This short article provides a quick look at recognizing possible arithmetic overflow for fixed block binary addition.
Although examples themselves are never proofs, they do make things clearer, so our prototypical addition is as follows:

$$\begin{array}{rr} & 1001\ 1101 \\ + & 1111\ 0010 \\ \hline & 1\ 1000\ 1111 \end{array}$$

The need for such an error recognition test is primarily for computing science applications. So in the above we're adding two 8-bit unsigned integers and we end up with a *carry* of 1: Thus an arithmetic overflow. Let's say an 8-bit block is the most our hardware (registers) can handle, in that case if using the builtin cpu addition algorithm you can't prevent possible overflows directly.

The recognition—and potential correction—of such overflow relies specifically on the mathematical property of integers:

Lemma 0.1

$$0 \le a, b < m \quad \Longrightarrow \quad a + b < 2m$$

First note if $a = 0$ or $b = 0$ then this trivially holds. Otherwise the proof relies on the fact that if $d < e$ and $f > 0$ then $f + d < f + e$. It's pretty basic undergrad math, but let's go over it anyway, starting with the simple derivation:

$$\begin{array}{rrcl} & a & < & m \\ & 0 & < & b \\ \Longrightarrow & a + b & < & b + m \end{array}$$

With the same logic, we also have:

$$\begin{array}{rrcl} & b & < & m \\ & 0 & < & m \\ \Longrightarrow & b + m & < & 2m \end{array}$$

Putting these together we get:

$$\begin{array}{rccccc} & a + b & < & b + m & < & 2m \\ \Longrightarrow & a + b & & < & & 2m \end{array}$$

The main use of this is the following upperbound:

$$0 \le a, b < m \quad \Longrightarrow \quad a + b - m < m$$

The error recognizer then, is as follows:

Lemma 0.2

$$0 \le a, b < m \le a + b \quad \Longrightarrow \quad a + b - m < a, b$$

First off, if $m \le a + b$, then $0 \le a + b - m$. Secondly from our initial lemma we know that $a + b - m < m$:

$$0 \le a + b - m < m$$

I've focused on the value $a + b - m$ for the simple reason that it is the value returned if arithmetic overflow does occur. This is also equivalent to modular arithmetic—only more efficient than using division.

The proof is a simple contradiction. Focusing on a, assume $a + b - m \ge a$, but then follows that $b \ge m$. A one-step contradiction. A symmetric argument follows for b.

Our arithmetic overflow recognizer follows because if $a + b$ does not overflow, then $a, b \le a + b < m$, but if it does overflow, the equivalent of modular arithmetic by m (here $= 2^8$) will occur as a side effect of any cpu addition algorithm, and from our second lemma we have $a + b < a, b < m$. It's a clear logical dichotomy.