1-2003

# Are There Easy-To-Check Necessary and Sufficient Conditions for Straightforward Interval Computations to Be Exact?

Vladik Kreinovich
*The University of Texas at El Paso*, vladik@utep.edu

Luc Longpre
*The University of Texas at El Paso*, longpre@utep.edu

James J. Buckley



Comments:
UTEP-CS-02-11c.
Short version published in the *Extended Abstracts of 2002 SIAM Workshop on Validated Computing*, Toronto, Canada, May 23-25, 2002, pp. 94-96; full paper published in *Reliable Computing*, 2003, Vol. 9, No. 5, pp. 349-358.

# Are There Easy-to-Check Necessary and Sufficient Conditions for Straightforward Interval Computations To Be Exact?

Vladik Kreinovich[1], Luc Longpré[1], and James J. Buckley[2]

[1]Department of Computer Science, U. of Texas at El Paso
El Paso, TX 79968, USA, {vladik,longpre}@cs.utep.edu
[2]Mathematics Department, U. of Alabama at Birmingham
Birmingham, AL 35294-1170, USA, buckley@math.uab.edu

**Abstract**

We prove that no "efficient" (easy-to-check) necessary and sufficient conditions are possible for checking whether straightforward interval computations lead to the exact result.

**One of the main problems of interval computations.** One of the main problems of interval computations is to find a range of a given function on given intervals. To be more precise, the problem is: given $n$ input intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$ and an algorithm $f(x_1, \ldots, x_n)$ that transforms $n$ real numbers $x_1, \ldots, x_n$ into a real number $y = f(x_1, \ldots, x_n)$, find the range

$$\mathbf{y} = f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \{f(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}.$$

Usually, the endpoints of the intervals $\mathbf{x}_i$ come from measurements, and measurement usually produces rational numbers, so we can assume that the intervals $\mathbf{x}_i$ have rational endpoints. If we cannot compute the exact range, we can at least try to find an enclosure $\mathbf{Y} \supseteq \mathbf{y}$ for the range.

**Straightforward interval computations: its advantages and drawbacks.** Historically the first method for computing the enclosure for the range is the method which is sometimes called "straightforward" interval computations. This method is based on the fact that inside the computer, every algorithm consists of elementary operations (arithmetic operations, min, max, etc.). For each elementary operation $f(x, y)$, if we know the intervals $\mathbf{x}$ and $\mathbf{y}$ for $x$ and $y$, we can compute the exact range $f(\mathbf{x}, \mathbf{y})$. The corresponding formulas form the so-called *interval arithmetic*. In straightforward interval computations, we repeat the computations forming the program $f$ step-by-step, replacing each operation

with real numbers by the corresponding operation of interval arithmetic. It is known that, as a result, we get an enclosure for the desired range.

In some important cases, the enclosure obtained by using straightforward interval computations is actually the exact range. There are several sufficient conditions for straightforward interval computations to be exact: e.g., it is exact when $f(x_1, \ldots, x_n)$ is an explicit expression in which each variable occurs only once; another condition is given in [4].

However, there are known cases when the resulting enclosure is much wider than the actual range. For example, for the expression $f(x_1, x_2) = x_1 + x_1 \cdot x_2$, straightforward interval computations are exact when $\underline{x}_2 \geq 0$ and not exact when, e.g., $\mathbf{x}_1 = [\underline{x}_1, \overline{x}_1]$ is a non-degenerate interval and $\mathbf{x}_2 = [-1, -1]$. Indeed, in the second case, $f(x_1, x_2) = 0$, so we have a 1-point range $[0, 0]$, but straightforward interval computations result in $[\underline{x}_1 - \overline{x}_1, \overline{x}_1 - \underline{x}_1]$.

**More sophisticated methods and the first methodological question.** Several methods have been proposed to reduce the excess width: centered form, bisection, monotonicity check, etc. Some methods – like Hansen's generalized interval arithmetic [3] – decrease the excess width by taking into account dependence between interval variables; in these methods, the range of $x_1 + x_1 \cdot (-1)$ is correctly computed as $[0, 0]$.

Each new method improves the enclosures, often reducing the enclosure to the exact range, but for each known method, there are cases when this method still leads to excess width.

In such situations, when many methods have been proposed and none of them is perfect, a natural question is: *Is a perfect method* – that would always return the exact range in reasonable time – *possible at all?* This methodological question is important for algorithm designers:

- If a perfect method is possible, then it is reasonable to spend some time looking for it.
- On the other hand, if such a method is not possible at all, then looking for a perfect method would be a waste of time – like looking for a solution-in-radicals of a general fifth order algebraic equation or for a ruler-and-compass angle trisection.

If no general perfect method is possible, then, instead of wasting time looking for such a method, we should look either for *classes* of functions and/or domains for which it is possible to compute the exact range, or for algorithms that still lead to excess width, but produce *better* interval estimates than the existing ones.

**A (known) answer to the first methodological question.** For interval computations, this important methodological question was answered in 1981, when Gaganov proved [1, 2] that the problem of computing the range is NP-hard (see, e.g., [5] and references therein).

Crudely speaking, NP-hard means that there are no general ways for solving this problem (i.e., computing the exact range) in reasonable time. (As an aside, it is possible to compute the range exactly in time that increases exponentially with $n$ [5].) Of course, every NP-hard problem has easier-to-solve subclasses, and the problem of range estimation is no exception: as we have mentioned, there are several important classes of functions for which we can compute the exact range in reasonable time. However, the NP-hardness result means that when we design a general range estimation algorithm, we can, in general, only compute *enclosures* for the desired range.

Maybe the difficulty comes from the requirement that the range be computed exactly? In practice, it is often sufficient to compute, in a reasonable amount of time, usefully accurate bounds for $\mathbf{y}$, i.e., bounds which are accurate within a given accuracy $\varepsilon > 0$. Alas, for any $\varepsilon$, such computations are also NP-hard.

This NP-hardness is not so bad from a practical viewpoint as it may sound. For example, in [5], we analyzed "in what sense" the computation is NP-hard: with respect to the dimension $n$ or to the number of operations?

- We showed that the problem remains NP-hard if we only consider quadratic functions of the arbitrary number of variables – so it is, in this sense, "NP-hard with respect to the dimension $n$".
- However, if we fix the dimension, then it is already possible to have a polynomial-time (feasible) algorithm for exact range estimation – i.e., the problem is *not* "NP-hard with respect to the number of operations".

Since in many practical problems, dimensionality $n$ is reasonably small, we have a lot of practical problems for which a feasible algorithm is possible.

**Second methodological question.** When we use an algorithm – e.g., straightforward interval computations – to estimate the range, we know that the result *may* contain excess width. But does it? Can we efficiently check whether straightforward interval computations are exact?

As we have mentioned, there are many important *sufficient* conditions under which straightforward interval computations produce an exact range. New better sufficient conditions are being discovered. However, none of the known conditions is *necessary*, because for each of these conditions, there are cases not covered by this condition in which the results are nevertheless exact.

Again, we have a natural question: are perfect (i.e., necessary, sufficient, and easy to check) conditions possible at all? If they are possible, then it is reasonable to spend some time looking for them. If such conditions are not possible, then looking for such perfect conditions would be a useless waste of time.

**Our answer to this question.** Let us consider algorithms $f(x_1, \ldots, x_n)$ that consist only of the operations $+$, $-$, $\cdot$, min, and max.

**Theorem.** *The problem of checking whether for a given algorithm $f(x_1, \ldots, x_n)$ and given intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$, straightforward interval computations are exact, is NP-hard.*

**Proof.** In the proof of Theorem 3.1 from [5], we have shown that a known NP-hard problem – checking satisfiability of a 3-CNF propositional formula $F$ – can be reduced to checking whether for an appropriate quadratic polynomial $f$, the lower endpoint $\underline{y}$ of the range $\mathbf{y}$ is $\underline{y} = 0$ (if $F$ is satisfiable) or $\underline{y} \geq 0.09$ (if $F$ is not satisfiable). Let us describe this reduction in detail and show how it can be modified to prove our result.

1°. Propositional satisfiability problem for 3-CNF formulas (also known as *3-SAT*) was historically the NP-complete problem proved to be NP-complete.

This problem consists of the following: Suppose that an integer $v$ is fixed; we have $v$ propositional variables $z_1, \ldots, z_v$ (i.e., variables that can only take values "true" or "false"), and a formula $F$ of the type $F_1 \& F_2 \& \ldots \& F_k$ is given, where each of the expressions $F_j$ has the form $a \vee b$ or $a \vee b \vee c$, and $a, b, c$ are either the variables $z_1, \ldots, z_v$, or their negations $\neg z_1, \ldots, \neg z_v$ (these $a, b, c, \ldots$ are called *literals*).

For *example*, we can take a formula $(z_1 \vee \neg z_2) \& (\neg z_2 \vee z_4 \vee \neg z_5)$.

If we assign arbitrary Boolean values ("true" or "false") to $v$ variables $z_1, \ldots, z_v$, then, applying the standard logical rules, we get the truth value of $F$. A formula $F$ is called *satisfiable* if there exist truth values $z_1, \ldots, z_v$ for which the truth value of the expression $F$ is "true". (In the computer, usually, "true" is represented as 1, and "false" as 0.) The 3-SAT problem is: given $F$, check whether it is satisfiable.

2°. Let us show how this 3-SAT problem can be reduced to the problem of computing the range of a quadratic polynomial. Specifically, for each 3-CNF formula $F = F_1 \& \ldots \& F_k$ with the Boolean variables $z_1, \ldots, z_v$, we will build a quadratic function $f^F(x_1, \ldots, x_n)$ with $n \stackrel{\text{def}}{=} v + k$ (real-valued) variables $x_1, \ldots, x_n$; for clarity, we will use notations $p_1$ for $x_{v+1}$, $p_2$ for $x_{v+2}$, $\ldots$, and $p_k$ for $x_{v+k}$. The construction of $f^F$ is as follows:

- To each propositional variable $z_i$, we put into correspondence a real-number variable $f^{z_i} = x_i$.
- To each negative literal $\neg z_i$, we put into correspondence a linear expression $f^{\neg z_i} = 1 - x_i$.
- To each expression $F_j$ of the type $a \vee b$, we put into correspondence the expression $f^{F_j} = (f^a + f^b + p_j - 2)^2$. Since $f^a$ and $f^b$ are linear in the variables $x_i$, the resulting expression is quadratic in $x_i$ and $p_j$.
- To each expression $F_j$ of the type $a \vee b \vee c$, we put into correspondence the expression $f^{F_j} = (f^a + f^b + f^c + 2p_j - 3)^2$. The resulting expression is quadratic in $x_i$ and $p_j$.

- To the formula $F$, we put into correspondence the quadratic function

$$f(x_1,\ldots,x_v,p_1,\ldots,p_k)=\sum_{i=1}^{v} x_i\cdot(1-x_i)+\sum_{j=1}^{k} f^{F_j}.$$

*Example.* Let us take $F=(z_1\vee z_2\vee z_3)\&(z_1\vee\neg z_2)$. For this formula, $v=3$, $k=2$, so, we need $v+k=5$ real-number variables $x_1$, $x_2$, $x_3$, $p_1$, and $p_2$. Here:

- $f^{\neg z_2}=1-x_2$.
- $f^{F_1}=(x_1+x_2+x_3+2p_1-3)^2$.
- $f^{F_2}=(x_1+(1-x_2)+p_2-2)^2$.
- $f^F(x_1,x_2,p_1,p_2)=x_1\cdot(1-x_1)+x_2\cdot(1-x_2)+x_3\cdot(1-x_3)+f^{F_1}+f^{F_2}$.

We choose $\mathbf{x}_i=\mathbf{p}_j=[0,1]$, and estimate the lower and the upper endpoints $\underline{y}$ and $\overline{y}$ of the range of the function $f^F$ on the corresponding box:

$$\mathbf{y}=[\underline{y},\overline{y}]\stackrel{\rm def}{=}f^F(\mathbf{x}_1,\ldots,\mathbf{x}_v,\mathbf{p}_1,\ldots,\mathbf{p}_k).$$

3°. Before we start estimating $\underline{y}$ and $\overline{y}$, let us notice that $f^{F_j}$ is defined as a square, and therefore, $f^{F_j}\geq 0$. Also, if $x_i\in[0,1]$, then $x_i(1-x_i)\geq 0$. Therefore, the function $f^F$ is a sum of non-negative numbers and is, thus, non-negative. Hence, $\underline{y}\geq 0$.

4°. Let us show that if the formula $F$ is satisfiable, then $\underline{y}=0$.

Indeed, if the formula $F$ is satisfiable, i.e., it is true for some propositional vector $z_1,\ldots,z_v$, then we take $x_i=z_i$ (i.e., $x_i=1$ if $z_i=$"true" and $x_i=0$ if $z_i=$"false"). The values of $p_j$ are chosen as follows:

- If $F_j=a\vee b$, and both $a$ and $b$ are true for $z_i$, then we take $p_j=0$.
- If $F_j=a\vee b$, and only one of the literals $a$ and $b$ is true for a given choice of $z_i$, then we take $p_j=1$.
- If $F_j=a\vee b\vee c$, and all three literals are true, then $p_j=0$.
- If $F_j=a\vee b\vee c$, and two out of three literals are true, then $p_j=0.5$.
- If $F_j=a\vee b\vee c$, and only one of the three literals is true, then $p_j=1$.

In all five cases, $f^{F_j} = 0$ for all $j$. Therefore, for these $x_i$ and $p_j$, $f^F(x_1, \ldots, x_n, p_1, \ldots, p_k) = 0$; hence, $\underline{y} = \min f^F \leq 0$. Since we know that $\underline{y} \geq 0$, we conclude that $\underline{y} = 0$.

5°. Let us show that if the formula $F$ is not satisfiable, then $\underline{y} \geq 0.09$.

We will prove this statement by reduction to a contradiction: we will assume that $\underline{y} < 0.09$, and conclude that $F$ is satisfiable. The minimum of a continuous function of a compact $[0,1]^{v+2k}$ is always attained; therefore, there exist values $x_i$ and $p_j$, for which $f^F(x_1, \ldots, x_v, p_1, \ldots, p_k) = \underline{y} < 0.09$. Since $f^F$ is defined as the sum of non-negative terms $x_i \cdot (1 - x_i)$ and $f^{F_j}$, from this inequality, it follows that each of these terms is $< 0.09$.

In particular, it follows that $x_i \cdot (1 - x_i) < 0.09$. The function $x \cdot (1 - x)$ is increasing for $x < 0.5$ and decreasing afterwards. So, from $x_i \cdot (1 - x_i) < 0.09$ and from the fact that $0.1 \cdot (1 - 0.1) = 0.9 \cdot (1 - 0.9) = 0.09$, it follows that $x_i < 0.1$ or $x_i > 0.9$ for all $i$. Let us take $z_i =$"true" if $x_i > 0.9$, and $z_i =$"false" if $x_i < 0.1$, and let us show that these propositional values make the formula $F$ true (i.e., they make all the expressions $F_j$ true). Indeed:

- If $F_j = a \vee b$, then from $f^{F_j} = (f^a + f^b + p_j - 2)^2 < 0.09$, it follows that $f^a + f^b + p_j - 2 > -0.3$, and $f^a + f^b > 1.7 - p_j$. Since $p_j \leq 1$, we conclude that $f^a + f^b > 0.7$. Therefore, the values $f^a$ and $f^b$ cannot be both $< 0.1$. Therefore, one of these two values is $> 0.9$. The corresponding literal is equal to "true", and hence, $F_j$ is true.
- If $F_j = a \vee b \vee c$, then from $f^{F_j} = (f^a + f^b + f^c + 2p_j - 3)^2 < 0.09$, it follows that $f^a + f^b + f^c + 2p_j - 3 > -0.3$, and $f^a + f^b + f^c > 2.7 - 2p_j$. Since $p_j \leq 1$, we conclude that $f^a + f^b + f^c > 0.7$. Therefore, the values $f^a$, $f^b$, and $f^c$ cannot be all $< 0.1$. Therefore, one of these three values is $> 0.9$. The corresponding literal is equal to "true", and hence, $F_j$ is true.

So, $F$ is satisfiable. The contradiction with our assumption that $F$ is not satisfiable proves that, under this assumption, the inequality $\underline{y} < 0.09$ is not possible and thus, under this assumption, we have $\underline{y} \geq 0.09$.

6°. Let us now show that the upper endpoint $\overline{y}$ of the range $\mathbf{y}$ of the function $f^F$ is always $\geq 0.25$.

Indeed, for $x_1 = \ldots = x_v = p_1 = \ldots = p_k = 0.5$, we have $x_1 \cdot (1 - x_1) = 0.25$. Since the function $f^F$ is the sum of non-negative terms including $x_1 \cdot (1 - x_1)$, we can conclude that $f^F(x_1, \ldots, x_v, p_1, \ldots, p_k) \geq 0.25$ and therefore, that $\overline{y} \geq 0.25$.

7°. Let us now consider, for every 3-CNF formula $F$, a new function

$$f_0^F(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \max(0.04, \min(f^F(x_1, \ldots, x_n), 0.25)).$$

This is the function whose range $\mathbf{y}_0 = [\underline{y}_0, \overline{y}_0]$ on the box $[0,1] \times \ldots \times [0,1]$ we will be estimating. Let us show that:

- if the original formula $F$ is satisfiable, then the actual range $\mathbf{y}_0$ of the function $f_0^F$ is equal to $[0.04, 0.25]$; and
- if the original formula $F$ is not satisfiable, then the actual range $\mathbf{y}_0$ of the function $f_0^F$ is equal to $[z, 0.25]$ for some value $z \geq 0.09$.

7.1°. Let us first show that for every formula $F$, we have $\overline{y}_0 \leq 0.25$.

Indeed, by definition of min, we have $\min(f^F(x_1, \ldots, x_n), 0.25) \leq 0.25$. Since the function max is non-decreasing in both arguments, we can conclude that

$$f_0^F(x_1, \ldots, x_n) = \max(0.04, \min(f^F(x_1, \ldots, x_n), 0.25)) \leq$$

$$\max(0.04, 0.25) = 0.25.$$

Since all the values of the function $f_0^F$ are $\leq 0.25$, its maximum also cannot exceed 0.25, i.e., $\overline{y}_0 \leq 0.25$.

7.2°. Let us now show that for every formula $F$, we have $\overline{y}_0 = 0.25$.

In view of Part 7.1 of this proof, it is sufficient to prove that there exist values $x_1, \ldots, x_n$ for which $f_0^F(x_1, \ldots, x_n) = 0.25$. Indeed, as we have shown in Part 6 of the proof, for $x_1 = \ldots = x_n = 0.5$, we have $f^F(x_1, \ldots, x_n) \geq 0.25$. For these values, $\min(f^F(x_1, \ldots, x_n), 0.25) = 0.25$, and therefore,

$$f_0^F(x_1, \ldots, x_n) = \max(0.04, \min(f^F(x_1, \ldots, x_n), 0.25)) =$$

$$\max(0.04, 0.25) = 0.25.$$

7.3°. Let us show that for every formula $F$, we have $\underline{y}_0 \geq \min(\underline{y}, 0.25)$.

Indeed, by definition of $\underline{y}$, for all $x_i$, we have $f^F(x_1, \ldots, x_n) \geq \underline{y}$. Since the function min is non-decreasing in each variable, we conclude that

$$\min(f^F(x_1, \ldots, x_n), 0.25) \geq \min(\underline{y}, 0.25).$$

Since $\max(a, b) \geq b$, we can conclude that

$$f_0^F(x_1, \ldots, x_n) = \max(0.04, \min(f^F(x_1, \ldots, x_n), 0.25)) \geq \min(\underline{y}, 0.25).$$

7.4°. As a corollary of Part 7.3, we conclude that for non-satisfiable formulas, for which $\underline{y} \geq 0.09$, we get $z \stackrel{\text{def}}{=} \underline{y}_0 \geq \min(\underline{y}, 0.25) \geq \min(0.09, 0.25) = 0.09$.

7.5°. To complete the proof of Part 7, we must show that for satisfiable formulas $F$, we have $\underline{y}_0 = 0.04$.

Let us first prove that $\underline{y}_0 \geq 0.04$. Indeed, by definition of the function $f_0^F$, we have $f_0^F(x_1, \ldots, x_n) = \max(0.04, \ldots) \geq 0.04$. Since all the values of the function $f_0^F$ are $\geq 0.04$, its minimum also cannot be smaller than 0.04, i.e., $\underline{y}_0 \geq 0.04$.

To complete the proof, we must show that the function $f_0^F$ actually attains the value 0.04. Indeed, in Part 4 of this proof, we have shown that if a formula $F$ is satisfiable, then there exist values $x_1, \ldots, x_n$ for which $f^F(x_1, \ldots, x_n) = 0$. For these values,

$$f_0^F(x_1, \ldots, x_n) = \max(0.04, \min(f^F(x_1, \ldots, x_n), 0.25)) =$$

$$\max(0.04, \min(0, 0.25)) = \max(0.04, 0) = 0.04.$$

The statement is proven.

8°. Let us now analyze the result of applying straightforward interval computations to the function $f_0^F$. We will denote this result by $\mathbf{Y}_0 = [\underline{Y}_0, \overline{Y}_0]$.

The interval $\mathbf{Y}_0$ is an enclosure for the actual range $\mathbf{y}_0$, i.e., $\mathbf{y}_0 \subseteq \mathbf{Y}_0$. This means that $\underline{Y}_0 \leq \underline{y}_0$ and $\overline{y}_0 \leq \overline{Y}_0$.

We have shown, in Part 7 of this proof, that $\overline{y}_0 = 0.25$, therefore, we have $0.25 \leq \overline{Y}_0$.

9°. Let us show that for every formula $F$, for the corresponding function $f_0^F$, we have $\underline{Y}_0 \geq 0.04$ and $\overline{Y}_0 = 0.25$.

9.1°. Let us first show that $\underline{Y}_0 \geq 0.04$.

Indeed, we have defined $f_0^F$ as $\max(0.04, g(x_1, \ldots, x_n))$, where

$$g(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \min(0.25, f^F(x_1, \ldots, x_n)).$$

If we represent the algorithm for computing $f_0^F$ as a sequence of elementary operations, then the last operation will be min applied to 0.04 and $g(x_1, \ldots, x_n)$. In straightforward interval computations, we replace each elementary operation with real number by the corresponding operation of interval arithmetic. Therefore, the range $\mathbf{Y}_0$ is obtained as follows:

- first, we make this replacement for all the elementary operations involved in computing the function $g(x_1, \ldots, x_n)$; as a result, we get an enclosure $\mathbf{G} = [\underline{G}, \overline{G}]$ for the range $\mathbf{g}$ of the function $g(x_1, \ldots, x_n)$;
- then, we apply the interval analogue of max to the degenerate interval $[0.04, 0.04]$ and the interval $\mathbf{G}$: $\mathbf{Y}_0 = \max([0.04, 0.04], \mathbf{G})$.

The interval analogue of max is well-known:

$$\max([\underline{a}, \overline{a}], [\underline{b}, \overline{b}]) = [\max(\underline{a}, \underline{b}), \max(\overline{a}, \overline{b})].$$

Thus, we have $\underline{Y}_0 = \max(0.04, \underline{G})$ hence $\underline{Y}_0 \geq 0.04$.

9.2°. Let us now show that $\overline{Y}_0 = 0.25$.

Indeed, we have defined $f_0^F$ as $\max(0.04, \min(0.25, f^F(x_1, \ldots, x_n)))$. If we represent the algorithm for computing $f_0^F$ as a sequence of elementary operations, then the last two operations will be:

- min applied to 0.25 and $f^F(x_1, \ldots, x_n)$; and
- max applied to 0.04 and $g(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \min(0.25, f^F(x_1, \ldots, x_n))$.

In straightforward interval computations, we replace each elementary operation with real numbers by the corresponding operation of interval arithmetic. Therefore, the range $\mathbf{Y}_0$ is obtained as follows:

- first, we make this replacement for all the elementary operations involved in computing the function $f^F(x_1, \ldots, x_n)$; as a result, we get an enclosure $\mathbf{Y} = [\underline{Y}, \overline{Y}]$ for the range $\mathbf{y}$ of the function $f^F(x_1, \ldots, x_n)$;
- then, we apply the interval analogue of min to the degenerate interval $[0.25, 0.25]$ and the interval $\mathbf{Y}$: $\mathbf{G} = \min([0.25, 0.25], \mathbf{Y})$.
- finally, we apply the interval analogue of max to the degenerate interval $[0.04, 0.04]$ and the interval $\mathbf{G}$: $\mathbf{Y}_0 = \max([0.04, 0.04], \mathbf{G})$.

We have already used the interval analogue of max; the interval analogue of min is similar:

$$\min([\underline{a}, \overline{a}], [\underline{b}, \overline{b}]) = [\min(\underline{a}, \underline{b}), \min(\overline{a}, \overline{b})].$$

Thus, we have $\overline{G} = \min(0.25, \overline{Y})$ – hence $\overline{G} \leq 0.25$ – and then $\overline{Y}_0 = \max(0.04, \overline{G})$.

Since $0.04 \leq 0.25$ and $\overline{G} \leq 0.25$, we conclude that $\overline{Y}_0 = \max(0.04, \overline{G}) \leq 0.25$.

We have shown, in Part 8 of this proof, that $\overline{Y}_0 \geq 0.25$, so we can conclude that $\overline{Y}_0 = 0.25$.

10°. Let us now summarize what we have proven. We have shown that the actual range $\mathbf{y}_0 = [\underline{y}_0, \overline{y}_0]$ of the function $f_0^F$ has the following properties:

- if the formula $F$ is satisfiable, then $\mathbf{y}_0 = [0.04, 0.25]$;
- if the formula $F$ is not satisfiable, then $\mathbf{y}_0 = [z, 0.25]$ for some $z \geq 0.09$.

We have also shown that the result $\mathbf{Y}_0 = [\underline{Y}_0, \overline{Y}_0]$ of applying straightforward interval computations to the function $f_0^F$ has the following properties:

- $\overline{Y}_0 = 0.25$;
- $0.04 \leq \underline{Y}_0 \leq \underline{y}_0$.

From these properties, we can conclude that for every formula $F$, for the corresponding function $f_0^F$, both the upper endpoint $\overline{y}_0$ for the actual range and the upper endpoint $\overline{Y}_0$ for the enclosure are equal to 0.25 – and thus, equal to each other.

Thus, if we could check whether straightforward interval computations are exact, i.e., whether $\mathbf{Y}_0 = \mathbf{y}_0$, we could thus check whether $F$ is satisfiable:

- If $\underline{Y}_0 > 0.04$, this means (due to $\underline{y}_0 \geq \underline{Y}_0$) that $\underline{y}_0 > 0.04$, hence the formula $F$ is not satisfiable.
- If $\underline{Y}_0 = 0.04$ and the result of straightforward interval computations is exact, then $\underline{y}_0 = 0.04$ hence $F$ is satisfiable.
- If $\underline{Y}_0 = 0.04$ and the result of straightforward interval computations is not exact, then, since the upper endpoints are equal, the only possibility of not exactness is $\underline{y}_0 \neq \underline{Y}_0$. Since we always have $\underline{y}_0 \geq \underline{Y}_0$, this inequality means that $\underline{y}_0 > \underline{Y}_0 = 0.04$, and so the propositional formula $F$ is not satisfiable.

This reduction to a known NP-hard problem proves that our problem is NP-hard as well. The theorem is proven.

*Comment.* A similar result holds if we allow division as well.

**Practical conclusions.** In other words, no feasible necessary and sufficient conditions are possible for checking whether the estimate obtained by using straightforward computations is exact. As a result, instead of trying to find such conditions, we should fully concentrate on identifying classes of functions (or functions and box values) for which straightforward computations lead to the exact range. For example, it is known that for the Gauss elimination algorithm, under certain conditions, we get the exact intervals for the solution; it is also known that completing the square of any quadratic function of one variable can be used to compute its exact range. Finding more cases like that is worth the effort.

*Mathematical comment.* This result easily implies that the exact computation of the range is NP-hard, but we know of no easy way to deduce our new result from the NP-hardness of interval computations. In this sense, our new result is stronger than the known result that computing the range is NP-hard.

**Related open problems.** As we have mentioned, in practice, it is usually sufficient to compute the range within a given accuracy $\varepsilon$. How difficult is it to check whether for a given algorithm $f(x_1, \ldots, x_n)$ and given intervals $\mathbf{x}_1, \ldots, \mathbf{x}_n$, straightforward interval computations are accurate within the given accuracy? We think that this problem is NP-hard, but we could not prove it.

What if we consider other methods – such as centered form? Again, we could not prove it either way.

**Acknowledgments.** This work was supported in part by NASA grants NCC5-209 and NCC 2-1232, by NSF grants CDA-9522207, EAR-0112968, EAR-0225670, and 9710940 Mexico/Conacyt, by AFOSR grant F49620-00-1-0365, and by IEEE/ACM SC2001 and SC2002 Minority Serving Institutions Participation Grants.

The authors are thankful to Eldon Hansen, Weldon A. Lodwick, Bill Walster, and to the anonymous referees for fruitful discussions.

## References

[1] A. A. Gaganov, *Computational complexity of the range of the polynomial in several variables*, Leningrad University, Math. Department, M.S. Thesis, 1981 (in Russian).

[2] A. A. Gaganov, "Computational complexity of the range of the polynomial in several variables", *Cybernetics*, 1985, pp. 418–421.

[3] E. R. Hansen, "A generalized interval arithmetic", In: K. Nickel (ed.), *Interval mathematics*, Springer Lecture Notes in Computer Science, 1975, Vol. 29, pp. 7–18.

[4] E. Hansen, "Sharpness in interval computations", *Reliable Computing*, 1997, Vol. 3, pp. 7–29.

[5] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1997.