Compact Position Reporting Algorithm

A verified floating-point implementation in C

Mariano M. Moscato
mariano.moscato@nianet.org
National Institute of Aerospace

Joint effort: Cesar Muñoz (NASA), Laura Titolo (NIA), Aaron Dutle (NASA), François Bobot (CEA-list)

Sound Static Analysis for Security Workshop - June 27th, 2018
The Algorithm
The ADS-B System

- **Automatic Dependent Surveillance - Broadcast**
  - Supports NextGen
    - Next generation of air traffic management systems
  - Aircraft periodically *broadcasts* accurate surveillance information to ground stations and near aircraft
    - position and velocity
  - *Automatic* — no pilot intervention needed
  - *Dependent* — on navigation system

- Mandatory on Jan 1, 2020 (in USA and Europe)
  - More than 40000 aircraft currently equipped
The ADS-B Protocol

- **Pros: broadcast vs. radar-based approaches**
  - ✓ More precise
    - ○ NextGen requirement: position granularity of ~5.1 meters
  - ✓ More coverage
- **Cons: Make use of existent hardware**
  - ✗ TCAS transponders
  - ✗ 35 bits for position data in the broadcast message
  - ✗ Too coarse granularity (~300 meters)
    - ○ if raw positions are transmitted
Compact Position Reporting

Contiguous transmitted positions share prefixes

Idea: transmit only 17 less significative bits
Focus on Latitude First
Latitude Zones

- Divide the globe into 60 equally sized zones
- Divide each zone in $2^{17}$ bins

Zone Size: $\text{Dlat} = \frac{360}{60} = 6$ degrees
Broadcast only the corresponding *bin number* (YZ)
To encode lat, calculate:

1. Distance from southern edge of enclosing zone
   - \( \text{mod} (\text{lat}, \text{Dlat}) \)
2. Proportion w.r.t. the entire zone
   - \( \text{mod} (\text{lat}, \text{Dlat}) \cdot \frac{1}{\text{Dlat}} \)
3. Correspondent \textit{bin} number
   - \( \text{mod} (\text{lat}, \text{Dlat}) \cdot \frac{1}{\text{Dlat}} \cdot 2^{17} \)
4. Round to the nearest integer
   - \( ZY = \left\lfloor \text{mod} (\text{lat}, \text{Dlat}) \cdot \frac{1}{\text{Dlat}} \cdot 2^{17} + \frac{1}{2} \right\rfloor \)
How to Recover the Zone Index
Recovering Zone Index

Assuming Parallel-to-Equator Trajectory

---

Equator
Recovering Zone Index

Assuming Parallel-to-Equator Trajectory
Recovering Zone Index

Assuming Parallel-to-Equator Trajectory
Recovering Zone Index

Assuming Parallel-to-Equator Trajectory
Zone Index: \( ZI := \left[ ZO_0 - ZO_1 + \frac{1}{2} \right] \)
Relaxing parallel-to-the-Equator restriction

- According to the standard, if two latitudes $A$ and $B$ are less than half zone offset apart from each other,
  - $A$ and $B$ lie in the same zone, or
  - $A$ is one zone ahead w.r.t. $B$

- To deal with both cases

$$ZI = \begin{cases} 
\text{mod} \left( \left\lfloor ZO_0 - ZO_1 + 1/2 \right\rfloor , 60 \right) & \text{even zone index} \\
\text{mod} \left( \left\lfloor ZO_0 - ZO_1 + 1/2 \right\rfloor , 59 \right) & \text{odd zone index}
\end{cases}$$
Global Decoding

Given an even and an odd bin number \( YZ_0 \) and \( YZ_1 \), the recovered latitude \( Rlat_i \) is defined as

\[
Rlat_i(YZ_0, YZ_1) := Dlat_i \left( \mod \left( \left\lfloor ZO_0 - ZO_1 + 1/2 \right\rfloor, 60 - i \right) + YZ_i \frac{1}{2^{17}} \right)
\]

where \( ZO_i \) (zone offset) \( ZO_i := \frac{Dlat_i}{ZO} \cdot \frac{YZ_i}{2^{17}} \) where \( i \in \{0, 1\} \)

- Note that
  - \( Rlat_i \) returns the center of the bin where the input latitude lies.
  - Decoded latitude is at most at half-bin size from the input latitude
What About Longitudes?
Dealing with Longitudes

• Goal: same encoding resolution everywhere
  ▪ as close to a constant as possible all around the globe

• Same idea
  ▪ ~Equally sized zones divided in $2^{17}$ bins

• One distinctive feature
  ▪ Longitude (radial) size shrinks when approaching the poles
  ▪ Number of longitude zones is a function of latitude
    ◦ reducing the number of zones as latitude increases
• \( \text{NL(lat)} \): number of even longitude zones at latitude \( \text{lat} \)

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

• In practice, computing this function is inefficient
  ■ A lookup table of transition latitudes is pre-calculated
Global Decoding

- Latitude, given two encoded latitudes
  \[ R \text{lat}_i(YZ_0, YZ_1) := D \text{lat}_i \left( \mod \left( \left\lfloor \frac{59YZ_0-60YZ_1}{2^{17}} + \frac{1}{2} \right\rfloor, 60 \right) + \frac{YZ_4}{2^{17}} \right) \]

- Longitude, given two encoded positions
  \[ R \text{lon}_i(YZ_0, YZ_1, XZ_0, XZ_1) := D \text{lon}_i \left( \mod \left( \left\lfloor \frac{(nl-1)XZ_0-nl\cdot XZ_1}{2^{17}} + \frac{1}{2} \right\rfloor, nl'_i \right) + \frac{XZ_4}{2^{17}} \right) \]

where

- \( nl := \text{NL}(R \text{lat}_0(YZ_0, YZ_1)) \), must be equal to \( \text{NL}(R \text{lat}_1(YZ_0, YZ_1)) \)
- \( nl'_i := \max(nl - i, 1) \), since \( nl \) is 1 if \( |R \text{lat}_i(YZ_0, YZ_1)| > 87 \)
- \( D \text{lon}_i := 360/nl'_i \)
Local Decoding

- Additional decoding method
- Uses a *reference position* and one position message
  - instead of two position messages
- Positions apart for no more than half of a zone
  - According to the standard
  - Allows for bigger separation between received positions
- Idea: create a sliding region 1 zone wide
  - Centered on reference position
  - Each bin number occurs only once in the region
Known Issues

Reported by Airservices Australia (2007)
Accomplishments:

1. Found technical issues in the standard
   - Counterexamples for the real-valued model
2. Amended version proven correct
   - Prototype Verification System (PVS)
3. Proposed simpler formulation
   - reducing numerical complexity
4. Prototype implementation formally verified
   - C, PVS, Frama-C, Gappa, Alt-Ergo
Technical Issues

- Counterexamples found for both decoding settings
  - Even Assuming (exact) real-valued arithmetics
  - For example, in the *global decoding* case
    - $\text{lat}_0 = 363373617 \cdot 360/2^{32} \approx 30.4576247279$
    - $\text{lat}_1 = 363980245 \cdot 360/2^{32} \approx 30.5084716994$
    - decoded positions are further away for more than a *bin*

- Correctness proved on tightened requirements
  - max. distance of input positions decreased by half-bin size
Numerical Simplifications

• Mathematically equivalent expressions suggested
  - Numerically simpler
  - Equivalence formally proven

• Example: equivalent calculation of $NL$ lookup table
  - removing four operations in total
  - $\text{lat}_{NL}(nl) := \frac{180}{\pi} \arccos\left( \frac{\sin(\pi/60)}{\sin(\pi/nl)} \right)$.

• Example: cancellation instead of division
  - Reducing complexity of encoding algorithm
  - \[
    \frac{\text{mod}(a,b)}{b} = \frac{a-b*\left\lfloor \frac{a}{b} \right\rfloor}{b} = \frac{a}{b} - \left\lfloor \frac{a}{b} \right\rfloor
  \]
Example: Latitude Global Decoding

- According to the standard:

  \[ \text{Rlat}_0(YZ_0, YZ_1) := \text{Dlat}_0 \left( \mod \left( \left[ \frac{59YZ_0 - 60YZ_1}{2^{17}} + \frac{1}{2} \right], 60 \right) + \frac{YZ_0}{2^{17}} \right) \]

- Simplified version of global decoding (i=0) in ACSL

```plaintext
/*@ axiomatic real_function {
  logic real rLatr (int yz0,int yz1) =
    \let dLatr = 360.0 / 60.0;
    \let jar = (59.0*yz0 - 60.0*yz1 + 0x1.0p+16)*0x1.0p-17;
    \let jr = floor(jar);
    \let j60ir = jr/60.0;
    dLatr*((jr-60.0*(floor(j60ir)))+yz0*0x1.0p-17); } @*/
```
Example: Latitude Global Decoding

- Simplified version of global decoding (i=0) in ACSL

```plaintext
/*@ axiomatic real_function {
  logic real rLatr (int yz0, int yz1) =
  \let dLatr = 360.0 / 60.0;
  \let jar = (59.0*yz0 - 60.0*yz1 + 0x1.0p+16)*0x1.0p-17;
  \let jr = \floor(jar);
  \let j60ir = jr/60.0;
  dLatr*((jr-60.0*\floor(j60ir))+yz0*0x1.0p-17); } @*/
```

- Translated by hand into a PVS declaration

```plaintext
rLatr_i_0 (yz0,yz1:int): real =
  LET dLatr = 360 / 60 IN
  LET jar = (59*yz0 - 60*yz1 + 2^16) * 2^-17 IN
  LET jr = floor(jar) IN
  LET j60ir = jr/60 IN
  dLatr * ((jr - 60*floor(j60ir)) + yz0 * 2^-17)
```

- Proven to be equivalent to version from the standard
Example: Latitude Global Decoding

Frama-C/WP & Alt-Ergo+Gappa: the floating-point result is at most 0.000022888° (half bin size) apart from the logical result.
Floating-point version has the expected granularity: decoded and input positions are less than $\frac{1}{2}$ bin apart

- Amended CPR version has been proved correct, i.e.,
  - decoded latitude lies in the center of a bin and
  - it is less than half bin apart from the input
- It coincides with the ACSL logic definition
- C version is less than half bin size apart from it
• logic ACSL declarations translated to PVS by hand
• proved equivalent to existent CPR formalization
• C code verified using Frama-C/WP/Alt-Ergo/Gappa
Concluding Remarks

- Synergetic use of diverse analysis tools on
  - complex verification effort
  - relatively simple algorithm
    - no loops, no pointers, no arrays

- Proposed algorithm is being considered as reference implementation of CPR
  - RTCA DO-260B/Eurocae ED-102A
Future Work

- Extend results to other CPR modalities
  - Airborne, Surface, Coarse TIS-B
- Develop CPR integer-valued version
  - correctness (PVS) + verified implementation (Frama-C)
- Analysis of Floating-Point Programs
  - Frama-C: WP plugin to export VCs directly to PVS
  - Floating-point programs: Frama-C + PRECiSA
    - [http://precisa.nianet.org/](http://precisa.nianet.org/)
Compact Position Reporting Algorithm

A verified floating-point implementation in C

Mariano M. Moscato
mariano.moscato@nianet.org
National Institute of Aerospace

Thank you for your attention