A Formal Analysis of the Compact Position Reporting Algorithm

Aaron Dutle[1], Mariano Moscato[2], Laura Titolo[2], and César Muñoz[1]

[1] NASA Langley Research Center, Hampton, VA, US,
{aaron.m.dutle,cesar.a.munoz}@nasa.gov
[2] National Institute of Aerospace, Hampton, VA, US,
{mariano.moscato,laura.titolo}@nianet.org

Abstract. The Compact Position Reporting (CPR) algorithm is a safety-critical element of the Automatic Dependent Surveillance - Broadcast (ADS-B) protocol. This protocol enables aircraft to share their current states, i.e., position and velocity, with traffic aircraft in their vicinity. CPR consists of a collection of functions that encode and decode aircraft position data (latitude and longitude). Incorrect position reports from CPR have been reported to the American and European organizations responsible for the ADS-B standard. This paper presents a formal analysis of the CPR algorithm in the Prototype Verification System (PVS). This formal analysis shows that the published requirements for correct decoding are insufficient, even if computations are assumed to be performed using exact real arithmetic. As a result of this analysis tightened requirements are proposed. These requirements, which are being considered by the standards organizations, are formally proven to guarantee correct decoding under exact real arithmetic. In addition, this paper proposes mathematically equivalent, but computationally simpler forms to several expressions in the CPR functions in order to reduce imprecise calculation.

1 Introduction

Automatic Dependent Surveillance - Broadcast (ADS-B) is arguably the most important change to the operation of aircraft in national and international airspace since the introduction of radar. The Federal Aviation Administration has mandated that ADS-B out capability be installed on almost all general aviation aircraft for most classes of airspace before the year 2020. ADS-B allows for a wide variety of information to be broadcast from an aircraft to any nearby receiver, enabling many new capabilities, including increased situational awareness for pilots. To enable this technology, the industry and regulatory agencies agreed on a standard message format based on an existing transponder[3]; the 1090 Mhz Mode-S Extended Squitter. The broadcast message is 112 bits, of which 56 bits are the data frame, the rest being aircraft identification, message

[3] In fact, there are several allowable transponders and formats, though the majority of current applications use the 1090 ES message described here.

type, and parity check information. When the data frame is a position message, 21 bits go to transmitting status information and altitude, leaving 35 bits total for latitude and longitude. If raw latitude and longitude data were taken and approximated to 17 bits each, the resulting precision would be worse than 300 meters, which would not be useful for precise navigation.

To remedy this, an algorithm referred to as Compact Position Reporting (CPR) was developed to allow for more accurate position reporting. The general idea is as follows. Each direction (latitude and longitude) is divided into zones approximately 360 nautical miles long, and each zone into 2^{17} bins. The position broadcast corresponds to the centerline of the bin where the aircraft is currently located in. This corresponds to one position in each zone. Depending on the type of decoding being performed, the correct zone is then determined from either a previously known position (for local decoding) or from a matched pair of messages (for global decoding). This allows for position accuracy of approximately 5 meters in airborne applications. It should be noted that because the number of longitude degrees in 360 nautical miles differs based on latitude, the number of zones used for calculating the longitude message also depends on the latitude. The function that determines the number of longitude zones, named NL, can be calculated directly from the latitude, but in practice is determined from a pre-calculated lookup table.

Anecdotal evidence from pilots and manufacturers suggests that decoding of CPR messages can lead to incorrect position reports for target aircraft. A priori, these errors could stem from any number of places, including issues with the functions themselves, issues with the requirements under which the functions may be used, numerical computation errors, environmental factors, or any other number of unknown causes. The work described here addresses the first three of these possibilities.

On the practical side, this paper has two significant contributions that are presented in the form of recommendations to the standards organizations in charge of the ADS-B protocol.[4] These recommendations aid in more reliable usage and implementation of the CPR algorithm and do *not* alter the logic of the algorithm in its pure mathematical form. Hence, they do not impact implementations that are already in place and operating reliably. The first recommendation is a tightening of the requirements on conditions for reliable decoding. These strengthened requirements were discovered during the interactive construction of the proof of correctness, during which a class of examples meeting the published requirements for correct decoding were found to give significantly incorrect answers. The second recommendation consists of a collection of simplified expressions for computations performed in the algorithm. These simplifications reduce the numerical complexity of the expressions. This second class of results are intended to aid future implementors of the algorithm in producing simpler and more reliable code.

From the theoretical standpoint, the main contribution of this work is a formal analysis of the CPR algorithm in the Prototype Verification System

[4] These organizations are RTCA in the US and EUROCAE in Europe.

(PVS) [5]. The analysis includes a mechanically verified proof that the encoding and decoding functions work as designed under the proposed tightened requirements, and with the assumption of real number computation. This formal analysis is meant to increase confidence that the functions themselves, in their pure mathematical forms, are correct. In addition, the formal specification itself is done in a way that allows the CPR algorithm to be executed in a number of different computational modes. By instantiating a parameter, any of the CPR functions can be evaluated in single precision floating-point, double precision floating-point, or exact rational arithmetic. Transcendental functions that occur in the algorithm can be evaluated in either one of the floating-point implementations or to a user specified precision. This allows for simple comparison of the algorithm's results under different computation models, without the need to write separate versions of the algorithm for each model.

The remainder of the paper is organized as follows. Section 2 presents the formal development of the CPR algorithm, including its main properties and some rationale for how the requirements for proper decoding arise. Section 3 details the main practical results from the formal analysis, including the tightened requirements for proper decoding and a number of computational simplifications or the CPR algorithm. Section 4 discusses a method used to animate the specification of CPR in different computational modes. Finally, Section 5 concludes this work.

The formulas and theorems presented in this paper are written in PVS. For readability, this paper uses mathematical notation as opposed to PVS syntax. The formal development is available at `http://shemesh.larc.nasa.gov/fm/CRP` and requires the latest version the NASA PVS Library, which is available at `http://github.com/nasa/pvslib`.

2 The Compact Position Reporting Algorithm

This section presents the formal development of the CPR algorithm, which closely follows its standard definition in [6]. The CPR algorithm allows for three different classes of position messages known as *coarse*, *airborne*, and *surface*, which provide accuracies of approximately 165 m, 5 m, and 1.3 m, respectively. For simplicity, the analysis presented in this paper only considers airborne messages. The analysis of the NL function, as well as the mathematical simplifications of computations, apply to all three versions. The requirement tightening, as well as the formal verification of correct decoding, applies only to the airborne version. Generalization to the other classes of messages is not theoretically challenging but would require a non-trivial amount of work.

The principle of the CPR encoding and decoding functions is that transmitting the entire latitude and longitude measurement of a target would be a) a prohibitively large message for sufficient precision, and b) wasteful, as the higher order bits of such a transmission are very unlikely to change over a short period of time. To remedy both of these issues, CPR transmits a version of the lower order bits, and uses two different techniques to recover the higher order bits.

To accomplish this, CPR divides each of latitude and longitude into a number of equally sized zones. The number of these zones depends on if the message is an *even* or *odd* format, and when encoding longitude, on the current latitude of the target. Each zone is then divided into 2^{17} bins. The transmitted information is the number corresponding to the bin that the target is currently in. The difficult part of the process is then determining the correct zone. This is done in the *local* decoding case by identifying a reference position close enough to the target that only one of the possibly bins is feasible, and in the *global* decoding case by using an odd and an even message, and employing the difference in size of odd versus even zones.

2.1 Number of Longitude Zones

As previously stated the number of longitude zones depends on both the format (even or odd) and the present latitude of the target. Note that if the number of zones used for longitude encoding were a constant with respect to latitude, the size of one such zone would vary significantly between the poles and the equator. This would make decoding much more difficult at the poles, since the zone number would be necessarily change more rapidly at high latitudes. In order to alleviate this, the number of longitude zones is variable depending on the latitude. This NL value is meant to keep the size of a zone nearly constant.

For latitude, the size of an even zone is 6 degrees, while the odd zone is slightly larger, at 360/59 degrees. To keep longitude similarly spaced, there are 60 even longitude zones at the equator, and one fewer odd zones (the number of odd zones is always one fewer than the number of even zones). This even zone is approximately 360 NMI wide. The number of even zones drops to 59 at the circle of latitude where 59 zones of size 360 NMI suffice to cover the distance (assuming a spherical earth). More precisely, the number of longitude zones (or NL value) corresponding to a specific latitude lat is given by

$$\mathrm{NL}(lat) = \begin{cases} 59 & \text{if } lat = 0, \\ \left\lfloor 2\pi \left(\arccos\left(1 - \frac{1-\cos\left(\frac{\pi}{30}\right)}{\cos^2\left(\frac{\pi}{180}|lat|\right)} \right) \right)^{-1} \right\rfloor & \text{if } |lat| < 87, \\ 1 & \text{if } |lat| > 87, \\ 2 & \text{otherwise.} \end{cases} \tag{1}$$

In practice, computing this function is inefficient and would be burdensome to perform each time an encoding is done. Instead, a lookup table of *transition latitudes* is pre-calculated, and the NL value is determined from this table. In PVS, the NL table is specified as follows.

$$
\begin{aligned}
\text{NL\_Table}(lat) = \ & \texttt{if } |lat| > transition(2) \texttt{ then } 1 \\
& \texttt{elsif } |lat| > transition(3) \texttt{ then } 2 \\
& \texttt{elsif } |lat| > transition(4) \texttt{ then } 3 \\
& \quad \vdots \\
& \texttt{elsif } |lat| > transition(59) \texttt{ then } 58 \\
& \texttt{else } 59 \\
& \texttt{endif}.
\end{aligned}
$$

The transition latitudes are given for a value nl from 2 to 59 by the following formula.

$$
transition(nl) = \frac{180}{\pi} \arccos\left(\sqrt{\frac{1-\cos(\pi/30)}{1-\cos(2\pi/nl)}}\right). \tag{2}
$$

The following theorem about the correctness of this table is proven in PVS.

Theorem 1. *For every latitude value lat,*

$$
NL(lat) = NL\_Table(lat).
$$

During the process of encoding, extra precaution must be taken to ensure that the NL value used for the longitude encoding is consistent with the latitude broadcast. To do so, the latitude message to be broadcast is decoded onboard, and *this* latitude is used to determine the NL value, ensuring that message sent is consistent.

2.2 Encoding

As mentioned in Section 1, the position message consists of 35 bits of information. The first bit is used to describe the *format* of the message. The message called even if the bit is 0, and odd if the bit is 1.

Here, and throughout all computation in CPR, the mod function is defined by

$$
\mathrm{mod}(x, y) = x - y\left\lfloor \frac{x}{y} \right\rfloor.
$$

While this is fairly standard for mathematics, it differs from the version used in practice in standard programming languages, where the function is generally restricted to integers.

Let $dlat_i = 360/(60 - i)$, where i is the format bit of the message to be sent. The value of $dlat_i$ is the size of a latitude zone. Next, for a latitude value lat, compute

$$
YZ_i = \left\lfloor 2^{17} \frac{\mathrm{mod}(lat, dlat_i)}{dlat_i} + \frac{1}{2} \right\rfloor. \tag{3}
$$

The latitude message, $\widehat{YZ}_i$, is then the last 17 bits of this value. That is,

$$\widehat{YZ}_i = \text{mod}(\, YZ_i, 2^{17}). \tag{4}$$

In Formula (3), $\text{mod}(lat, dlat_i)$ corresponds to the distance that lat is from the bottom of a zone edge. Thus $\frac{\text{mod}(lat, dlat_i)}{dlat_i}$ denotes the fractional amount that lat is into this zone. Multiplying by 2^{17} gives a value between 0 and 2^{17}, while $\lfloor x + \frac{1}{2} \rfloor$ rounds a number x to the nearest integer. The interval of latitudes inside a zone that are mapped to a particular number is referred to as a *bin*, and the number they map to as the *bin number*. The latitude to be recovered is in the center of this interval, and is referred to as the *bin centerline*. The final truncation to 17 bits to determine $\widehat{YZ}_i$ may appear to discard some information, but in actuality only affects half of a bin at the top of a zone, and is accounted for by the adjacent zone.

In order to compute the longitude portion of the message, the NL value of the encoded latitude must be determined. To do so, the latitude that is intended to be decoded is computed as

$$\text{rlat} = dlat_i \left(\left\lfloor \frac{lat}{dlat_i} \right\rfloor + \frac{YZ_i}{2^{17}} \right).$$

The NL value of $rlat$ is then used to compute the longitude equivalent of $dlat_i$ as follows.

$$dlon_i = 360 / \max\{1, NL(rlat) - i\}. \tag{5}$$

Note that the denominator in the above expression uses the max operator for the case of latitudes beyond $\pm$ 87 degrees, where there is only one longitude zone. In this case the even and odd longitude encodings are identical.

With $dlon_i$ calculated, the encoding of a longitude lon is nearly identical to that of latitude.

$$XZ_i = \left\lfloor 2^{17} \frac{\text{mod}(lon, dlon_i)}{dlon_i} + \frac{1}{2} \right\rfloor. \tag{6}$$

The longitude message is the final 17 bits of this value.

$$\widehat{XZ}_i = \text{mod}(XZ_i, 2^{17}). \tag{7}$$

The final message to be broadcast is then the concatenated string of bits $(i, \widehat{YZ}_i, \widehat{XZ}_i)$. In theory, it would be desirable to have the messages sent strictly alternate between odd and even. In practice, the format for broadcast is chosen by any number of methods (including randomly) to ensure an equiprobable distribution.

It is worth noting that every latitude lat can be exactly and uniquely determined by the following formula.

$$lat = dlat_i \left(\left\lfloor \frac{lat}{dlat_i} \right\rfloor + \frac{2^{17} \text{mod}(lat, dlat_i)/dlat_i}{2^{17}} \right). \tag{8}$$

The only difference between this and the value intended to be recovered is in the rounding of $2^{17}\mathrm{mod}(lat, dlat_i)/dlat_i$ to the nearest integer, which induces an error of at most 1/2. Hence the upper bound for the difference between a latitude and its correctly encoded and decoded value is $dlat_i/2^{18}$. Similarly, a longitude and its recovered value should differ by no more than $dlon_i/2^{18}$.

The formal development includes specification of the encoding algorithm as a single function *encode* that takes as parameters the format i, and lat, lon, the latitude and longitude to encode, and returns the pair $(\widehat{YZ}_i, \widehat{XZ}_i)$ containing the encoded latitude and longitude. The following lemma, formally proven in PVS, ensures that the encoding fits into the available space for broadcast.

Theorem 2. *For all $i \in \{0, 1\}$, latitudes lat and longitudes lon, if $(Y, X) = encode(i, lat, lon)$, then Y and X integers and*

$$0 \leq X, Y < 2^{17}.$$

2.3 Local Decoding

Since a broadcast message corresponds to a position inside each zone, in order to recover the correct position, one needs only to determine which zone is the correct one, and in the case of longitude, how many zones there are.

Local decoding does this using a reference position that is known to be near the broadcast position. This can be a previously decoded position, or known by some other means. The concept is simple, and uses the observation that the interval one zone wide centered around any position contains exactly one point corresponding to each bin centerline.

From this reasoning, it would seem to follow that if the target and the reference position are separated by at most half the length of a zone, the decoding should be reliable. This is the requirement given in the standards document [6] for local decoding. However, during the formal analysis, it was discovered that this is too generous, as proven in Theorem 3 below.

The local decoding uses the following formula to calculate the zone index number j of the target using the format i, the latitude message $\widehat{YZ}_i$, and a reference latitude lat_{ref}.

$$j = \left\lfloor \frac{lat_{\mathrm{ref}}}{dlat_i} \right\rfloor + \left\lfloor \frac{1}{2} + \frac{\mathrm{mod}(lat_{\mathrm{ref}}, dlat_i)}{dlat_i} - \frac{\widehat{YZ}_i}{2^{17}} \right\rfloor. \tag{9}$$

The first term in this sum calculates which zone the reference latitude lies in, while the second term adjusts it by -1, 0, or 1 based on the difference between the reference latitude and the broadcast message. This value is then used to compute the recovered latitude $rlat$ using the following formula.

$$rlat = dlat_i \left(j + \frac{\widehat{YZ}_i}{2^{17}} \right). \tag{10}$$

This decoded latitude is used to determine the NL value used for encoding the longitude, which is then used to determine the value of $dlon_i$ by Formula (5). Using $dlon_i$, a reference longitude lon_{ref}, and the longitude message $\widehat{XZ}_i$, the longitude zone index m and recovered longitude $rlon$ are determined nearly identically to the latitude case.

$$m = \left\lfloor \frac{lon_{\text{ref}}}{dlon_i} \right\rfloor + \left\lfloor \frac{1}{2} + \frac{\text{mod}(lon_{\text{ref}}, dlon_i)}{dlon_i} - \frac{\widehat{XZ}_i}{2^{17}} \right\rfloor. \tag{11}$$

$$rlon = dlon_i \left(m + \frac{\widehat{XZ}_i}{2^{17}} \right). \tag{12}$$

Local decoding is specified as a pair of functions $Rlat_i$ and $Rlon_i$. The function $Rlat_i$ takes as input a reference latitude lat_{ref}, a format i, and a non-negative integer Y less than 2^{17} meant to be an encoded latitude. The function $Rlon_i$ takes an entire reference position $lat_{\text{ref}}, lon_{\text{ref}}$, a format i and a pair Y, X of non-negative integers at most 2^{17} meant to be the encoded pair. The longitude decoding requires the latitude inputs in order to calculate the correct NL value to decode with.

The two main theorems concerning local decoding are with respect to the requirements for correct decoding. The first states that the published requirements are not sufficient for local decoding.

Theorem 3. *For each format i, there exist latitudes lat, lat_{ref} with $|lat - lat_{\text{ref}}| < \frac{dlat_i}{2}$, but*

$$|lat - Rlat_i(lat_{\text{ref}}, i, \widehat{YZ}_i)| > 5.9,$$

where $(\widehat{YZ}_i, \widehat{XZ}_i) = encode(i, lat, lon)$.

The value of 5.9 is in degrees latitude, which at a longitude of 0 is more than 300 nautical miles. This theorem is formally proven in PVS by giving actual latitude values that decode incorrectly. One such pair, for even encoding, is

$$lat = 71582788 * 360/2^{32} \approx 5.99999997765,$$
$$latref = 35791394 * 360/2^{32} \approx 2.99999998882.$$

The next theorem states that local decoding does work properly for the set of tightened requirements, which reduce the bound between position and reference by 1/2 bin.

Theorem 4. *For all pairs of positions $(lat, lon), (lat_{\text{ref}}, lon_{\text{ref}})$, let $(\widehat{YZ}_i, \widehat{XZ}_i) = encode(i, lat, lon)$. If*

$$|lat - lat_{\text{ref}}| < \frac{dlat_i}{2} - \frac{dlat_i}{2^{18}},$$

then

$$|lat - Rlat_i(lat_{\text{ref}}, i, \widehat{YZ}_i)| \leq \frac{dlat_i}{2^{18}},$$

Furthermore, if $dlon_i$ is calculated using this decoded latitude, and

$$|lon - lon_{\text{ref}}| < \frac{don_i}{2} - \frac{dlon_i}{2^{18}},$$

then

$$|lon - Rlon_i(lat_{\text{ref}}, lon_{\text{ref}}, i, \widehat{YZ}_i, \widehat{XZ}_i)| \leq \frac{dlon_i}{2^{18}}.$$

2.4 Global Decoding

Global decoding is used when an approximate position for the target is unknown. This can occur when a target is first encountered, or when messages have not been received for a significant amount of time.

Similar to local decoding, the receiver must determine the correct zone in which the broadcast message lies, as well as (for longitude) the number of zones. Global decoding does this through means of a pair of messages, one of each format type. Using a method that is essentially the Chinese Remainder Theorem, the algorithm determines the number of *zone offsets* (the difference between an odd zone length and an even zone length) from the origin (either equator or prime meridian) to the broadcast position. This can be used to determine the zone for either message type, and hence used to decode either message. The most recently received message is used to provide more accurate information. Similar to the local decoding, it seems that this should tolerate pairs of positions separated by no more than half of a zone offset, since this is the critical parameter in the computation. The formal analysis shows that this is too generous, as proven in Theorem 5 below.

The first step in global decoding is to determine j, which is the number of zone offsets between the southern boundaries of the two encoded latitudes.

$$j = \left\lfloor \frac{59\widehat{YZ}_0 - 60\widehat{YZ}_1}{2^{17}} + \frac{1}{2} \right\rfloor.$$

In order to convert this into the correct zone index number for the even or odd message to be decoded, the positive value modulo $60 - i$ is calculated. This is then used to determine the recovered latitude, as follows.

$$rlat = dlat_i \left(\text{mod}(j, 60 - i) + \frac{\widehat{YZ}_i}{2^{17}} \right). \tag{13}$$

For global decoding of longitude, care must be taken that both the even end odd messages being used were calculated with the same number of zones. As such, *both* even and odd latitude messages are decoded, and the NL values for each are determined. If they differ, the messages are discarded and not decoded until further broadcast meet this criterion. In the case both recovered latitudes have the same NL value, longitude decoding proceeds as follows, where NL is the common NL value computed, $dlon_i$ is calculated according to Formula (5), and $n_i = \max\{NL - i, 1\}$.

Calculate m, the number of zone offsets between the western zone boundaries of the messages.

$$m = \left\lfloor \frac{(NL-1)\widehat{XZ}_0 - NL\widehat{XZ}_1}{2^{17}} + \frac{1}{2} \right\rfloor.$$

Convert this value to a zone index number by taking the positive value modulo n_i, and use this to determine the recovered longitude.

$$rlon = dlon_i \left(\mathrm{mod}(m, n_i) + \frac{\widehat{XZ}_i}{2^{17}} \right). \tag{14}$$

Global decoding is specified as a pair of functions $Rlat_g$ and $Rlon_g$. The function $Rlat_g$ takes as inputs a format i and natural numbers Y_0, Y_1 meant to be odd and even latitude messages. The function $Rlon_g$ takes as inputs a format i and four numbers Y_0, Y_1, X_0, X_1 meant to describe odd and even messages of latitude and longitude. Each latitude message is decoded, and the NL value computed. If the values do not match, the computation is aborted. If they do, the function returns both the even and odd decoded longitude.

As with local decoding, there are two accompanying theorems. In the following, the latitude zone offset is denoted by ZO_{lat}. This is calculated as $ZO_{lat} = dlat_1 - dlat_0$. Similarly, $ZO_{lon} = dlon_1 - dlon_0$ where it is assumed that the NL value used is known from the context.

Theorem 5. *For each format i, there exist latitudes lat_0, lat_1 with $|lat_0 - lat_1| < \frac{ZO_{lat}}{2}$, but*

$$|lat - Rlat_g(i, \widehat{YZ}_0, \widehat{YZ}_1)| > 5.9,$$

where $(\widehat{YZ}_j, \widehat{XZ}_j) = encode(j, lat, lon)$ for $j \in \{0, 1\}$.

Again, the units of 5.9 are degrees latitude, which corresponds to over 30 nautical miles at longitude 0. This theorem is formally proven in PVS by giving actual latitude values that decode incorrectly. One such pair, which decodes incorrectly using either format, is

$$lat_0 = 363373617 * 360/2^{32} \approx 30.4576247279,$$
$$lat_1 = 363980245 * 360/2^{32} \approx 30.5084716994.$$

The next theorem states that the tightened requirements, given by shrinking the bound by the size of one odd bin, suffice for proper global decoding.

Theorem 6. *For all pairs of positions $(lat_0, lon_0), (lat_1, lon_1)$, let $(\widehat{YZ}_j, \widehat{XZ}_j) = encode(j, lat, lon)$ for $j \in \{0, 1\}$. If*

$$|lat_0 - lat_1| < \frac{ZO_{lat}}{2} - \frac{dlat_1}{2^{17}},$$

then

$$|lat - Rlat_g(i, \widehat{YZ}_0, \widehat{YZ}_1)| \leq \frac{dlat_i}{2^{18}},$$

for each $i \in \{0, 1\}$. *Furthermore, if these decoded latitudes have a common NL value,* $dlon_i$ *is calculated using this value, and*

$$|lon - lon_{\text{ref}}| < \frac{ZO_{lon}}{2} - \frac{dlon_1}{2^{17}},$$

then

$$|lon - Rlon_g(i, \widehat{YZ}_0, \widehat{YZ}_1, \widehat{XZ}_0, \widehat{XZ}_1)| \leq \frac{dlon_i}{2^{18}}$$

for each $iin\{0, 1\}$.

These correctness theorems, while lacking the need for groundbreaking mathematical insight to formulate, are nonetheless long and difficult proofs to develop in an interactive proof system. For example, the proof of the correctness for only the longitude portion of the global correctness theorem is composed of 763 individual proof commands.

3 Practical Results

The main practical results of the formal analysis conducted are essentially in two categories. The first set of results, presented in Section 3.1, concerns the requirements for both local and global decoding. As discussed in Section 2.3 and Section 2.4, the formal analysis led to the discovery of examples that meet the stated algorithmic requirements for decoding, but decode incorrectly. A set of tightened requirements were discovered that are formally proven to guarantee correct decoding. In addition to the algorithmic requirements, an arguably *less* restrictive operation requirement is developed for global decoding. This proposed requirement allows for CPR applications for aircraft with a much wider performance envelope than the original specification, as well as a longer possible time delay between received messages.

The second set of results, in Section 3.2, examine expressions in the CPR algorithm, and give mathematically equivalent, but in a simpler or numerically more stable form. These equivalent expressions are meant to assist implementors of the CPR algorithm in creating more reliable code.

3.1 Decoding Requirements

The requirement stated in [6] for local decoding is that the reference position and the encoded position must be within 1/2 of a zone to guarantee correct decoding. As mentioned in Section 2, this stems from the fact that an interval one zone long, centered at the reference position, encounters exactly one bin centerline for each possible broadcast message, so only one recovered position is possible. While this statement is true, it is not necessarily true that the position being within 1/2 zone from the reference position ensures that the corresponding *bin centerline* is within 1/2 zone of the reference position. For example, if the reference position is slightly above a bin centerline, then the half-bin at the

bottom of the 1 zone length interval centered around the reference position is mapped to a bin centerline that occurs *outside* of this one zone region. The bin centerline with the same number, but lying inside the one zone region, occurs at the *top* of this region. Hence local decoding in this case is inaccurate by the size of one zone, approximately 360 nautical miles, as is the case in the example after Theorem 3.

During the formal analysis, several examples illustrating this phenomenon were discovered, and this discovery led to the tightened requirement of the target position and reference position being required to be separated by no more than half a zone minus half of a bin for reliable local decoding.

For global decoding, the requirement in [6] is that the two messages used are received within ten seconds of each other. This is based on two conditions. The first condition is a restriction on the performance limits of the aircraft to which the standard applies. The second condition is an algorithmic restriction. The document states that the positions for the odd and even messages be separated by no more than half of a zone offset to ensure reliable decoding. As with local decoding, this algorithmic condition is nearly correct, but fails to account for the positions not being on the bin centerline. The *correct* requirement is that the bin centerlines of the encoded positions be within 1/2 zone offset. Since a position is at most half of a bin size away from the corresponding centerline, shrinking the original requirement by one odd bin size is sufficient to guarantee correct global decoding. As with the local decoding, examples were discovered that meet the published algorithmic requirement, but decode incorrectly by the length of one zone, approximately 360 nautical miles, as is the case in the example after Theorem 5.

The published global decoding requirement enforces the closeness of the original positions of the two messages by means of a limit on the time between two messages, paired with a limit on the speed of the aircraft. While this is a testable and practical method of enforcing the algorithmic requirements, it limits the applications that can be correctly decoded due to speed assumptions[5], while artificially limiting the time between messages for slow moving targets.

To loosen this restriction, while still providing a testable and practical method for guaranteeing that the even and odd pair of messages meet the global decoding algorithmic requirements, the following alternative requirement is proposed.

The receiver waits for three alternating messages, either even-odd-even or odd-even-odd, where it is known (through a time restriction or some other means) that the first and last messages were broadcast without having travelled more than 1/2 zone. In addition, the difference between the values of the first and last messages transmitted should be less than 1000 (modulo 2^{17}). The second condition ensures that the bookend messages were broadcast within 1/2 zone offset (minus an odd zone) of each other, unless they are separated by a full zone, which is impossible by the first condition. For longitude decoding, the

[5] This is an issue that affects the usability of ADS-B for hypersonic aircraft and for sub-orbital applications, both of which are poised to become more ubiquitous in the near future.

NL value of all three latitudes messages must also stay constant. The proposed requirement allows for a much longer time frame to collect messages, even with an increased performance threshold for the target. It also more directly enforces the actual algorithmic requirement.

3.2 Numerical Simplifications

In addition to the formal specification and proof of the algorithm with the tightened requirements, the formal analysis revealed several expressions in the CPR algorithm that can be simplified or rewritten in a way that is mathematically equivalent, but numerically simpler. Each pair of equivalent formulas was specified in PVS, and proven to be equal.

The formula for calculating the NL table, used as a lookup-table for calculating NL values for a latitude is given in Formula (2). An equivalent version, removing four operations in total, is defined as follows.

$$lat_{NL}(nl) = \frac{180}{\pi} \arccos\left(\frac{\sin(\pi/60)}{\sin(\pi/nl)}\right). \tag{15}$$

The remainder of the simplifications essentially rely on two observations. The first observation is that when the mod operator is divided by its second argument, a cancellation can be made instead of a division. That is,

$$\frac{\mathrm{mod}(a,b)}{b} = \frac{a - b * \left\lfloor \frac{a}{b} \right\rfloor}{b} = \frac{a}{b} - \left\lfloor \frac{a}{b} \right\rfloor. \tag{16}$$

The second observation is that the floor function and addition of integers are commutative. That is, for any number x and any integer z,

$$\lfloor z + x \rfloor = z + \lfloor x \rfloor. \tag{17}$$

Using the simplifications of Formula (16) and Formula (17) on the local decoding formulas (9) and (11) yields

$$j = \left\lfloor \frac{1}{2} + \frac{lat_{\mathrm{ref}}}{dlat_i} - \frac{\widehat{YZ}_i}{2^{17}} \right\rfloor, \tag{18}$$

and

$$m = \left\lfloor \frac{1}{2} + \frac{lon_{\mathrm{ref}}}{dlon_i} - \frac{\widehat{YZ}_i}{2^{17}} \right\rfloor. \tag{19}$$

The most significant simplification is in the encoding algorithm, and applies to both latitude and longitude. Let x denote the position, either latitude or longitude, and let dl denote $dlat_i$ or $dlon_i$ accordingly, then Formula (3) and Formula (6) can be simplified as follows.

$$\left\lfloor 2^{17} \frac{\mathrm{mod}(x, dl)}{dl} + \frac{1}{2} \right\rfloor = \left\lfloor 2^{17} \frac{x}{dl} + \frac{1}{2} \right\rfloor - 2^{17} \left\lfloor \frac{x}{dl} \right\rfloor. \tag{20}$$

The simplifications presented in this section reduce the number of operations overall and remove computation of several expressions that strictly cancel mathematically. For instance, on the right hand side of Formula (20), once the term x/dl is computed, the subtracted term can be calculated exactly as an integer.

4 Animation of the CPR Specification

In contrast to a programming language, PVS is designed to manipulate and reason about real numbers. For example, the value of π in PVS is the real, irrational, transcendental number that exactly relates a diameter to a circumference. In this paper, the exact, ideal version of an algorithm or quantity in referred to as the *platonic* version. For instance, the functions presented in Section 2 correspond to the platonic version of CPR. However, since the CPR algorithm is implemented on actual hardware, numerical imprecisions are unavoidable. In addition to the formal verification of the CPR algorithm, the formal specification of CPR was used to compare in a set of inputs the evaluation of the platonic algorithm versus the algorithm implemented in both single and double precision floating-point.

To achieve this goal, the CPR specification is written in a way that arithmetic operators can be ground-evaluated in PVSio [4] using *semantic attachments* [1]. PVSio allows for the evaluation of PVS functional specifications using the ground evaluator. A semantic attachment is a Lisp function that is called by the ground evaluator when a particular function is not evaluable in PVS, e.g., square root, trigonometric functions, etc. Since semantic attachements are external to the PVS logic, ground evaluations in PVSio may not be logically sound. However, PVSio provides a practical way to quickly test a PVS specification on concrete values. See [2] for more details.

PVSio and, in particular, semantic attachments enable the evaluation of CPR functions on concrete inputs using different computation models, e.g., real arithmetic, single or double floating-point arithmetic, etc. Using this method on the latitude encoding, it has been checked that the right-hand side of Formula (20) performed in double precision floating-point agrees with the platonic calculation for all angular weighted binary (AWB) latitudes [3]. These are latitudes of the form $n \cdot \frac{360}{2^{32}}$ with n a whole number, and are a widely used format for providing position. Furthermore, a test of the standard formulation of the latitude encoding using Formula (3) revealed that when performed with double precision floating-point, the encoding differed from the correct value by 1 in 27,259 cases. While this is a relatively small number compared to the 2^{32} test cases, it shows how different expressions of the same quantity may lead to numerical errors in calculation.

The animation of the CPR specification also confirmed reported observations that a straightforward implementation of CPR in single precision floating-point arithmetic is unsound. The appendix T of [6] includes several tables containing the expected output of the CPR algorithm on a reduced set of AWB latitudes. Encoding these latitudes in a single precision implementation of Formula (3), resulted in 162 wrong encodings (with respect to the expected output in Appendix

T) over a total of 232 input AWB latitudes. In the case of local decoding, 46 encoded positions over a total of 116 were wrongly decoded by using single precision floating-point numbers. Finally, in the case of global decoding, the number of wrong cases detected was 28 out of 116.

5 Conclusion

This paper presents a formal analysis of the CPR algorithm used for encoding and decoding position messages for ADS-B broadcast. The formal analysis includes a formal specification in PVS and a proof of the correctness of the algorithm for a set of tightened requirements from those originally proposed. These tightened requirements are also shown to be necessary, by proving that there exist positions meeting the original requirements, but not decoding to a correct position.

The paper also presents a collection of simplifications of some the mathematical expressions used in the algorithm, which are proven to be mathematically equivalent to the original expressions, but also shown to be numerically simpler in the sense that the expressions evaluate in floating-point to values closer to the platonic computation. The evaluation of these simplifications was aided by an approach in the formal specification that allowed for the evaluation of arithmetic operators in a variety of computation models. This approach may be useful outside of the current work to examine the effect of numerical imprecision on the floating-point implementation of a platonic algorithm.

A possible further direction is the completion of the formal analysis for the two types of CPR messages, coarse and surface, that were not addressed in this work. This would not be theoretically difficult, as the existing specification and proofs would serve as a clear roadmap, but would take a significant amount of work. An area of current research is the formal numerical analysis of fixed-point and floating-point implementations of CPR. This analysis will enable the development of formally verified CPR implementations that could serve as reference implementations of the standard mathematical definition.

References

1. Judy Crow, Sam Owre, John Rushby, Natarajan Shankar, and David Stringer-Calvert. Evaluating, testing, and animating PVS specifications. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA, March 2001. Available at `http://www.csl.sri.com/users/rushby/abstracts/attachments`.
2. Aaron Dutle, César Muñoz, Anthony Narkawicz, and Ricky Butler. Software validation via model animation. In Jasmin Blanchette and Nikolai Kosmatov, editors, *Proceedings of the 9th International Conference on Tests & Proofs (TAP 2015)*, volume 9154 of *Lecture Notes in Computer Science*, pages 92–108, L'Aquila, Italy, July 2015. Springer.
3. ICAO. *Manual on the Universal Access Transceiver (UAT).*, volume 9861 of *Doc (International Civil Aviation Organization)*. 2012.

4. César Muñoz. Rapid prototyping in PVS. Contractor Report NASA/CR-2003-212418, NASA, Langley Research Center, Hampton VA 23681-2199, USA, May 2003.
5. Sam Owre, John Rushby, and Natarajan Shankar. PVS: A Prototype Verification System. In Deepak Kapur, editor, *Proceeding of the 11th International Conference on Automated Deductioncade*, volume 607 of *Lecture Notes in Artificial Intelligence*, pages 748–752. Springer, June 1992.
6. RTCA SC-186. RTCA-DO-260B, Minimum Operational Performance Standards for 1090 MHz extended squitter Automatic Dependent Surveillance - Broadcast (ADS-B) and Traffic Information Services - Broadcast (TIS-B), December 2009.