On computing the semi-sum of two integers

Salvatore Ruggieri
Dipartimento di Informatica, Università di Pisa,
Via F. Buonarroti 2, 56125 Pisa ITALY
`ruggieri@di.unipi.it`

Abstract

We derive a sound program for computing the semi-sum of two integers using only integer operators and without incurring overflow.

Keywords: formal methods, program derivation.

1 Problem statement

Given two integers a and b, we wish to compute[1] $\lfloor (a+b)/2 \rfloor$, also called the *semi-sum* of a and b. While the problem may seem elementary, we must tackle some implementation issues that demand a non-trivial solution.

Consider the "first-answer" solution consisting of the simple C/Java expression `(a+b)/2`. First, recall that division between integer expressions in C, Java and other common languages, rounds towards zero. Specifically, for an integer `n`, the expression `n/2` evaluates to $\lfloor n/2 \rfloor$ when $n \geq 0$ and evaluates to $\lceil n/2 \rceil$ when $n \leq 0$. Therefore, `(a+b)/2` is a sound implementation of $\lfloor (a+b)/2 \rfloor$ only when $(a+b) \geq 0$.

Second, computation of sub-expression `(a+b)` may cause an overflow, i.e. its value may be out of the range of representable integers. Let us write $rep(n)$ iff n is a representable integer. For now, we do not consider any specific representation (later on we will assume a two's complement representation) but we only assume that $rep(a) \wedge rep(b)$, that $rep(0)$ and that:

$$\forall m, n, p \in Z. \quad (\ rep(m) \wedge rep(n) \wedge m \leq p \leq n \quad \Rightarrow \quad rep(p)\ ). \tag{1}$$

Since $min\{a, b\} \leq \lfloor (a+b)/2 \rfloor \leq max\{a, b\}$, by (1) the semi-sum of a and b is a representable integer.

We are now in a position to formally state the problem specification.

Problem statement. *Derive a C program* `SEMI-SUM` *such that for* `a`, `b`, `s` *variables of type* `int` *the following Hoare triple is valid:*

$$\{true\}\ \texttt{SEMI-SUM}\ \{s = \lfloor (a+b)/2 \rfloor\}$$

and such that all expressions in `SEMI-SUM` *denote representable integers only.*

[1] Throughout this paper we adopt standard mathematical notation (see e.g. [GKP89]) concerning the floor ($\lfloor\ \rfloor$) and ceiling ($\lceil\ \rceil$) operators.

Such a calculation occurs quite often in computer programs, e.g. in the well-known binary search algorithm. As another example, the semi-sum is computed in the C4.5 decision tree induction algorithm [Qui93], where the following (here, simplified) recursive procedure is adopted. Given an array of distinct integers, two elements a and b in the array are selected according to some criterion and then the array is split into two parts: those elements at most the semi-sum of a and b, and those elements greater than such a semi-sum. The procedure is recursively applied to each of the two arrays unless their length is less than two.

In testing an implementation of the C4.5 algorithm [Rug02], infinite loops originated from the "first-answer" calculation. First, the computed semi-sum of -3 and -2 was -2 (wrong, since it is -3). This led to splitting an array such as $[-3, -2]$ into $[-3, -2]$ and an empty one: recursion on the first split yielded the infinite loop. Second, the computed semi-sum of 2^{30} and 2^{30} was -2^{30} (wrong, since it is 2^{30}). This led to splitting an array such as $[2^{30}, 2^{30}]$ into itself and an empty one: as before, this led to an infinite loop.

2 Second-answer calculation

As discussed in the previous section, the "first-answer" calculation is a sound implementation only when $(a + b) \geq 0$ and $rep(a + b)$, i.e.:

$$\{(a + b) \geq 0 \ \wedge \ rep(a + b)\} \texttt{ s = (a+b)/2; } \{s = \lfloor (a + b)/2 \rfloor\}. \tag{2}$$

In (2), the expression `(a+b)` denotes a representable integer by assumption, and the expression `(a+b)/2` denotes a representable integer since it coincides with the semi-sum of a and b, which is representable.

Consider now the case $(a + b) \leq 0$. Using the identity ([GKP89, (3.17)]):

$$\forall m \in Z. \quad (\ m = \lfloor m/2 \rfloor + \lceil m/2 \rceil \ )$$

we derive: $\lfloor (a + b)/2 \rfloor = (a + b) - \lceil (a + b)/2 \rceil$. Since $(a + b) \leq 0$, $\lceil (a + b)/2 \rceil$ coincides now with `(a+b)/2`. Therefore:

$\{(a + b) \leq 0 \ \wedge \ rep(a + b)\}$
`int sum = a+b;`
$\{sum = (a + b) \ \wedge \ sum \leq 0 \ \wedge \ rep(sum)\}$
`s = sum - sum/2`
$\{s = \lfloor (a + b)/2 \rfloor\}$.

Also, note that $sum \leq \lceil sum/2 \rceil \leq 0 \ \wedge \ rep(sum)$ and $rep(0)$ imply by (1) $rep(\lceil sum/2 \rceil)$, i.e. `sum/2` denotes a representable integer. Also $rep(sum - \lceil sum/2 \rceil)$ holds since $sum - \lceil sum/2 \rceil$ is the semi-sum of a and b. Therefore, all expressions denote representable integers. Finally, merging (2) with the last program to get a "second-answer" program `SEMI-SA`:

```
int sum = a+b;
if( sum >= 0 )
   s = sum / 2;
else
   s = sum - sum/2;
```

for which the following Hoare triple is valid:

$$\{rep(a + b)\} \texttt{ SEMI-SA } \{s = \lfloor (a + b)/2 \rfloor\}. \tag{3}$$

3 Nonnegative-division calculation

There is a second identity ([GKP89, (3.6)]):

$$\forall x \in R\ \forall m \in Z.\quad (\ \lfloor m + x \rfloor = m + \lfloor x \rfloor\ )$$

that allows us to rewrite $\lfloor (a+b)/2 \rfloor = \lfloor a + (b-a)/2 \rfloor = a + \lfloor (b-a)/2 \rfloor$. When $a \le b$, the value $b - a$ is a non-negative integer, and $\lfloor (b-a)/2 \rfloor$ coincides with `(b-a)/2`. Therefore:

$$\{a \le b\ \wedge\ rep(b-a)\}\ \texttt{s = a + (b-a)/2;}\ \{s = \lfloor (a+b)/2 \rfloor\}.$$

As in the last section, it is readily checked that `(b-a)/2` and `a + (b-a)/2` denote representable integers. Similarly, when $b \le a$:

$$\{b \le a\ \wedge\ rep(a-b)\}\ \texttt{s = b + (a-b)/2;}\ \{s = \lfloor (a+b)/2 \rfloor\}.$$

We can then conclude that for the program `SEMI-NND`:

```
if(a <= b)
   s = a + (b-a)/2;
else
   s = b + (a-b)/2;
```

the following Hoare triple is valid[2]:

$$\{rep(max\{a,b\} - min\{a,b\})\}\ \texttt{SEMI-NND}\ \{s = \lfloor (a+b)/2 \rfloor\}. \tag{4}$$

4 Semi-sum calculation

Both `SEMI-SA` and `SEMI-NND` make a precondition on sub-expressions in order to prevent overflow. A way to satisfy those preconditions is to cast a and b up to a larger numeric data type (e.g., from 32-bit to 64-bit integers), and then to cast the result back to the original data type. However, it may be the case that a larger data type is not available. In this section, we derive a general solution.

Consider again the triple (3). It differs from the problem specification in making the additional assumption $rep(a+b)$. We observe:

$$\begin{array}{lll} rep(a+b) & \Leftarrow & \{\ (1) \text{ with } p = (a+b),\ m = a \text{ and } n = b\ \} \\ & & rep(a) \wedge rep(b) \wedge a \le (a+b) \wedge (a+b) \le b \\ & \equiv & \{\ rep(a), rep(b), \text{cancellation}\ \} \\ & & a \le 0 \le b \end{array}$$

Analogously, we derive $rep(a+b) \Leftarrow b \le 0 \le a$. By the consequence rule of Hoare logic, these two implications and (3) lead to:

$$\{min\{a,b\} \le 0 \le max\{a,b\}\}\ \texttt{SEMI-SA}\ \{s = \lfloor (a+b)/2 \rfloor\}. \tag{5}$$

Let us apply the same reasoning to the triple (4). For simplifying the notation, let $x = min\{a,b\}$ and $y = max\{a,b\}$. We have:

$$\begin{array}{lll} rep(y-x) & \Leftarrow & \{\ (1) \text{ with } p = (y-x),\ m = 0 \text{ and } n = y\ \} \\ & & rep(0) \wedge rep(y) \wedge 0 \le (y-x) \wedge (y-x) \le y \\ & \equiv & \{\ rep(0), rep(a), rep(b), x \le y, \text{cancellation}\ \} \\ & & 0 \le x \end{array}$$

[2] Note that by (1), $0 \le a\ \wedge\ 0 \le b \Rightarrow rep(max\{a,b\} - min\{a,b\})$. By the consequence rule of Hoare logic, (4) implies $\{0 \le a\ \wedge\ 0 \le b\}$ `SEMI-NND` $\{s = \lfloor (a+b)/2 \rfloor\}$, which states that `SEMI-NND` is sound for computing the semi-sum of two representable *natural numbers*.

Also, we can show that $rep(y-x)$ if $y<0$. In order to achieve this, we assume from now on the standard two's complement representation of integers using p bits plus sign: integers representable with the `int` data type range then from -2^p to 2^p-1. In addition to (1) and to $rep(0)$, two's complement notation implies:

$$\forall n \in Z. \quad (\ rep(n) \wedge n < 0 \quad \Rightarrow \quad rep(-n-1)\ ). \tag{6}$$

Let us show now that $rep(y-x)$ if $y<0$.

$$\begin{array}{rcl}
rep(y-x) & \Leftarrow & \quad \{\ (1) \text{ with } p=(y-x),\ m=0 \text{ and } n=(-x-1)\ \} \\
& & rep(0) \wedge rep(-x-1) \wedge 0 \leq (y-x) \wedge (y-x) \leq (-x-1) \\
& \equiv & \quad \{\ rep(0), x \leq y, \text{cancellation}\ \} \\
& & rep(-x-1) \wedge y < 0 \\
& \Leftarrow & \quad \{\ (6) \text{ with } n=x\ \} \\
& & rep(x) \wedge x < 0 \wedge y < 0 \\
& \equiv & \quad \{\ rep(a), rep(b), x \leq y\ \} \\
& & y < 0
\end{array}$$

By the consequence rule of Hoare logic, the last two implications and (4) lead to:

$$\{(0 \leq min\{a,b\} \ \vee \ max\{a,b\} < 0)\}\ \texttt{SEMI-NND}\ \{s = \lfloor (a+b)/2 \rfloor\}. \tag{7}$$

By observing that:

$$(0 \leq min\{a,b\} \ \vee \ max\{a,b\} < 0) \vee (min\{a,b\} \leq 0 \leq max\{a,b\})$$

we can design our final program `SEMI-SUM` by combining (5) and (7):

```
if( (0 <= a && 0 <= b) || (a < 0  && b < 0) ) {
        SEMI-NND
} else {
        SEMI-SA
}
```

For such a program the specification triple $\{true\}$ `SEMI-SUM` $\{s = \lfloor (a+b)/2 \rfloor\}$ is valid, and all expressions denote representable integers, i.e. no overflow occurs.

Acknowledgements

I am grateful to Prof. Roland Backhouse and to Prof. David Gries for their comments on a preliminary version of this paper.

References

[GKP89] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics: a Foundation for Computer Science.* Addison-Wesley Publishing Company, 1989.

[Qui93] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[Rug02] S. Ruggieri. Efficient C4.5. *IEEE Transactions on Knowledge and Data Engineering*, 14:438–444, 2002.