing the proof-by-arithmetic book.

```
(include-book "make-event/proof-by-arith" :dir :system)
```

We then prove the a sequence of simple arithmetic facts, using some combination of the `linear`, `match-free`, and `all` rule classes. First, like we did in the prior section, we show that the exponent is monotonic. Second, we show that for all $n \in \mathbb{N}$, $n < 2^n$, and thus, if n is positive, then $1/2^n < 1/n$. Then we introduce an arithmetic trick by which we can extract a number smaller than ε, namely, if $\varepsilon = x/y$ is a positive rational, then $1/y < \varepsilon$. Combining these results yields the following two lemmas.

Lemma BM1. *For all $\alpha \leq 1/2$ in $\mathbb{Q}_+$, and for all $d \in \mathbb{N}_+$, $\alpha^d \leq 1/2^d$.*

Lemma BM2. *For all $\alpha, \varepsilon = x/y \in \mathbb{Q}_+$, where $x, y \in \mathbb{N}_+$, if $\alpha \leq 1/2$, then $\alpha^y \leq \varepsilon$.*

At this point, the desired result follows directly from Lemma BM2.

3.2.2 Semi-Automatic Proof of $0 \leq \alpha \leq 1/2 \implies \lim_{n\to\infty} \alpha^n = 0$.

In the semi-automatic proof, we begin by defining two functions. The first function, $\mu : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, is defined by $(b,q) \mapsto b$ *if* $q < 2^b$ *else* $\mu(b+1,q)$, and can be viewed as a "helper function" to the second, $d : \mathbb{Q}_+ \to \mathbb{N}$, which is defined by $\varepsilon \mapsto \mu(0, \text{denominator}(\varepsilon))$. When we define these two functions, ACL2s automatically proves that they terminate, meaning that for all possible inputs of $\varepsilon \in \mathbb{Q}_+$, $\mu(0,\varepsilon)$ terminates. We use this with an inverse argument to show $1/2^{d(\varepsilon)} \leq \varepsilon$, which we then manipulate to get the desired result. Because the argument and manipulation require writing down additional lemmas, we call this proof *semi*-automatic. Next, we prove three lemmas about these functions. The first two go through automatically, whereas the third requires a manual proof.

Lemma BA1. *For all $b, q \in \mathbb{N}$, $q < 2^{\mu(b,q)}$.*

Lemma BA2. *For all $q \in \mathbb{N}_+$, $1/2^{\mu(0,q)} < 1/q$.*

Lemma BA3. *For all $\varepsilon > 0$, $1/2^{d(\varepsilon)} < \varepsilon$.*

Proof. First observe that $1/\text{denominator}(\varepsilon) \leq \varepsilon$. Then observe that $1/2^{\text{denominator}(\varepsilon)} < 1/\text{denominator}(\varepsilon)$. The rest follows automatically. □

Next, we establish the monotonicity of the exponent, both strictly ($<$) and otherwise ($\leq$). This allows us to prove the following.

Lemma BA4. *For all $k \leq n \in \mathbb{N}_+$ and $\alpha < 1$ in $\mathbb{Q}_+$, $\alpha^n \leq \alpha^k$.*

Lemma BA5. *For all $\varepsilon > 0$ in $\mathbb{Q}_+$ and $n > d(\varepsilon)$ in $\mathbb{N}$, $1/2^n < \varepsilon$.*

Proof. Follows from the (non-strict) monotonicity of the exponent, Lemma BA3, and the observation that for all $n \in \mathbb{N}$, $1/2^n = (1/2)^n$. □

We prove the next lemma in its given form, and rewritten using *numerator* and *denominator*.

Lemma BA6. *For all $p,q \in \mathbb{N}_+$, if $p/q < 1$ then $p < q$.*

Lemma BA7. *For all $x \leq y$ and α in $\mathbb{N}_+$, $\alpha/y \leq \alpha/x$.*

Finally, we get the desired result.

Lemma BA8. *For all $\alpha \leq 1/2$ and ε in $\mathbb{Q}_+$, and for all $n \geq d(\varepsilon)$ in $\mathbb{N}$, $\alpha^n < \varepsilon$.*

Proof. Follows from Lemmas BA5 and BA7. □

Having shown two ways to derive the first step of our outlined proof strategy, we now move on to the second step, where we invoke the binomial theorem.

3.2.3 Remainder of Binomial Proof

The remainder of the proof begins with the following lemma. The lemma is the same regardless of which strategy we take for part 1 (i.e., manual, or semi-automatic), however, its proof differs slightly with each choice. Thus for brevity, we only include the proof assuming we took the semi-automatic approach, since it is the most recent in the text and thus the easiest to compare to here. The alternative version given the manual approach is very nearly identical.

Lemma BB1. *For all $\alpha = x/y \in \mathbb{Q}_+$, where $x,y \in \mathbb{N}_+$, $\alpha \leq x/(1+x)$.*

Proof. Follows from Lemmas BA6 and BA7. □

Next we import the binomial theorem into our proof.

```
(include-book "arithmetic/binomial" :dir :system)
```

Note that the `binomial` book was written in ACL2. Since we are in ACL2s, we have an additional obligation to check types. So, we prove four convenient lemmas about the types involved in the binomial expansion as defined in that book.

Lemma BB2. *The codomain of the `choose` function is a subset of $\mathbb{Z}$.*

Lemma BB3. *The integer-exponent of an integer is an integer.*

Lemma BB4. *The binomial `expansion` (defined in the `binomial` book) is a list of naturals.*

Lemma BB5. *For any list of naturals, its summation under the `sumlist` function is a natural.*

Now we get to the crux of the argument. Essentially, we will show that if $\alpha < 1$ is a rational with numerator p and denominator q, then $\alpha = p/q \leq p/(p+1)$ and thus $\alpha^p \leq p^p/(p+1)^p$. By the binomial theorem, $(p+1)^p \geq 2p^p$, thus $\alpha^p \leq 1/2$, allowing us to reduce to the argument from step 1. Formally speaking, we accomplish this through the following sequence of lemmas.

Lemma BB6. *For all $n \in \mathbb{N}_+$, $2n^n \leq (1+n)^n$.*

Proof. Follows from the binomial theorem because $(1+n)^n \leq 1 + \ldots + nn^{n-1} + n^n$. □

Now we prove a "helper lemma", similar to what we did in prior proof strategies but this time for the goal of "squeezing" α^n below $1/2$.

Lemma BB7. *For all $\alpha < 1$ in $\mathbb{Q}_+$, $\alpha^{numerator(\alpha)} \leq 1/2$.*

Proof. Let $x/y = \alpha$. By Lemma BB6, $2y^y \leq (1+y)^y$. By Lemma BA7, $(y^y)/(1+y)^y < (y^y)/(2y^y)$. But since for all $a,b,c \in \mathbb{Q}_+, a^b/c^b = (a/c)^b$ and $a/2a = 1/2$, it immediately follows that $(y/(1+y))^y < 1/2$. By Lemma BB1, $\alpha \leq \alpha/(1+\alpha)$. Combining this with the (non-strict) monotinicity of the exponent, clearly $\alpha^y \leq (\alpha/(1+\alpha))^y < 1/2$. The rest follows from Lemma BA8 (or the equivalent result, in the case of the manual proof). □

3.3 Summary and Closing Thoughts

In this Section we provided two rational proof strategies for Theorem 1 – the "ceiling proof" and the "binomial proof" – both of which we implemented using ACL2s. Since the rationals are dense in the reals, these proof strategies equally apply to the real numbers. The ceiling proof, which is the one we used in our prior work [4], involved first proving an intermediary lemma about the ceiling function. Specifically, assuming $0 < \alpha < 1$ and setting $k = \lceil \alpha/(1-\alpha) \rceil$ and $\delta = \lceil k\alpha^k/\varepsilon \rceil$, we proved that $\forall n \geq \delta$, $\alpha^n \leq f_\alpha(n)$. Since $f_\alpha(n) \leq \varepsilon$, the result directly followed. However, to prove this we first had to establish many arithmetic lemmas about the ceiling function, so although straightforward on paper, the proof was comparatively arduous in ACL2s.

For the "binomial proof", we broke the problem into two steps, first showing that if $0 < \alpha \leq 1/2$ then $\lim_{n\to\infty} \alpha^n = 0$, and then for any $\alpha \in (1/2, 1)$, constructing a δ such that $n > \delta \implies \alpha^n \leq 1/2$. For the first step, we showed two different approaches, one manual and the other semi-automatic. What made the second approach semi-automatic was that we used the termination analysis capabilities of ACL2s to find the "δ" for our ε/δ proof automatically. However, we still had to prove that this δ satisfied Definition 1. For the second step, we used the binomial theorem to show that for all positive integers p, $2p^p \leq (p+1)^p$ and therefore, if $\alpha = p/q \leq p/(p+1)$ then $\alpha^p \leq (p/(p+1))^p \leq 1/2$. Overall, both versions of the binomial proof were considerably simpler (in terms of lines of code) than the ceiling proof in ACL2s, and the comparison is fair given that both strategies required importing preexisting books (refer to Table 1).

Next, we return to the RTO, and show how any proof of Theorem 1 allows us to characterize the asymptotic behaviors of the srtt and rttvar. We also discuss the implications of these results for the rto.

4 Analysis of RTO Calculation

Recall from Eqn 1 that the rto is defined over the rttvar, srtt, and RTT sample S; the rttvar is defined over the S and the prior rttvar and srtt; and the srtt is defined over the S and prior srtt. Thus, going from the inside out, we begin by characterizing the srtt; then we use that analysis to aid our characterization of the rttvar; and finally we bring it all together to analyze the rto.

All of our work will be done under the "steady-state" assumption defined below. The purpose of this assumption is to define what it means for the network to exhibit bounded amounts of oscillation. Note however that this assumption does not limit the *rate* of oscillation in the sample values, for example, it does not require that the samples be drawn from the image of some Lipschitz continuous function.

Definition 2 (c/r Steady State)**.** *Let $c, r > 0$ be rationals and suppose that $\mathsf{S}_i, \mathsf{S}_{i+1}, \ldots, \mathsf{S}_{i+n} \in [c-r, c+r]$. Then we refer to the samples S_j for $j = i, \ldots, i+n$ as being in a c/r* steady-state.

For example, if $\mathsf{S}_i, \mathsf{S}_{i+1}, \ldots, \mathsf{S}_{i+n}$ are drawn from the uniform distribution over $[12.3, 75]$ms, then they are in a $43.65/31.35$ steady-state. Since every finite set achieves both a minimum and a maximum, all finite sequences of samples are technically speaking steady-state sequences, however, the same cannot be said for infinite sequences, such as the infinite sequence $\mathsf{S}_j = 12.3 + 2j$ for $j \in \mathbb{N}$.

Without loss of generality, for the rest of this section we will assume samples $\mathsf{S}_i, \mathsf{S}_{i+1}, \ldots, \mathsf{S}_{i+n}$ are in a c/r steady-state. We say WLOG because, we will consider both $n \in \mathbb{N}$ and the limit as $n \to \infty$. We begin by analyzing the srtt. Recall, $\mathsf{srtt}_i = \mathsf{S}_i$ if $i = 1$ else $(1-\alpha)\mathsf{srtt}_{i-1} + \alpha\mathsf{S}_i$. Since $\mathsf{S}_i \in [c-r, c+r]$:

$$(1-\alpha)\mathsf{srtt}_{i-1} + \alpha(c-r) \leq \mathsf{srtt}_i \leq (1-\alpha)\mathsf{srtt}_{i-1} + \alpha(c+r) \tag{10}$$

Note that both bounds in Eqn. 10 have the shape $(1-\alpha)\mathsf{srtt}_{i-1} + \alpha C$ for some constant C. Recursing on this shape, we get the following.

Lemma 1. *Let $C \in \mathbb{Q}_+$ and suppose that for all natural $0 \leq k \leq n$, we have $f(k) = (1-\alpha)\mathsf{srtt}_{i-1} + \alpha C$. Then the following holds for all $0 \leq k \leq n$.*

$$\begin{aligned} f(k) &= (1-\alpha)^{k+1}\mathsf{srtt}_{i-1} + \big(\sum_{j=0}^{k}(1-\alpha)^j\alpha\big)C \\ &= (1-\alpha)^{k+1}\mathsf{srtt}_{i-1} + \big((\alpha-1)(1-\alpha)^k + 1\big)C \end{aligned} \tag{11}$$

Applying Lemma 1 to Eqn. 10 we get the following.

Theorem 2. *Suppose $\mathsf{S}_i, \mathsf{S}_{i+1}, \ldots, \mathsf{S}_{i+n}$ are in a c/r steady-state. Then $L \leq \mathsf{srtt}_{i+n} \leq H$ where ...*

$$\begin{aligned} L &= (1-\alpha)^{n+1}\mathsf{srtt}_{i-1} + \big(1-(1-\alpha)^{n+1}\big)(c-r), \text{ and} \\ H &= (1-\alpha)^{n+1}\mathsf{srtt}_{i-1} + \big(1-(1-\alpha)^{n+1}\big)(c+r) \end{aligned} \tag{12}$$

In ACL2s, our proof strategy goes as follows. First we derive the closed form for $\sum_{j=0}^{k}(1-\alpha)^j\alpha$ and use it to rewrite srtt_{i+n} under the assumption that $\mathsf{S}_i = \mathsf{S}_{i+1} = \ldots = \mathsf{S}_{i+n}$. Then we show that in the c/r steady-state scenario, the lower bound L on the srtt_{i+n} is the value srtt_{i+n} would take if all the samples equaled $c-r$, and likewise the upper bound H is the value srtt_{i+n} would take if all the samples equaled $c+r$. Finally, we simplify and get the desired result. Finally, we look at the asymptotic case.

Theorem 3. $\lim_{n\to\infty} L = c-r$ *and* $\lim_{n\to\infty} H = c+r$.

Proof. For simplicity consider: $f(n) = (1-\alpha)^{n+1}\mathsf{srtt}_{i-1} + \big((\alpha-1)(1-\alpha)^n + 1\big)C$. Since $0 < \alpha < 1$, we know $0 < 1-\alpha < 1$, so by Theorem 1, $(1-a)^{n+1} \to 0$, and likewise for $(1-\alpha)^n$. This leaves only the term C. Thus $\lim_{n\to\infty} f(n) = C$. Since L is just $f(n)$ with $C = c-r$ and H is just $f(n)$ with $C = c+r$, the result immediately follows. □

Next, we consider the rttvar calculation. Recall that $\mathsf{rttvar}_i = \mathsf{S}_i/2$ if $i = 1$ else $(1-\beta)\mathsf{rttvar}_{i-1} + \beta|\mathsf{srtt}_{i-1} - \mathsf{S}_i|$, where $\beta < 1$ is constant in $\mathbb{Q}_+$. To simplify this equation, we will consider the case where $|\mathsf{srtt}_{i-1} - \mathsf{S}_i|$ is upper-bounded by some constant Δ. (Then we will show how to derive such a Δ).

Lemma 2. *Suppose $\mathsf{S}_i, \mathsf{S}_{i+1}, \ldots, \mathsf{S}_{i+n}$ are in a c/r steady-state, and $\Delta > 0$ upper-bounds $|\mathsf{srtt}_{j-1} - \mathsf{S}_j|$ for each $j = i, i+1, \ldots, i+n$. Then:*

$$\mathsf{rttvar}_{i+n} \leq (1-\beta)^{n+1}\mathsf{rttvar}_{i-1} + (1-(1-\beta)^{n+1})\Delta \tag{13}$$

Now we want to derive a bound on $|\mathsf{srtt}_{j-1} - \mathsf{S}_j|$. Note that this bound is at most:

$$\max\{|L-(c+r)|, |H-(c-r)|\} \tag{14}$$

If the larger of the two options is $|H-(c-r)|$ we derive the following.

$$\begin{aligned}\Delta &= |(1-\alpha)^{j+1}\mathsf{srtt}_{i-1} + (1-(1-\alpha)^{j+1})(c+r) - (c-r)| \\ &= |(1-\alpha)^{j+1}\mathsf{srtt}_{i-1} + (c+r) - (1-\alpha)^{j+1}(c+r) - c + r| \\ &= |(1-\alpha)^{j+1}\mathsf{srtt}_{i-1} + 2r - (1-\alpha)^{j+1}(c+r)|\end{aligned} \tag{15}$$

By Theorem 1, clearly $\lim_{j\to\infty}\Delta = 2r$. On the other hand, if the larger option is $|L-(c+r)|$, we derive $\Delta = |(1-\alpha)^{j+1}\mathsf{srtt}_{i-1} - 2r + (1-\alpha)^{j+1}(c-r)|$ which asymptotes at $|-2r| = 2r$. So either way, as j grows $\to\infty$, Δ converges to $2r$. Now suppose $\Delta = 2r$. Then the upper bound on srtt_{i+n} from Eqn 13 is $(1-\beta)^{n+1}\mathsf{rttvar}_{i-1} + (1-(1-\beta)^{n+1})2r$. Which gives us the following.

Theorem 4. *Suppose* $\mathsf{S}_i, \mathsf{S}_{i+1}, \ldots, \mathsf{S}_{i+n}$ *are in a* c/r *steady-state. Then there exists an upper bound on* rttvar_{i+n} *which, as* $n\to\infty$*, converges to* $2r$*.*

Proof. Follows from Theorem 1 because $\lim_{n\to\infty}(1-\beta)^{n+1}\mathsf{rttvar}_{i-1} = \lim_{n\to\infty}(1-\beta)^{n+1}2r = 0$. □

Finally, we turn our attention to the rto calculation. Recall, $\mathsf{rto}_i = \mathsf{srtt}_i + \max(G, 4\cdot\mathsf{rttvar}_i)$ for some constant G. On the one hand, if the rttvar is consistently very small (less than 1/4 of G) then clearly the rto is bounded by $[L+G, H+G]$. In this case, if $G > 2r$, we are assured that timeouts will never happen. But what if G is small relative to the radius of the steady-state interval? If $G < 2r$ and the rttvar can achieve a value $\leq G$ then a timeout can occur. And in fact, we can easily construct a scenario where exactly this happens infinitely many times.

For the pathological scenario, suppose that every 100^{th} sample equals 75, while all the rest equal 60. Clearly the samples are in a $67.5/7.5$ steady-state. At the spikes (where $\mathsf{S}_{i+100n} = 75$), $\mathsf{srtt} \approx 61.88, \mathsf{rttvar} \approx 3.75$, and $\mathsf{rto} \approx 61$. Since $61 < 75$, a timeout occurs. There are infinitely many "spikes" where timeouts occur. However, when we simulate a scenario where the samples are uniformly random over $[c-r, c+r]$, little to no timeouts occur. Both scenarios are illustrated below in Fig. 2.[8]

The RTO calculation is specified in RFC6298 [9] which says the constant G should be set to the "clock granularity" of the sender, in seconds. In the interest of avoiding excessive timeouts, a protocol implementer might want to consider adding the additional criteria that G should exceed 1/2 the diameter of the maximally large sample interval that they consider to be "stable". Depending on the nature and context of the protocol, this could be either a fixed or dynamic value. However, caution should be taken on the other hand to ensure G is not too large, since timeouts *should* occur when there really is congestion on the network, in order to avoid congestion collapse. Also, a dynamic value of G could exacerbate the risk of choosing too large of a timeout value, and might even be vulnerable to targeted manipulation.

As we alluded when introducing Def 2, one interesting direction for future research is to investigate minimal, sufficient analytic conditions to ensure timeouts do not occur. For example, it might be sufficient to require the samples be drawn from the image of a Lipschitz continuous function, and then to require some relationship between the Lipschitz bound, the choice of β, and the radius r. Although this is purely speculative, it is certainly the case that the derivative of the samples must be taken into account when analyzing the magnitude of the rttvar, providing multiple interesting directions for future research.

[8] Adapted from Fig. 3 of [4], first published in volume 14067, page 56, 2023, by Springer Nature.

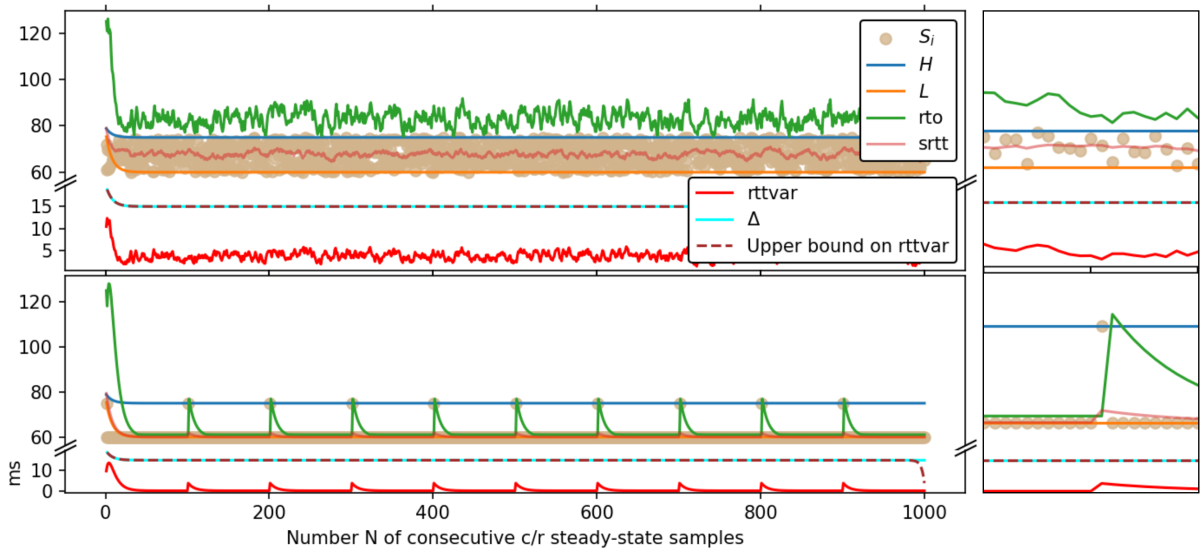

Figure 2: On the left are two 67.5/7.5 steady-state scenarios. On top the samples are drawn from the uniform distribution over the bounds, and timeouts rarely, if ever, occur. In the bottom (pathological) scenario, every 100$^{\text{th}}$ sample equals $c+r=75$ while the rest equal $c-r=60$, and at each "spike", a timeout occurs. There are infinitely many spikes, and one is shown on the right ($n=[350,450]$).

5 Discussion

In this work we considered multiple approaches to proving $\forall\alpha\in[0,1)::\lim_{n\to\infty}\alpha^n=0$. Our first was in ACL2(r) and resembled the obvious pen-and-paper proof, but could not be imported into an ACL2 environment. Our second and third were in ACL2s and required proving some arithmetic lemmas. Of these, the semi-automated version of the third is the most stylistically aligned with ACL2s because it takes advantage of automated termination analysis. The ACL2(r) proof has the lowest character count of all the proofs, and certifies the most quickly, but imports the most books and is not portable to ACL2. Considering books and portability, the semi-automated binomial proof is probably the best (see Tab. 1).

While working on the third approach we encountered an inconvenience in ACL2s: in ACL2s, `definecs` support `function-` and `body-contract-hints`, but `property` definitions do not. We found two ways to address this. The first was to redefine the problematic property as a decision procedure in a

Proof	LoC	Chars	Props/Thms	Functions	Books	Cert Time (s)
Real	161	4,224	17	1	5	0.58
Ceiling	408	16,103	20	3	0	64.17
Binomial (M)	154	5,652	22	1	2	2.54
Binomial (SA)	122	5,402	22	2	1	3.84

Table 1: Proof comparison. (M) refers to "manual" while (SA) refers to "semi-automatic". Lines of code and character count are computed without comments or empty lines, however, the proofs are not styled identically. Props/Thms counts instances of `property` and `defthm`, while Functions counts `definecs`, `definecds`, and `defuns`. Certification time is measured on a 16GB M1 Macbook Air.

`definec` with appropriate hints, and then write a new property saying that on all inputs, the procedure returns *true*. The second was simply to set `:check-contracts? nil`. To ameliorate this issue, we plan to add hints to `property` definitions in a future update to ACL2s.

While writing the ACL2s proofs, we took advantage of ACL2s features not available in ACL2 or ACL2(r). Most notably, we used termination analysis in the semi-automated binomial proof. There are also smaller ways that ACL2s was easier: type annotations improved the readability of our proofs, and contract-checking gave us some lemmas for free, including type-checking in the `xargs` to our `defun-sk`. In contrast, we had to manually prove contracts for d_α in our ACL2(r) proof, and it is harder to read than either ACL2s proof (with lengthier antecedents on `defthms`) because it lacks type annotations.

6 Conclusion

Recently, the Internet Engineering Task Force created a Usable Formal Methods Research Group, of which we are members, to integrate formal methods into the RFC drafting process. In many protocols, *performance* matters: we want to know how quickly the protocol achieves a desired outcome under load. Usually performance is studied using simulations or measurements, which can give a sense of how protocols behave "in the wild". But what about how protocols behave in the worst case? What if the worst case never happens in the measured environments or simulations? For this, we need a way to *prove* performance bounds, namely, formal methods. Meanwhile, real analysis provides a convenient framework with which to ask and answer questions about performance bounds in the long run. In our prior work [4], we used formal methods with real analysis to prove useful bounds on the internal variables of the RTO calculation. But we also ran into hurdles. We could not use the most obvious proof, which requires real numbers, because ACL2s only supports rationals. When we came up with an alternative approach (the "ceiling proof") it required us to convince ACL2s of numerous arithmetic lemmas. Only post-publication did we find a simpler solution, based on the binomial theorem.

As relatively novice users of ACL2s[9], our work leads us to identify three areas where we feel the ACL2 ecosystem could be improved to support work such as ours. First, ACL2 (and ACl2s in particular) could benefit from a richer, more comprehensively documented, and more easily searchable library of purely mathematical theorems, relating to the ceiling, floor, exponent, and logarithm, as well as metric spaces and limits. Searching for proofs is difficult enough, and ACL2 does not come with any kind of semantic proof search tool. And often, even when the desired theorems exist in the ACL2 books, they are unmentioned in the documentation. For example, the documentation on "arithmetic" does not mention the RTL books, and neither does the documentation on "math". Moreover, the rewrite rules from different libraries may conflict, so even if you find the desired theorems, importing them into a singular environment may be non-trivial. It would also be useful to have more mathematics formalized in ACL2s, so as to avoid additional proof obligations (for function contracts and termination) when using imported books. Second, ACL2(r) could benefit from the addition of the generic exponent and logarithm. This could be done using the translational Lemma given in §2. Third, and most importantly, though ACL2(r) and ACL2 have incompatible theories, it is nevertheless true that certain kinds of theorems over the reals should hold over the rationals, because the rationals are dense in the reals. It would useful to have a kind of "bridge" between ACL2(r) and ACL2, by which the user could justify that a given theorem, if true over the reals, must also hold over the rationals; prove the theorem in ACL2(r); and then import the theorem, using its "justification", into ACL2. Hopefully our experience provides insight for future work in both protocol analysis and extending the ACL2 ecosystem.

[9] Excluding the second author.

Acknowledgments. The first author would like to thank Ruben Gamboa for providing technical support in ACL2(r) and suggesting Lemma R1, and Ankit Kumar for providing technical support in ACL2s.

References

[1] Harsh Chamarthi, Peter C. Dillinger, Panagiotis Manolios & Daron Vroon (2011): *The "ACL2" Sedan Theorem Proving System*. In: *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, doi:10.1007/978-3-642-19835-9_27.

[2] Peter C. Dillinger, Panagiotis Manolios, Daron Vroon & J Strother Moore (2007): *ACL2s: "The ACL2 Sedan"*. In: *International Conference on Software Engineering (ICSE)*, doi:10.1109/ICSECOMPANION.2007.14.

[3] Ruben A Gamboa & Matt Kaufmann (2001): *Nonstandard analysis in ACL2*. *Journal of automated reasoning* 27, pp. 323–351, doi:10.1023/A:1011908113514.

[4] Max von Hippel, Kenneth L. McMillan, Cristina Nita-Rotaru & Lenore Zuck (2023): *A Formal Analysis of Karn's Algorithm*. In: *2023 International Conference on NETworked sYStems (NETYS)*, Springer, doi:10.1007/978-3-031-37765-5_4.

[5] Phil Karn & Craig Partridge (1987): *Improving round-trip time estimates in reliable transport protocols*. *ACM SIGCOMM Computer Communication Review* 17(5), pp. 2–7, doi:10.1145/55483.55484.

[6] Ankit Kumar & Panagiotis Manolios (2021): *Mathematical Programming Modulo Strings*. In: *2021 Formal Methods in Computer Aided Design (FMCAD)*, IEEE, pp. 261–270, doi:10.34727/2021/isbn.978-3-85448-046-4_36.

[7] Panagiotis Manolios & Daron Vroon (2006): *Termination analysis with calling context graphs*. In: *Computer Aided Verification: 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006. Proceedings 18*, Springer, pp. 401–414, doi:10.1007/11817963_36.

[8] Oded Padon, Kenneth L McMillan, Aurojit Panda, Mooly Sagiv & Sharon Shoham (2016): *Ivy: safety verification by interactive generalization*. In: *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 614–630, doi:10.1145/2908080.2908118.

[9] V. Paxson, M. Allman, J. Chu & M. Sargent (2011): *Computing TCP's Retransmission Timer*. `https://datatracker.ietf.org/doc/html/rfc6298`. Accessed 15 June 2023.

[10] Brian M. Scott (https://math.stackexchange.com/users/12042/brian-m scott): *Nested Division in the Ceiling Function*. Mathematics Stack Exchange. Available at `https://math.stackexchange.com/q/233684`. (version: 2012-11-09).