

# From Computing Sets of Optima, Pareto Sets, and Sets of Nash Equilibria to General Decision-Related Set Computations

**Vladik Kreinovich**

(University of Texas at El Paso, USA  
vladik@utep.edu)

**Bartłomiej Jacek Kubica**

(Warsaw University of Technology, Poland  
B.Kubica@elka.pw.edu.pl)

**Abstract:** Several algorithms have been proposed to compute sets of optima, Pareto sets, sets of Nash equilibria. In this paper, we present a general algorithms for decision-related set computations that includes all these algorithms as particular cases.

**Key Words:** computing sets, sets of optima, Pareto sets, Nash equilibria

**Category:** F.2.1, F.4, G.1.m

## 1 Introduction

**Optimization is important.** In many practical decision making problems, we are interested in finding the alternative which is the best (under given constraints). In many cases, an objective function  $f(x)$  is explicitly given. In these cases, “the best” means that we want to find a solution which maximizes the value of this objective function, i.e., a solution  $x^*$  for which the value  $f(x^*)$  cannot be improved – i.e., for every element  $x$  of the set  $X$  of all possible solutions, we have  $f(x^*) \geq f(x)$ .

In formal terms, the condition that  $x^*$  is a location of a global optimum can be described as

$$\forall x (f(x^*) \geq f(x)). \quad (1)$$

**Computing sets of optima is important.** Often, there are several optima. In this case, it is desirable to provide the user with the set of all these optima, so that the user can see all the options when selecting an alternative.

From this viewpoint, it is desirable to “compute” (in some reasonable sense) the set of all the optima – i.e., the set of all the points  $x^*$  that satisfy the condition (1).

**Computing an exact global optimum of a computable function is, in general, not algorithmically possible.** It is known that even for functions

defined on the interval  $[0, 1]$ , in general, it not possible to have an algorithm that, given a computable function  $f : [0, 1] \rightarrow \mathbb{R}$ , returns a location  $x^*$  of its local maxima; see, e.g., [Kreinovich et al. 1998] or [Nachbar and Zame 1996] (definitions of a computable function will be reminded later in this paper).

For example, in [Kreinovich et al. 1998], it has been proven that no algorithm is possible that, given a computable polynomial of one variable which attains its optimum at exactly two points, will return these two optimizing points.

**In practice, it is sufficient to be able to compute an approximation to the set of optima.** From the practical viewpoint, the above algorithmic impossibility result is not that negative, since in practice, small differences in the values of the objective function can be safely ignored. Also, the objective function  $f(x)$  describing the consequences of selecting an alternative  $x$  is also only known approximately.

Let us denote, by  $\varepsilon$ , the accuracy below which differences in the values of  $f(x)$  can be safely ignored. Thus,

- the given value  $f(x^*)$  means that the actual (unknown) value  $f_{\text{act}}(x^*)$  of the objective function can be any number from the interval  $[f(x^*) - \varepsilon, f(x^*) + \varepsilon]$ ;
- the given value  $f(x)$  means that the actual (unknown) value  $f_{\text{act}}(x)$  of the objective function can be any number from the interval  $[f(x) - \varepsilon, f(x) + \varepsilon]$ ;
- and
- the optimization requirement  $f_{\text{act}}(x^*) \geq f_{\text{act}}(x)$  means, in these terms, that there exist values  $f_{\text{act}}(x^*) \in [f(x^*) - \varepsilon, f(x^*) + \varepsilon]$  and

$$f_{\text{act}}(x) \in [f(x) - \varepsilon, f(x) + \varepsilon]$$

for which  $f_{\text{act}}(x^*) \geq f_{\text{act}}(x)$ .

If one of the elements of the interval  $[f(x^*) - \varepsilon, f(x^*) + \varepsilon]$  is larger than or equal to one of the elements of the interval  $[f(x) - \varepsilon, f(x) + \varepsilon]$ , then the largest element  $f(x^*) + \varepsilon$  of the interval  $[f(x^*) - \varepsilon, f(x^*) + \varepsilon]$  is larger than or equal to the smallest element  $f(x) - \varepsilon$  of the interval  $[f(x) - \varepsilon, f(x) + \varepsilon]$ , i.e.,  $f(x^*) + \varepsilon \geq f(x) - \varepsilon$ .

Vice versa, if  $f(x^*) + \varepsilon \geq f(x) - \varepsilon$ , then an element  $f(x^*) + \varepsilon$  of the interval  $[f(x^*) - \varepsilon, f(x^*) + \varepsilon]$  is larger than or equal to the element  $f(x) - \varepsilon$  of the interval  $[f(x) - \varepsilon, f(x) + \varepsilon]$ . Thus, the above condition is equivalent to  $f(x^*) + \varepsilon \geq f(x) - \varepsilon$ , i.e., to  $f(x^*) \geq f(x) - 2 \cdot \varepsilon$ .

So, it is desirable to describe the set of all the values  $x^*$  for which  $f(x^*) \geq f(x) - 2 \cdot \varepsilon$  for all  $x \in X$ , i.e., in formal notations, for which

$$\forall x (f(x^*) \geq f(x) - 2 \cdot \varepsilon). \tag{2}$$

It is reasonable to call the alternatives  $x^*$  that satisfy this requirement  $(2 \cdot \varepsilon)$ -*optima*. In these terms, instead of computing the set of all the optima, we are arriving at a modified formalization of the original practical problem: compute the set of all  $(2 \cdot \varepsilon)$ -optima.

**Additional complication: the approximation accuracy is also not exactly known.** The above modified formulation implicitly assumes that we know the exact approximation accuracy  $\varepsilon$ . In other words, we assume that we know the exact value  $\varepsilon$  such that smaller differences between the values of the objective function  $f(x)$  can be safely ignored.

In practice, of course, this “threshold” value  $\varepsilon$  is also known with uncertainty. A reasonable person can say that, e.g., 1% difference in the values of the objective function can be safely ignored but 2% difference is no longer negligible. However, it is difficult to expect a user to claim that a difference below 1.235% can be safely ignored, while any difference above this threshold value 1.235% is not negligible.

In other words, instead of a single exact value  $\varepsilon$ , we usually have two bounds  $\underline{\varepsilon} < \bar{\varepsilon}$ , so that:

- every difference smaller than  $\underline{\varepsilon}$  can be safely ignored, while
- differences larger than  $\bar{\varepsilon}$  cannot be ignored.

**Computing the set of optima: the final formulation of the problem.** If we take into account the uncertainty with which we know the accuracy  $\varepsilon$ , then we come to the following conclusion:

- every  $(2 \cdot \underline{\varepsilon})$ -optimal alternative is desirable, and
- every desirable alternative must be  $(2 \cdot \bar{\varepsilon})$ -optimal.

In other words, given an objective function  $f$ , we would like to compute a set  $S$  with the following two properties:

- every  $(2 \cdot \underline{\varepsilon})$ -optimal alternative belongs to the set  $S$ , and
- every alternative from the set  $S$  is  $(2 \cdot \bar{\varepsilon})$ -optimal.

**The resulting problem is algorithmically solvable.** It turns out that the problem of computing such a set  $S$  is already algorithmically solvable; see, e.g., [G.-Toth and Kreinovich 2009]. Specifically, it is possible to produce a finite list of elements  $L$  and a rational value  $\delta > 0$  such that the set  $S$  of all the alternatives which are  $\delta$ -close to one of the elements of  $L$  is the desired set.

Formally, if we denote the set of all  $(2 \cdot \varepsilon)$ -optimal alternatives by  $M_{2 \cdot \varepsilon}(f)$ , then the following two conditions are satisfied:

- If  $x_0 \in M_{2,\underline{\varepsilon}}(f)$ , then  $d(x_0, x) \leq \delta$  for some  $\ell \in L$ .
- If  $d(x_0, \ell) \leq \delta$  for some  $\ell \in L$ , then  $x_0 \in M_{2,\bar{\varepsilon}}(f)$ .

In other words, the union of the corresponding balls  $B_\delta(\ell) \stackrel{\text{def}}{=} \{x : d(\ell, x) \leq \delta\}$  satisfies the following property:

$$M_{2,\underline{\varepsilon}}(f) \subseteq \bigcup_{\ell \in L} B_\delta(\ell) \subseteq M_{2,\bar{\varepsilon}}(f). \quad (3)$$

*Comment.* We do not describe the algorithm for computing the list  $L$ , since later in this paper, we present an algorithm for solving a more general problem.

*Comment about continuity.* The above result is based on the (implicit) assumption that the objective function  $f$  is continuous. Continuous objective functions describe the usual consequences of different actions, since usually a small change in the solution only leads to a small change in the consequences.

In principle, there are some cases when the objective function is *not* continuous. For example, for some undesired side products of an industrial process, there is usually a threshold beyond which heavy fines start. In such situations, however, the desire is to avoid exceeding this threshold. Thus, the environmentally proper way of handling these situations is *not* to incorporate these fines into the profit estimates, but rather to avoid such undesirable situations altogether, and to view these restrictions as constraints that limit the set  $X$  of possible solutions. On this restricted set, the objective function is continuous. Such constraint optimization problems will be discussed later in this section.

**A more general problem: computing the Pareto set.** The above description of the decision making problem assumes that we have a *single* objective function that we are trying to maximize – albeit an imprecisely known one. In other words, we assume that we have already agreed how to combine different characteristics describing different aspects of the problem into a single numerical quantity.

In practice, we usually have several objective functions

$$f(x) = (f_1(x), \dots, f_n(x))$$

describing different aspects of the possible solution  $x$ , such as profit, environmental friendliness, safety, etc. Ideally, we should maximize the values of *all* these characteristics, but in reality, there is often a trade-off: e.g., to achieve more environmental friendliness, it is often necessary to slightly decrease the profit; there is a similar trade-off between cost and durability.

In many situations, the user does not have a clear a priori idea which trade-offs are beneficial and which are not; in other words, the user does not have a

single combined objective function  $f(x)$  that would enable him or her to make an ultimate decision. In such situations, it is reasonable to present the user with the *set* of all possible solutions – and let the user decide between different possible solutions from this set. The only possible solutions  $x^*$  that we do not want to present to the user are solutions  $x^*$  which can be improved in all the senses, i.e., solutions for which, for some other solution  $x$ , we have  $f_j(x^*) \leq f_j(x)$  for all  $j$  and  $f_j(x^*) < f_j(x)$  for some  $j$ . The set of all such “non-improvable” solution is known as the *Pareto set*. The problem is how to compute the Pareto set.

This problem has many practical applications; see, e.g., [Figueira et al. 2004].

In formal terms, an alternative  $x^*$  is dominated if

$$\begin{aligned} \exists x (f_1(x) \geq f_1(x^*) \ \& \dots \ \& f_n(x) \geq f_n(x^*) \ \& \\ ((f_1(x) > f_1(x^*) \vee \dots \vee f_n(x) > f_n(x^*))) . \end{aligned} \quad (4)$$

Thus, the condition that  $x^*$  is *not* dominated (= Pareto optimal) takes the form

$$\begin{aligned} \forall x (f_1(x^*) > f_1(x) \vee \dots \vee f_n(x^*) > f_n(x) \vee \\ (f_1(x^*) \geq f_1(x) \ \& \dots \ \& f_n(x^*) \geq f_n(x))) . \end{aligned} \quad (5)$$

In these terms, the Pareto set is the set of all the alternatives  $x$  that satisfy the property (5).

**In general, the computation of a Pareto set is an algorithmically undecidable problem.** There exist efficient algorithms for computing the Pareto set for several important specific classes of problems: e.g., for special location problems [Nickel and Puerto 2005] and for problems with linear objective functions [Figueira et al. 2004].

In general, however, this problem is known to be computationally difficult; see, e.g., [Ruzika and Wiecek 2005]. This difficulty has a theoretical explanation – this problem is, in general, algorithmically undecidable. This undecidability directly follows from the fact that for  $n = 1$ , we get the problem of computing the set of optima, the problem which is (as we have mentioned earlier) algorithmically undecidable.

**The problem of computing a Pareto set becomes decidable if we take into account that the objective functions are known with some accuracy.** In practice, as we have mentioned, we know each of the objective functions  $f_j(x)$  only with some accuracy  $\varepsilon_j$ . It turns out that if we appropriately take this uncertainty into account, then (verified) algorithms for computing the resulting Pareto set become possible. Such algorithms were described, for the case of  $n = 2$  objective functions  $f_j$  defined on bounded subsets of  $\mathbb{R}^m$ , in [Fernández et al. 2006], [Fernández and Tóth 2006], [Tóth and Fernández 2006], [Fernández and Tóth 2007], [Fernández and Tóth 2009]. For the general case of

arbitrary computable objective functions defined on a general computable set  $X$ , the result is given in [G.-Toth and Kreinovich 2009].

*Comment.* A similar algorithm is presented in [Kubica and Woźniak 2008].

Specifically, we assume that for every  $j$ , we know the bounds  $\underline{\varepsilon}_j < \bar{\varepsilon}_j$  on the (unknown) accuracy  $\varepsilon_j$ . Similarly to the optimization case, for each combination of values  $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)$ , we say that an alternative  $x^*$  is  $(2 \cdot \varepsilon)$ -Pareto optimal if it satisfies the following property:

$$\begin{aligned} \forall x ((f_1(x^*) \geq f_1(x) - 2 \cdot \varepsilon_1) \vee \dots \vee (f_n(x^*) \geq f_n(x) - 2 \cdot \varepsilon_n) \vee \\ ((f_1(x^*) \geq f_1(x) - 2 \cdot \varepsilon_1) \& \dots \& (f_n(x^*) \geq f_n(x) - 2 \cdot \varepsilon_n))). \end{aligned} \quad (6)$$

*Comments.*

- Please note that since we are considering approximate values anyway, we replaced the strict equality  $f_1(x^*) > f_1(x) - 2 \cdot \varepsilon$  with a non-strict one  $f_1(x^*) \geq f_1(x) - 2 \cdot \varepsilon$ .
- After this replacement, the second part

$$((f_1(x^*) \geq f_1(x) - 2 \cdot \varepsilon) \& \dots \& (f_n(x^*) \geq f_n(x) - 2 \cdot \varepsilon))$$

of the condition (6) follows from the first one, so the requirement (6) take the simplified form:

$$\forall x ((f_1(x^*) \geq f_1(x) - 2 \cdot \varepsilon_1) \vee \dots \vee (f_n(x^*) \geq f_n(x) - 2 \cdot \varepsilon_n)). \quad (7)$$

*End of comments.*

Given a tuple of objective functions

$$f = (f_1, \dots, f_n),$$

we would like to compute a set  $S$  with the following two properties:

- every  $(2 \cdot \underline{\varepsilon})$ -Pareto optimal alternative belongs to the set  $S$ , and
- every alternative from the set  $S$  is  $(2 \cdot \bar{\varepsilon})$ -Pareto optimal.

Specifically, it is possible to produce a finite list of elements  $L$  and a rational value  $\delta > 0$  such that the set  $S$  of all the alternatives which are  $\delta$ -close to one of the elements of  $L$  is the desired set.

Formally, if we denote the set of all  $(2 \cdot \varepsilon)$ -Pareto optimal alternatives by  $P_{2 \cdot \varepsilon}(f)$ , then the following two conditions are satisfied:

- If  $x^* \in P_{2,\underline{\varepsilon}}(f)$ , then  $d(x^*, \ell) \leq \delta$  for some  $\ell \in L$ .
- If  $d(x^*, \ell) \leq \delta$  for some  $\ell \in L$ , then  $x^* \in P_{2,\bar{\varepsilon}}(f)$ .

In other words, the union of the corresponding balls  $B_\delta(\ell) \stackrel{\text{def}}{=} \{x : d(\ell, x) \leq \delta\}$  satisfies the following property:

$$P_{2,\underline{\varepsilon}}(f) \subseteq \bigcup_{\ell \in L} B_\delta(\ell) \subseteq P_{2,\bar{\varepsilon}}(f). \quad (8)$$

**A similar problem: computing the set of Nash equilibria.** In the previous text, we considered the problem of selecting an alternative in which one person (or one entity) makes the decision, and the results depend only on this person's decision.

In practice, often, we have several different persons, with potentially different objective functions  $f_1(x), \dots, f_m(x)$ , and we need to take into account the interests of all the participants. Such decision problems are handled in game theory. One of the most widely used solution concept is the concept of *Nash equilibrium*: participants select a joint decision  $x^* = (x_1^*, \dots, x_m^*)$  in such a way that none of them has the incentive to unilaterally change the decision. In other words, for every  $x_i \in X_i$ , we have

$$f_i(x_1^*, \dots, x_{i-1}^*, x_i^*, x_{i+1}^*, \dots, x_m^*) \geq f_i(x_1^*, \dots, x_{i-1}^*, x_i, x_{i+1}^*, \dots, x_m^*). \quad (9)$$

Formally, we can say that  $x^* = (x_1^*, \dots, x_m^*)$  is a Nash equilibrium if the following condition holds:

$$\begin{aligned} & (\forall x_1 (f_1(x_1^*, x_2^*, \dots, x_m^*) \geq f_1(x_1, x_2^*, \dots, x_m^*)) \ \& \ \dots \ \& \\ & (\forall x_m (f_m(x_1^*, \dots, x_{m-1}^*, x_m^*) \geq f_m(x_1^*, \dots, x_{m-1}^*, x_m))))). \end{aligned} \quad (10)$$

**In general, the computation of the set of all Nash equilibria is an algorithmically undecidable problem.** Indeed, similarly to the Pareto case, this undecidability directly follows from the fact that for  $m = 1$ , we get the problem of computing the set of optima, the problem which is (as we have mentioned earlier) algorithmically undecidable.

**The problem of computing the set of all Nash equilibria becomes decidable if we take into account that the objective functions are known with some accuracy.** In practice, as we have mentioned, we know each of the objective functions  $f_j(x)$  only with some accuracy  $\varepsilon_j$ . It turns out that if we appropriately take this uncertainty into account, then (verified) algorithms for computing the resulting set of (approximate) Nash equilibria become possible; see, e.g., [Kubica and Woźniak 2010].

**The above three cases are similar.** The above three cases are similar, the results and algorithms are similar. It is therefore desirable to find a general formulation of a decision making problem that would include these results as particular cases.

This desirability also comes from the fact that several other practically important decision making problems can be formulated in a similar manner – so a general result will enable us to solve all these problems as well.

**Another example of a similar problem.** An example of such a decision making problem is the problem of constraint optimization. In general, we can have constraints of the equality type  $a(x) = b(x)$  and constraints of the inequality type  $a(x) \geq b(x)$ . By moving all the terms to one side, we can have an equivalent reformulation as  $c(x) = 0$  and  $c(x) \geq 0$ , where  $c(x) \stackrel{\text{def}}{=} a(x) - b(x)$ . Every constraint  $c(x) = 0$  of the equality type can be represented as two inequality constraints  $c(x) \geq 0$  and  $-c(x) \geq 0$ . Thus, in general, we can formulate the constraints optimization problem as follows: optimizing a given objective function  $f(x)$  under constraints  $c_1(x) \geq 0, \dots, c_n(x) \geq 0$ .

For the alternative  $x^*$  to be the location of the conditional optimum, this alternative must satisfy the following two requirements:

- the alternative  $x^*$  must satisfy all  $n$  constraints  $c_i(x^*) \geq 0$ , and
- for every other alternative  $x$  that satisfies all  $n$  constraints, we must have  $f(x^*) \geq f(x)$ .

Formally, these two requirements have the following form

$$c_1(x^*) \geq 0 \& \dots \& c_n(x^*) \geq 0 \& \\ \forall x ((c_1(x) \geq 0 \& \dots \& c_n(x) \geq 0) \rightarrow f(x^*) \geq f(x)). \quad (11)$$

Replacing implication  $A \rightarrow B$  with the equivalent formula  $B \vee \neg A$ , we get the following equivalent reformulation of (11) which makes it even closer to our previous problems:

$$c_1(x^*) \geq 0 \& \dots \& c_n(x^*) \geq 0 \& \\ \forall x (f(x^*) \geq f(x) \vee c_1(x) < 0 \vee \dots \vee c_n(x) < 0). \quad (12)$$

It is desirable to prove that – similarly to the above results – the natural  $\varepsilon$ -approximation is computable.

*Comment.* For conditions of the type  $c(x) \geq 0$ , the effect of inaccuracy is somewhat different than for inequalities of the above type  $f(x) \geq f(x')$ .

Indeed, let us denote, by  $\varepsilon$ , the accuracy below which differences in the values of  $c(x)$  can be safely ignored. Thus, the given value  $c(x)$  means that the actual



(unknown) value  $c_{\text{act}}(x)$  of the corresponding quantity can be any number from the interval  $[c(x) - \varepsilon, c(x) + \varepsilon]$ .

The constraint  $c(x) \geq 0$  means, in these terms, that there exist a value  $c_{\text{act}}(x) \in [c(x) - \varepsilon, c(x) + \varepsilon]$  for which  $c_{\text{act}}(x) \geq 0$ .

If one of the elements of the interval  $[c(x) - \varepsilon, c(x) + \varepsilon]$  is larger than or equal to 0, then the largest element  $c(x) + \varepsilon$  of the interval

$$[c(x) - \varepsilon, c(x) + \varepsilon]$$

is larger than or equal to 0:  $c(x) + \varepsilon \geq 0$ .

Vice versa, if  $c(x) + \varepsilon \geq 0$ , then an element  $c(x) + \varepsilon$  of the interval

$$[c(x) - \varepsilon, c(x) + \varepsilon]$$

is larger than or equal to 0. Thus, the above condition is equivalent to  $c(x) + \varepsilon \geq 0$ , i.e., to  $c(x) \geq -\varepsilon$ .

Note that here we have  $-\varepsilon$  instead of  $-2 \cdot \varepsilon$ .

**Other possible examples.** Similarly, we can consider Pareto optimization under constraints, Nash equilibrium under constraints, etc.

Another case is when we do not have any objective function, we simply want to find all the alternatives that satisfy the given constraint(s).

Yet another case is when we want to find a alternative  $x^*$  which guarantees a certain level of outcome no matter what alternative  $y$  is selected by the second participant:  $\forall y (f(x^*, y) \geq f_0)$ , i.e., equivalently,  $\forall y (f(x^*, y) - f_0 \geq 0)$ .

It is also possible to look for a *maximin* solution, a solution  $x^*$  for which the worst-case outcome  $\min_y f(x^*, y)$  is the largest possible. In this solution,

- there is an alternative  $y_w$  for which the value  $f(x^*, y_w)$  is the smallest possible, and
- for every other selection  $x$ , the value  $f(x, y)$  can be smaller than or equal to  $f(x^*, y_w)$ , i.e., there exists  $y$  for which  $f(x, y) \leq f(x^*, y_w)$ .

Formally, this property has the form

$$\exists y_w (\forall y (f(x^*, y_w) \geq f(x^*, y)) \ \& \ \forall x \exists y (f(x^*, y_w) \geq f(x, y))).$$

In some problems, we look for *local optima*, i.e., for a value  $x^*$  for which, for all  $x$  within a certain radius  $d$  from  $x^*$ , we have  $f(x^*) \geq f(x)$ . Local optima are important in many practical problems: to separate an an image of an astronomical object into components; in spectroscopy to subdivide the observed spectrum into individual lines corresponding to different ions and chemical substances, etc.; see, e.g., [Villaverde and Kreinovich 1993].

Formally, a local maximum at  $x^*$  means that we have

$$\forall x (d(x, x^*) \leq d \rightarrow f(x^*) \geq f(x))$$

i.e.,

$$\forall x (d - d(x, x^*) \geq 0 \rightarrow f(x^*) \geq f(x)).$$

**What we do in this paper.** First, in Section 2, we recall the main definitions related to computability. Then, in Section 3, we formulate our main result. Its proof is presented in Section 4. Finally, in Section 5, we discuss the questions of computational complexity and feasibility of the resulting algorithms.

## 2 What is a computable set, what is a computable function: brief reminder

**Need to define computability.** In the global optimization problem, we have the set of alternatives  $X$  and we have an objective function  $f : X \rightarrow \mathbb{R}$  and we are interested in computing the set of all optima. In other problems, we have one or more functions, and we want to compute an appropriate set. In order to analyze these problems from the algorithmic viewpoint, we need to know how this information is represented in a computer, i.e., from the computational viewpoint. In other words, we must start with a “computable” set  $X$  and “computable” function(s)  $f$ , and we must generate the corresponding solution set  $S$ .

The notions of computable numbers, computable sets, and computable functions are known; they form the so-called *computable mathematics* (also known as *constructive mathematics*); see, e.g., [Beeson 1985], [Bishop and Bridges 1985], [Kushner 1985], [Beeson 1987], [Bridges and Vîță 2006], [Aberth 2007]. However, these notions are not unique: depending on the practical application, we may end up with different notions of constructive sets, constructive numbers, etc. Let us therefore analyze our problem from the computational viewpoint and see which definitions naturally appear.

**Towards a definition of a computable set.** Let us start with the representation of a set. The easiest set to represent in a computer is a finite set  $X = \{x^{(1)}, \dots, x^{(k)}\}$ : the finite set can be (and usually is) simply represented by listing all its elements  $x^{(1)}, \dots, x^{(k)}$ .

In real life, however, the set of alternatives is usually infinite, with one or more parameters which can take any values from certain intervals. In this case, it is not possible to *exactly* list all possible alternatives. It is also not possible to *exactly* produce the optimal solution to the optimization problem – e.g., to produce the exact real number, we need to describe infinitely many digits, and a computer can only produce finitely many digits in any given time interval. In

such cases, we can only generate an *approximation* to the optimal solution. For the notion of the approximation to be meaningful, we must be able, for every two given alternatives  $x, x' \in X$ , to describe how close these alternatives are. In other words, we need to be able to describe the *distance*  $d(x, x')$  between every two elements, i.e., the set  $X$  must be a *metric space*.

For given two elements  $x$  and  $x'$ , the distance  $d(x, x')$  is a real number. We cannot always compute this number exactly – this would require infinitely many bits, but we need to be able to compute the value of this metric with an arbitrary accuracy. In other words, the values of the distance must be a computable number in the following precise sense.

This real number can also only be computed with some accuracy. It is reasonable to say that a real number is computable if we can compute it with any given accuracy.

**Definition 1.** By a *computable real number*, we mean a pair  $(x, \mathcal{U})$ , where  $x$  is a real number, and the algorithm  $\mathcal{U}$ , given a natural number  $k$ , produces a rational number  $r_k$  for which  $|x - r_k| \leq 2^{-k}$ .

*Comment.* For example,  $\sqrt{2}$  is a computable real number because we can compute it with any given accuracy. Inside the computer, a computable number is represented by the algorithm  $\mathcal{U}$ . So, when we say that we can compute something (e.g.,  $x^2$ ) based on the computable real number input  $x$ , we mean that, based on the algorithm  $\mathcal{U}$  approximating the real number  $x$ , we can generate an algorithm approximating  $x^2$ .

It is known that standard arithmetic operations can be performed on computable real numbers: the sum, the difference, the product, etc., of two computable real numbers are computable as well. Similarly, for every computable real number  $x$ , the values  $\sin(x)$ ,  $\exp(x)$ ,  $\ln(x)$ , etc., are also computable; see, e.g., [Beeson 1985], [Bishop and Bridges 1985], [Kushner 1985], [Beeson 1987], [Bridges and Vîță 2006], [Aberth 2007].

Similarly, we can describe the notion of a computable set: we cannot list *exactly* all the elements of this set, but we should be able, for any given accuracy  $\varepsilon = 2^{-k}$ , to list all the elements with this accuracy, i.e., to produce a finite list  $\{x^{(1)}, \dots, x^{(k)}\}$  that represents all the elements from the set  $X$  with the accuracy  $\varepsilon$ . In other words, for every element  $x \in X$ , there is an  $\varepsilon$ -close element from this finite list, i.e., an element  $x^{(i)}$  for which  $d(x, x^{(i)}) \leq \varepsilon$ . Such a finite list is called an  $\varepsilon$ -*net*.

We must also be able to effectively compute the distance between any two listed elements – whether they are listed for the same accuracy  $2^{-k}$  or for two different accuracies  $2^{-k} \neq 2^{-k'}$ . Thus, we arrive at the following definitions.

**Definition 2.** Let  $(X, d)$  be a metric space, and let  $\varepsilon > 0$  be a real number. A finite set  $\{x^{(1)}, \dots, x^{(k)}\} \subseteq X$  is called an  $\varepsilon$ -net for  $X$  if for every  $x \in X$ , there exists an  $i$  for which  $d(x, x^{(i)}) \leq \varepsilon$ .

**Definition 3.** By a *computable set*, we mean a metric space  $(X, d)$  equipped with two algorithms:

- an algorithm that, given a natural number  $k$ , produces a (finite)  $2^{-k}$ -net  $X^{(k)}$  for  $X$ ; and
- an algorithm that for every two elements  $x \in X^{(k)}$  and  $x' \in X_{k'}$ , computes the distance  $d(x, x')$  (i.e., for any integer  $m > 0$ , computes a rational number which is  $2^{-m}$ -close to  $d(x, x')$ ).

*Comment 1.* For complete metric spaces, the existence of a finite  $\varepsilon$ -net for every  $\varepsilon > 0$  is equivalent to compactness. Because of this, what we call computable sets are sometimes called *computable compact sets*.

*Comment 2.* No additional information is required about the elements of each finite set

$$X^{(k)} = \{x^{(k,1)}, x^{(k,2)}, \dots, x^{(k,m_k)}\}.$$

Each element  $x^{(k,l)}$  can be represented, e.g., by its indices  $k$  and  $l$ .

**Example.** The simplest examples of computable sets are:

- A non-degenerate interval  $[a, \bar{a}]$ , with  $a < \bar{a}$ . For such an interval, we can take, as  $X^{(k)}$ , the set of all rational numbers of the type  $p/2^k$  (with integer  $p$ ) from this interval.
- A non-degenerate multi-interval (box)  $[a_1, \bar{a}_1] \times \dots \times [a_m, \bar{a}_m]$  with  $a_i < \bar{a}_i$  and the sup metric

$$d((a_1, \dots, a_m), (a'_1, \dots, a'_m)) = \max_{i=1, \dots, m} |a_i - a'_i|.$$

We can take, as  $X^{(k)}$ , the set of all rational-valued points  $(p_1/2^k, \dots, p_m/2^k)$  from this box.

For the Euclidean distance, we can choose a similar set but with coordinates of the type  $p_i/2^{k+k_0}$ , where  $2^{k_0} > \sqrt{m}$ .

A computable element can be now naturally defined as an element which can be approximated with any given accuracy.

**Definition 4.** Let  $(X, d)$  be a computable metric space, with  $2^{-k}$ -net  $X^{(k)}$ . By a *computable element* of  $X$ , we mean a pair  $(x, \mathcal{U})$ , where  $x \in X$  and  $\mathcal{U}$  is an algorithm that, given an integer  $k > 0$ , produces an element  $r^{(k)} \in X^{(k)}$  for which  $d(x, r^{(k)}) \leq 2^{-k}$ .

*Comment.* One can easily see that for the interval  $[\underline{a}, \bar{a}]$ , computable elements are simply computable real numbers from this interval. Similarly, for the  $m$ -dimensional box, computable elements are simply tuples of computable numbers  $(a_1, \dots, a_m)$  from this box.

**Need to define Cartesian products.** For problems like Nash equilibria, we need to describe a function  $f(x_1, \dots, x_n)$  of several variables  $x_1, \dots, x_m$ . In mathematical terms such functions can be described as follows.

Let  $X_i$  be the set of all possible values of  $x_i$ . Then, the set of all possible tuples  $x = (x_1, \dots, x_m)$  with  $x_i \in X_i$  is called a *Cartesian product*  $X_1 \times \dots \times X_m$  of the sets  $X_i$ . In these terms, a function  $f(x_1, \dots, x_m)$  of several variables can be described as a function from the Cartesian product to the set of all real numbers:  $f : X_1 \times \dots \times X_m \rightarrow \mathbb{R}$ . Thus, to handle such situations algorithmically, we must define a Cartesian product of computable sets (= constructive compact sets).

If  $X_1, \dots, X_m$  are computable sets with metrics  $d_i$ . Then, similarly to the above box example, we can define the following sup metric on the Cartesian product:

$$d((x_1, \dots, x_m), (x'_1, \dots, x'_m)) \stackrel{\text{def}}{=} \max(d_1(x_1, x'_1), \dots, d_m(x_m, x'_m)). \quad (13)$$

It is easy to check that for each  $k$ , once we select a  $2^{-k}$ -nets  $X_i^{(k)}$  for the sets  $X_i$  ( $i = 1, \dots, m$ ), the set of corresponding tuples

$$X^{(k)} \stackrel{\text{def}}{=} X_1^{(k)} \times \dots \times X_m^{(k)} = \{(x_{1,i_1}^{(k)}, \dots, x_{m,i_m}^{(k)}) : x_{1,i_1}^{(k)} \in X_1^{(k)}, \dots, x_{m,i_m}^{(k)} \in X_m^{(k)}\} \quad (14)$$

is a  $2^{-k}$ -net for the Cartesian product  $X$ . Indeed, for every tuple

$$x = (x_1, \dots, x_m) \in X_1 \times \dots \times X_m,$$

if for every  $i$ , the element  $x_i^{(k)} \in X_i^{(k)}$  is  $2^{-k}$ -close to  $x_i$ , i.e.,  $d_i(x_i^{(k)}, x_i) \leq 2^{-k}$ , then the tuple  $x^{(k)} \stackrel{\text{def}}{=} (x_1^{(k)}, \dots, x_m^{(k)})$  is  $2^{-k}$ -close to  $x$ :

$$d(x^{(k)}, x) = \max(d_1(x_1^{(k)}, x_1), \dots, d_m(x_m^{(k)}, x_m)) \leq 2^{-k}. \quad (15)$$

Thus, the Cartesian product is also a computable set (= constructive compact set).

**Definition of computable function.** To complete the description of our problems, we also need to define the notion of a computable function  $f$  from a computable set to real numbers. Intuitively, we must be able, given an arbitrary computable element  $x \in X$ , to compute the value  $f(x)$ . In the computer, a computable element is given by its  $2^{-l}$ -approximations  $r_l$ . Thus, the only way to compute  $f(x)$  with a given accuracy  $2^{-k}$  is to compute the value  $f(r_l)$  for an appropriate approximation  $r_l$  to  $x$ .

For example, since in the computer, the value  $\sqrt{2}$  is represented only approximately, to compute  $\sin(\sqrt{2})$  with a given accuracy, we must know with what accuracy we must determine  $\sqrt{2}$  to get the desired accuracy in  $\sin(\sqrt{2})$ .

So, we arrive at the following definition.

**Definition 5.** By a *computable function* from a computable set  $(X, d)$  (with  $2^{-k}$ -nets  $X_k$ ) to real numbers, we mean a function  $f : X \rightarrow \mathbb{R}$  which is equipped with two algorithms:

- an algorithm that, given a natural number  $k$  and an element  $x \in X_k$ , computes the value  $f(x)$  (i.e., for any integer  $m > 0$ , computes a rational number which is  $2^{-m}$ -close to  $f(x)$ );
- an algorithm that, given a natural number  $k$ , produces a natural number  $l$  for which  $d(x, x') \leq 2^{-l}$  implies  $|f(x) - f(x')| \leq 2^{-k}$ .

*Comment.* In other words, we must be able to compute both the values of the function  $f(x)$  and its *modulus of continuity*, i.e., the function  $\omega_f(\varepsilon)$  that transforms every real number  $\varepsilon$  into a number  $\delta = \omega_f(\varepsilon)$  for which  $d(x, x') \leq \delta$  implies  $|f(x) - f(x')| \leq \varepsilon$ .

As we have mentioned earlier, all standard computer-implemented functions such as  $\sqrt{\quad}$ ,  $\exp$ ,  $\sin$ ,  $\ln$ , etc., are computable in this sense. In particular, the possibility to find  $l$  from  $k$  is based on the fact that most of these functions have a Lipschitz property  $|f(x) - f(x')| \leq L \cdot d(x, x')$  for a known  $L$ .

It is also known that a composition of computable functions is also computable. Thus, all practical objective functions are computable in this sense.

**Operations on computable functions.** In the proof of our main result, we will need the following properties of computable functions:

- if  $f(x)$  is a computable function and  $c$  is a computable number, then the function  $c \cdot f(x)$  is also computable;
- if  $f(x)$  and  $g(x)$  are computable functions, then the difference  $f(x) - g(x)$ , the minimum  $\min(f(x), g(x))$  and  $\max(f(x), g(x))$  are also computable;

– if  $f : X \times Y \rightarrow \mathbb{R}$  is a computable function and  $X$  and  $Y$  are computable compact sets, then the functions  $\min_{x \in X} f(x, y)$  and  $\max_{x \in X} f(x, y)$  are also computable.

For the first two properties, the computability of the new functions is straightforward. The corresponding moduli of continuity are obtained as follows.

**Multiplying a computable function by a computable number.** For  $h(x) = c \cdot f(x)$ , we can take an upper bound  $\Delta > 0$  for  $|c|$  and take  $\omega_h(\varepsilon) = \omega_f\left(\frac{\varepsilon}{\Delta}\right)$ .

With this choice,  $d(x, x') \leq \delta = \omega_h(\varepsilon)$  implies  $|f(x) - f(x')| \leq \frac{\varepsilon}{\Delta}$  and therefore,

$$|h(x) - h(x')| = |c| \cdot |f(x) - f(x')| \leq \Delta \cdot \frac{\varepsilon}{\Delta} = \varepsilon.$$

**Difference of computable functions.** For  $h(x) = f(x) - g(x)$ , we can take

$$\omega_h(\varepsilon) = \min\left(\omega_f\left(\frac{\varepsilon}{2}\right), \omega_g\left(\frac{\varepsilon}{2}\right)\right).$$

Indeed, in this case,  $d(x, x') \leq \delta = \omega_h(\varepsilon)$  implies  $d(x, x') \leq \omega_f\left(\frac{\varepsilon}{2}\right)$  and  $d(x, x') \leq \omega_g\left(\frac{\varepsilon}{2}\right)$  hence  $|f(x) - f(x')| \leq \frac{\varepsilon}{2}$  and  $|g(x) - g(x')| \leq \frac{\varepsilon}{2}$ . Thus,

$$\begin{aligned} |h(x) - h(x')| &= |f(x) - g(x) - (f(x') - g(x'))| = \\ &= |(f(x) - f(x')) - (g(x) - g(x'))| \leq \\ &= |f(x) - f(x')| + |g(x) - g(x')| \leq \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon. \end{aligned}$$

**Minimum of two computable functions.** For  $h(x) = \min(f(x), g(x))$ , we take

$$\omega_h(\varepsilon) = \min(\omega_f(\varepsilon), \omega_g(\varepsilon)).$$

In this case,  $d(x, x') \leq \delta = \omega_h(\varepsilon)$  implies  $d(x, x') \leq \omega_f(\varepsilon)$  and  $d(x, x') \leq \omega_g(\varepsilon)$  hence  $|f(x) - f(x')| \leq \varepsilon$  and  $|g(x) - g(x')| \leq \varepsilon$ .

In particular, we have

$$f(x) \leq f(x') + \varepsilon \text{ and } g(x) \leq g(x') + \varepsilon.$$

Since

$$\min(f(x), g(x)) \leq f(x) \text{ and } \min(f(x), g(x)) \leq g(x),$$

we conclude that

$$\min(f(x), g(x)) \leq f(x') + \varepsilon \text{ and } \min(f(x), g(x)) \leq g(x') + \varepsilon.$$

Since  $\min(f(x), g(x))$  is smaller than two numbers  $f(x') + \varepsilon$  and  $g(x') + \varepsilon$ , it is smaller than the smallest of these two numbers, i.e., that

$$\min(f(x') + \varepsilon, g(x') + \varepsilon) = \min(f(x'), g(x')) + \varepsilon.$$

Thus, we have

$$\min(f(x), g(x)) \leq \min(f(x'), g(x')) + \varepsilon.$$

Similarly, we have

$$\min(f(x'), g(x')) \leq \min(f(x), g(x)) + \varepsilon,$$

hence

$$|\min(f(x), g(x)) - \min(f(x'), g(x'))| \leq \varepsilon.$$

**Maximum of two computable functions.** For  $h(x) = \max(f(x), g(x))$ , we take the same value

$$\omega_h(\varepsilon) = \min(\omega_f(\varepsilon), \omega_g(\varepsilon)).$$

In this case,  $d(x, x') \leq \delta = \omega_h(\varepsilon)$  implies  $d(x, x') \leq \omega_f(\varepsilon)$  and  $d(x, x') \leq \omega_g(\varepsilon)$  hence  $|f(x) - f(x')| \leq \varepsilon$  and  $|g(x) - g(x')| \leq \varepsilon$ .

In particular, we have

$$f(x') \leq f(x) + \varepsilon \text{ and } g(x') \leq g(x) + \varepsilon.$$

Since

$$f(x) \leq \max(f(x), g(x)) \text{ and } g(x) \leq \max(f(x), g(x)),$$

we conclude that

$$f(x') \leq \max(f(x), g(x)) + \varepsilon \text{ and } g(x') \leq \max(f(x), g(x)) + \varepsilon.$$

Since both numbers  $f(x')$  and  $g(x')$  are smaller than  $\max(f(x), g(x)) + \varepsilon$ , the largest of these two numbers is also smaller than  $\max(f(x), g(x)) + \varepsilon$ :

$$\max(f(x'), g(x')) \leq \max(f(x), g(x)) + \varepsilon.$$

Similarly, we have

$$\max(f(x), g(x)) \leq \min(f(x'), g(x')) + \varepsilon,$$

hence

$$|\max(f(x), g(x)) - \min(f(x'), g(x'))| \leq \varepsilon.$$

**Minimum over a computable set: computability.** For  $h(y) = \min_{x \in X} f(x, y)$ , computability is not trivial. To compute the value  $h(y)$  with a given accuracy  $\varepsilon > 0$ , we:



- take  $\delta = \omega_f\left(\frac{\varepsilon}{2}\right)$ ,
- find a  $\delta$ -net  $x^{(1)}, \dots, x^{(m)}$  for the constructive set  $X$ ,
- compute the values  $f(x^{(i)}, y)$  with accuracy  $\frac{\varepsilon}{2}$ , resulting in approximate values  $\tilde{f}(x^{(1)}, y), \dots, \tilde{f}(x^{(m)}, y)$ , and
- compute  $\tilde{h}(y) \stackrel{\text{def}}{=} \min\left(\tilde{f}(x^{(1)}, y), \dots, \tilde{f}(x^{(m)}, y)\right)$ .

Let us show that this value is indeed an  $\varepsilon$ -approximation to  $h(y)$ .

Indeed, from the fact that each value  $\tilde{f}(x^{(i)}, y)$  is an  $(\varepsilon/2)$ -approximation to  $f(x^{(i)}, y)$ , we conclude that

$$f(x^{(i)}, y) \leq \tilde{f}(x^{(i)}, y) + \frac{\varepsilon}{2}$$

for all  $i = 1, \dots, m$ . Thus, the smallest of the left-hand sides is smaller than or equal to the smallest of the right-hand sides:

$$\min_i \left( f(x^{(i)}, y) \right) \leq \min_i \left( \tilde{f}(x^{(i)}, y) + \frac{\varepsilon}{2} \right) = \min_i \tilde{f}(x^{(i)}, y) + \frac{\varepsilon}{2} = \tilde{h}(y) + \frac{\varepsilon}{2}.$$

Since  $\min_i \left( f(x^{(i)}, y) \right) \geq \min_{x \in X} f(x, y) = h(y)$ , we thus conclude that

$$h(y) \leq \tilde{h}(y) + \frac{\varepsilon}{2} < \tilde{h}(y) + \varepsilon. \quad (16)$$

Vice versa, since  $X$  is a compact, the minimum  $h(y) = \min_{x \in X} f(x, y)$  of the function  $f(x, y)$  is attained for some  $x_0 \in X$ :  $f(x_0, y) = h(y)$ . Since the values  $x^{(1)}, \dots, x^{(m)}$  form a  $\delta$ -net, there exists an  $i$  for which  $d(x_0, x^{(i)}) \leq \delta$ . Due to the choice of  $\delta = \omega_f\left(\frac{\varepsilon}{2}\right)$ , this implies  $|f(x_0, y) - f(x^{(i)}, y)| \leq \frac{\varepsilon}{2}$ , hence

$$f(x^{(i)}, y) \leq f(x_0, y) + \frac{\varepsilon}{2} = h(y) + \frac{\varepsilon}{2}.$$

Since  $\tilde{f}(x^{(i)}, y)$  is an  $(\varepsilon/2)$ -approximation to  $f(x^{(i)}, y)$ , we conclude that

$$\tilde{f}(x^{(i)}, y) \leq f(x^{(i)}, y) + \frac{\varepsilon}{2}$$

hence

$$\tilde{f}(x^{(i)}, y) \leq \left( h(y) + \frac{\varepsilon}{2} \right) + \frac{\varepsilon}{2} = h(y) + \varepsilon.$$

Since one of the values  $\tilde{f}(x^{(i)}, y)$  does not exceed  $h(y) + \varepsilon$ , the smallest  $\tilde{h}(y)$  of these values also does not exceed  $h(y) + \varepsilon$ :

$$\tilde{h}(y) \leq h(y) + \varepsilon.$$

Together with (16), this implies that  $|\tilde{h}(y) - h(y)| \leq \varepsilon$ .

**Minimum over a computable set: modulus of continuity.** Let us show that for  $h(y) = \min_{x \in X} f(x, y)$ , we can take  $\omega_h(\varepsilon) = \omega_f(\varepsilon)$ .

Let us show that if  $d(y, y') \leq \delta = \omega_f(\varepsilon)$ , then  $h(y) \leq h(y') + \varepsilon$ . Indeed, since  $X$  is a compact set and  $f$  is a continuous function, there exists a value  $x_0$  for which  $h(y') = \min_{x \in X} f(x, y') = f(x_0, y')$ . Here,

$$d((x_0, y), (x_0, y')) = \max(0, d(y, y')) = d(y, y') \leq \delta.$$

Due to our choice of  $\delta = \omega_f(\varepsilon)$ , we have hence  $|f(x_0, y) - f(x_0, y')| \leq \varepsilon$ , hence  $f(x_0, y) \leq f(x_0, y') + \varepsilon = h(y') + \varepsilon$ . Since  $h(y) = \min_{x \in X} f(x, y) \leq f(x_0, y)$ , we thus conclude that  $h(y) \leq h(y') + \varepsilon$ .

Similarly, we can prove that  $h(y') \leq h(y) + \varepsilon$ , so indeed  $|h(y) - h(y')| \leq \varepsilon$ .

**Maximum over a computable set: computability.** To compute the value  $h(y) = \max_{x \in X} f(x, y)$  with a given accuracy  $\varepsilon > 0$ , we:

- take  $\delta = \omega_f\left(\frac{\varepsilon}{2}\right)$ ,
- find a  $\delta$ -net  $x^{(1)}, \dots, x^{(m)}$  for the constructive set  $X$ ,
- compute the values  $f(x^{(i)}, y)$  with accuracy  $\frac{\varepsilon}{2}$ , resulting in approximate values  $\tilde{f}(x^{(1)}, y), \dots, \tilde{f}(x^{(m)}, y)$ , and
- compute  $\tilde{h}(y) \stackrel{\text{def}}{=} \max\left(\tilde{f}(x^{(1)}, y), \dots, \tilde{f}(x^{(m)}, y)\right)$ .

Let us show that this value is indeed an  $\varepsilon$ -approximation to  $h(y)$ .

Indeed, from the fact that each value  $\tilde{f}(x^{(i)}, y)$  is an  $(\varepsilon/2)$ -approximation to  $f(x^{(i)}, y)$ , we conclude that

$$\tilde{f}(x^{(i)}, y) \leq f(x^{(i)}, y) + \frac{\varepsilon}{2}$$

for all  $i = 1, \dots, m$ . Thus, the largest of the left-hand sides is smaller than or equal to the largest of the right-hand sides:

$$\tilde{h}(y) = \max_i \left( \tilde{f}(x^{(i)}, y) \right) \leq \max_i \left( f(x^{(i)}, y) + \frac{\varepsilon}{2} \right) = \max_i f(x^{(i)}, y) + \frac{\varepsilon}{2}.$$

Since  $\max_i f(x^{(i)}, y) \leq \max_{x \in X} f(x, y) = h(y)$ , we thus conclude that

$$\tilde{h}(y) \leq h(y) + \frac{\varepsilon}{2} < h(y) + \varepsilon. \quad (17)$$

Vice versa, since  $X$  is a compact, the maximum  $h(y) = \max_{x \in X} f(x, y)$  of the function  $f(x, y)$  is attained for some  $x_0 \in X$ :  $f(x_0, y) = h(y)$ . Since the values

$x^{(1)}, \dots, x^{(m)}$  form a  $\delta$ -net, there exists an  $i$  for which  $d(x_0, x^{(i)}) \leq \delta$ . Due to the choice of  $\delta = \omega_f\left(\frac{\varepsilon}{2}\right)$ , this implies  $|f(x_0, y) - f(x^{(i)}, y)| \leq \frac{\varepsilon}{2}$ , hence

$$h(y) = f(x_0, y) \leq f(x^{(i)}, y) + \frac{\varepsilon}{2}.$$

Since  $\tilde{f}(x^{(i)}, y)$  is an  $(\varepsilon/2)$ -approximation to  $f(x^{(i)}, y)$ , we conclude that

$$f(x^{(i)}, y) \leq \tilde{f}(x^{(i)}, y) + \frac{\varepsilon}{2}$$

hence

$$h(y) \leq \left(\tilde{f}(x^{(i)}, y) + \frac{\varepsilon}{2}\right) + \frac{\varepsilon}{2} = \tilde{f}(x^{(i)}, y) + \varepsilon.$$

Here,

$$\tilde{f}(x^{(i)}, y) \leq \max_j \tilde{f}(x^{(j)}, y) = \tilde{h}(y),$$

hence

$$h(y) \leq \tilde{h}(y) + \varepsilon.$$

Together with (17), this implies that  $|\tilde{h}(y) - h(y)| \leq \varepsilon$ .

**Maximum over a computable set: modulus of continuity.** Let us show that for  $h(y) = \max_{x \in X} f(x, y)$ , we can take  $\omega_h(\varepsilon) = \omega_f(\varepsilon)$ .

Let us show that if  $d(y, y') \leq \delta = \omega_f(\varepsilon)$ , then  $h(y) \geq h(y') - \varepsilon$ . Indeed, since  $X$  is a compact set and  $f$  is a continuous function, there exists a value  $x_0$  for which  $h(y') = \max_{x \in X} f(x, y') = f(x_0, y')$ . Here,

$$d((x_0, y), (x_0, y')) = \max(0, d(y, y')) = d(y, y') \leq \delta.$$

Due to our choice of  $\delta = \omega_f(\varepsilon)$ , we have hence  $|f(x_0, y) - f(x_0, y')| \leq \varepsilon$ , hence  $f(x_0, y) \geq f(x_0, y') - \varepsilon = h(y') - \varepsilon$ . Since  $h(y) = \max_{x \in X} f(x, y) \geq f(x_0, y)$ , we thus conclude that  $h(y) \geq h(y') - \varepsilon$ .

Similarly, we can prove that  $h(y') \geq h(y) - \varepsilon$ , so indeed  $|h(y) - h(y')| \leq \varepsilon$ .

**Towards the main result.** Now, we have all the desired definitions, so we are ready to start the formulation and the analysis of our problem – of computing different types of sets.

### 3 Definitions and the main result

**Discussion.** All above definitions of decision-related properties were formed in a similar manner: we had basic expressions of the type  $a \geq b$  or  $a > b$  or  $a = b$ , and we combined them by using logical connectives  $\vee$ ,  $\&$ ,  $\neg$  (“not”),  $\rightarrow$ , and quantifiers  $\exists t$  and  $\forall t$ . An additional restriction is that the basic expressions

contained one unknown function: we had expressions of the type  $f_i(\dots) \geq f_i(\dots)$  or  $f_i(\dots) \geq 0$  – but not, e.g., expressions of the type  $f_i \geq f_j$  for  $i \neq j$ .

To analyze such expressions, let us perform some simplifications. First, we can eliminate equalities by replacing each equality  $a = b$  with an equivalent combination of two inequalities  $(a \geq b) \ \& \ (b \geq a)$ .

Second, we can replace each implication  $A \rightarrow B$  with an equivalent logical formula  $B \vee \neg A$ , this eliminating all implication symbols too.

Third, we can move negations inside the formulas, so that negations appear only in front of basic expressions:

- we replace  $\neg(A \ \& \ B)$  with an equivalent formula  $\neg A \vee \neg B$ ;
- we replace  $\neg(A \vee B)$  with an equivalent formula  $\neg A \ \& \ \neg B$ ;
- we replace  $\neg(\exists t \ A(t))$  with an equivalent formula  $\forall t \ (\neg A(t))$ ; and
- we replace  $\neg(\forall t \ A(t))$  with an equivalent formula  $\exists t \ (\neg A(t))$ .

Now, when negations are only at basic inequality expressions:

- we replace  $\neg(a \geq b)$  with  $b > a$  and
- we replace  $\neg(a > b)$  with  $b \geq a$ .

Thus, we eliminate all the negation symbols as well.

Thus, we have basic expressions of the type  $a \geq b$  and  $a > b$ , and a general formula can be obtained by using  $\vee$ ,  $\&$ , and quantifiers. Finally, as we have mentioned in Section 1, when we have a strict inequality, we replace it with a non-strict one anyway. Thus, we arrive at the following definition.

**Definition 6.** Let  $X_1, \dots, X_m$  be computable compact sets, and let  $f_i$ ,  $i = 1, \dots, n$ , be computable functions from  $X = X_1 \times \dots \times X_m$  to the set  $\mathbb{R}$  of real numbers.

- By a  *$f_i$ -expression*, we mean a formula of the type  $f_i(x_1, \dots, x_m) \geq 0$  or of the type  $f_i(x_1, \dots, x_m) \geq f(x'_1, \dots, x'_m)$ , where  $x_i$  and  $x'_i$  are variables (possibly coinciding).
- By a *decision-related property*, we mean an expression that is obtained from  $f_i$ -expressions by using  $\vee$ ,  $\&$ , and quantifiers  $\forall t_{t \in X_i}$  and  $\exists t_{t \in X_i}$ .
- For each decision-related property  $P$  with free variables  $z = (z_1, \dots)$ , by an *decision-related set*, we mean the set of all the tuples  $z$  that satisfy the property  $P$ .

*Comment.* One can easily check that all the above examples are particular cases of this general definition.

**Definition 7.** Let  $X_1, \dots, X_m$  be computable compact sets, let  $f_i, i = 1, \dots, n$ , be computable functions from  $X = X_1 \times \dots \times X_m$  to the set  $\mathbb{R}$  of real numbers, and let  $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)$  be a tuple of positive real numbers.

– By an  $\varepsilon$ -version of a decision-related property  $P$ , we mean the formula  $P_\varepsilon$  in which

- each  $f_i$ -expression of the type  $f_i(x_1, \dots, x_m) \geq 0$  is replaced by

$$f_i(x_1, \dots, x_m) \geq -\varepsilon_i,$$

and

- each  $f_i$ -expression of the type  $f_i(x_1, \dots, x_m) \geq f(x'_1, \dots, x'_m)$  is replaced by  $f_i(x_1, \dots, x_m) \geq f(x'_1, \dots, x'_m) - 2 \cdot \varepsilon_i$ .

- We say that a tuple  $z$   $\varepsilon$ -satisfies the property  $P$  if it satisfies the formula  $P_\varepsilon$ .
- The set of all the tuples that  $\varepsilon$ -satisfy the property  $P$  will be denoted by  $S_\varepsilon$ .

**Example: inequality.** In particular, an inequality  $f_i \geq 0$  is transformed into  $f_i \geq -\varepsilon_i$ . For this constraint example, the  $\varepsilon$ -modification can be justified by the following simple result:

**Definition 8.** Let  $\varepsilon = (\varepsilon_1, \dots, \varepsilon_n)$  be a tuple of positive real numbers.

- We say that a function  $f_i(x)$  is  $\varepsilon_i$ -close to a function  $g_i(x)$  if

$$|f_i(x) - g_i(x)| \leq \varepsilon_i$$

for all  $x$  and  $i$ .

- We say that a tuple of functions  $f(x) = (f_1(x), \dots, f_n(x))$  is  $\varepsilon$ -close to a tuple  $g(x) = (g_1(x), \dots, g_n(x))$  if  $|f_i(x) - g_i(x)| \leq \varepsilon_i$  for all  $x$  and  $i$ .

**Proposition 9.** Let  $f_i(x)$  be a function,  $\varepsilon_i > 0$  be a number, and  $x^*$  be a value. Then, the following two conditions are equivalent to each other:

- the inequality  $f_i(x^*) \geq -\varepsilon_i$  holds for a given function  $f_i(x)$  and given value  $x^*$ ;
- the inequality  $g_i(x^*) \geq 0$  holds for some function  $g_i$  which is  $\varepsilon_i$ -close to  $f_i(x)$ .

**Example: equality.** The original equality  $f_i = 0$  is first represented as two inequalities  $f_i \geq 0$  and  $f_j \geq 0$ , where  $f_j \stackrel{\text{def}}{=} -f_i$ . For the computable function  $f_j = -f_i$ , the accuracy  $\varepsilon_j$  with which we know  $f_j$  is the same as the accuracy  $\varepsilon_i$  with which we know  $f_i$ : indeed,  $|f_j - \tilde{f}_j| = |-f_i - (-\tilde{f}_i)| = |f_i - \tilde{f}_i|$ . Thus, the two inequalities get replaced with two modified inequalities  $f_i \geq -\varepsilon_i$  and  $-f_i \geq -\varepsilon_i$ . The second modified inequality is equivalent to  $f_i \leq \varepsilon_i$  and thus, this system of two modified inequalities is equivalent to  $|f_i| \leq \varepsilon_i$ . This is a reasonable  $\varepsilon_i$ -approximate analogue of the original equality  $f_i = 0$ .

**Proposition 10.** *Let  $f_i(x)$  be a function,  $\varepsilon_i > 0$  be a number, and  $x^*$  be a value. Then, the following two conditions are equivalent to each other:*

- the inequality  $|f_i(x^*)| \leq \varepsilon_i$  holds for a given function  $f_i(x)$  and given value  $x^*$ ;
- the equality  $g_i(x^*) = 0$  holds for some function  $g_i$  which is  $\varepsilon_i$ -close to  $f_i(x)$ .

**Equalities can be safely added.** In general, our main theorems (see below) remain true if we allow equalities  $f_i(\cdot) = f_i(\cdot)$  and  $f_i(\cdot) = 0$  as  $f_i$ -expressions, and say that these equalities are  $\varepsilon_i$ -satisfied if, correspondingly,  $|f_i(\cdot) - f_i(\cdot)| \leq 2 \cdot \varepsilon_i$  and  $|f_i(\cdot)| \leq \varepsilon_i$ .

**Examples: optimality, Pareto optimality, and Nash equilibrium.** For optimality and Pareto optimality, results similar to Propositions 9 and 10 have been proven in [G.-Toth and Kreinovich 2009]. A similar result holds for the Nash equilibrium:

**Proposition 11.** *Let  $f : X_1 \times \dots \times X_m \rightarrow \mathbb{R}$  be a tuple of functions,  $\varepsilon$  be a tuple of numbers, and  $x_0 = (x_1, \dots, x_m) \in X_1 \times \dots \times X_m$  be a value. Then, the following two conditions are equivalent to each other:*

- the value  $x^*$  is a  $\varepsilon$ -Nash equilibrium, i.e., for every  $i$  and for every  $x_i \in X_i$ , we have

$$\begin{aligned} f_i(x_1^*, \dots, x_{i-1}^*, x_i^*, x_{i+1}^*, \dots, x_m^*) &\geq \\ f_i(x_1^*, \dots, x_{i-1}^*, x_i, x_{i+1}^*, \dots, x_m^*) &- 2 \cdot \varepsilon_i; \end{aligned} \quad (18)$$

- for some tuple of functions  $g(x)$  which is  $\varepsilon$ -close to  $f(x)$ , the value  $x^*$  is a Nash equilibrium, i.e., for every  $i$  and for every  $x_i \in X_i$ , we have:

$$g_i(x_1^*, \dots, x_{i-1}^*, x_i^*, x_{i+1}^*, \dots, x_m^*) \geq g_i(x_1^*, \dots, x_{i-1}^*, x_i, x_{i+1}^*, \dots, x_m^*). \quad (19)$$

**In general, negation cannot be safely added.** In the original transformation of general formulas, we eliminated equality and we eliminated negation. Let us show that while equality can be easily added to this formulation, it is not possible to consistency add negation. Indeed, let us consider a condition  $\neg(f_i > 0)$ .

- According to the current methodology, we replace  $\neg(f_i > 0)$  with  $-f_i \geq 0$  and then replace it with a modified inequality  $-f_i \geq -\varepsilon_i$ , i.e.,  $f_i \leq \varepsilon$ .
- On the other hand, if we could simply allow negations in the above definition, then, to modify the above formula, we would replace  $f_i > 0$  with  $f_i \geq -\varepsilon$ . Then, the original formula  $\neg(f_i > 0)$  would be replaced with a modified formula  $\neg(f_i \geq -\varepsilon)$ , i.e.,  $f_i < -\varepsilon$ .

One can see that the resulting formulas  $f_i \leq \varepsilon$  and  $f_i < -\varepsilon$  are indeed different. Moreover, the second formula  $f_i < -\varepsilon$  does not allow the possibility  $f_i = 0$  which is perfectly in line with the original formula  $\neg(f_i > 0)$ .

**Because of the difficulty with negation, the  $\varepsilon$ -modification is not always meaningful.** In Section 1, we gave examples of several problems for which  $\varepsilon$ -modifications make sense. It should be mentioned, however, that because of the above difficulty with negation, not all formulas get a meaningful modification. As an example, let us take the requirement

$$\forall x (f_i(x) = 0 \rightarrow f_j(x) = 0).$$

According to our methodology, we first eliminate equality and transform this requirement into

$$\forall x ((f_i(x) \geq 0 \& -f_i(x) \geq 0) \rightarrow (f_j(x) \geq 0 \& -f_j(x) \geq 0)).$$

Second, we eliminate implication, resulting in

$$\forall x (\neg(f_i(x) \geq 0 \& -f_i(x) \geq 0) \vee (f_j(x) \geq 0 \& -f_j(x) \geq 0)).$$

Third, we move negations inside, resulting in

$$\forall x (\neg(f_i(x) \geq 0) \vee \neg(-f_i(x) \geq 0) \vee (f_j(x) \geq 0 \& -f_j(x) \geq 0))$$

and in

$$\forall x ((-f_i(x) > 0) \vee (f_i(x) > 0) \vee (f_j(x) \geq 0 \& -f_j(x) \geq 0)).$$

The corresponding  $\varepsilon$ -modification has the form

$$\forall x ((-f_i(x) \geq -\varepsilon_i) \vee (f_i(x) \geq -\varepsilon_i) \vee (f_j(x) \geq -\varepsilon_j \& -f_j(x) \geq -\varepsilon_j)).$$

This modified condition is, however, meaningless because if it satisfied for all possible values  $x$ . Indeed:

- If  $f_i(x) \geq -\varepsilon_i$ , then the second condition  $f_i(x) \geq -\varepsilon_i$  from the disjunction is satisfied.
- Otherwise, if  $f_i(x) < -\varepsilon_i$ , then  $-f_i(x) > \varepsilon_i$  and since  $\varepsilon_i > -\varepsilon_i$ , the first condition  $-f_i(x) \geq -\varepsilon_i$  from the disjunction is satisfied.

**Monotonicity of  $\varepsilon$ -satisfaction.** In our proofs and algorithms, we will use the fact that if  $\underline{\varepsilon}_i \leq \bar{\varepsilon}_i$  for each  $i$ , then:

- the inequality  $f_i(x_1, \dots, x_m) \geq -\underline{\varepsilon}_i$  implies  $f_i(x_1, \dots, x_m) \geq -\bar{\varepsilon}_i$ , and
- the inequality  $f_i(x_1, \dots, x_m) \geq f(x'_1, \dots, x'_m) - 2 \cdot \underline{\varepsilon}_i$  implies  $f_i(x_1, \dots, x_m) \geq f(x'_1, \dots, x'_m) - 2 \cdot \bar{\varepsilon}_i$ .

Logical operations  $\vee$ ,  $\&$ ,  $\forall t_{i \in X_i}$ , and  $\exists t_{i \in X_i}$  are monotonic in terms of implication: e.g., if  $A$  implies  $A'$  and  $B$  implies  $B'$ , then  $A \vee B$  implies  $A' \vee B'$ .

We can therefore conclude that  $P_{\underline{\varepsilon}}$  implies  $P_{\bar{\varepsilon}}$ , i.e., that  $S_{\underline{\varepsilon}} \subseteq S_{\bar{\varepsilon}}$ .

**Theorem 12.** *There exists an algorithm that, given a decision-related property  $P$  and two tuples  $\underline{\varepsilon}$  and  $\bar{\varepsilon}$  of rational numbers for which  $0 < \underline{\varepsilon}_i < \bar{\varepsilon}_i$ , produces a finite list of elements  $L$  and a rational number  $\delta > 0$  with the following two properties:*

- *Every tuple  $z^*$  that  $\underline{\varepsilon}$ -satisfies the property  $P$  is  $\delta$ -close to some element from the list  $L$ .*
- *Every tuple that is  $\delta$ -close to some element of the list  $L$   $\bar{\varepsilon}$ -satisfies the property  $P$ .*

The list  $L$  and the accuracy  $\delta$  provide a description of the desired decision-related set. Specifically, the desired set  $S$  is approximated by the set of all the elements which are  $\delta$ -close to one of the elements from the given list, i.e., by the union of the corresponding balls  $B_\delta(\ell) \stackrel{\text{def}}{=} \{z : d(\ell, z) \leq \delta\}$ :

$$S_{\underline{\varepsilon}} \subseteq \bigcup_{\ell \in L} B_\delta(\ell) \subseteq S_{\bar{\varepsilon}}.$$

It is worth mentioning that while in some decision-related problems like optimization, the solution always exists, in other problems – such as constraint satisfaction – it is possible that there are no solutions. In such cases, the above algorithm will produce an empty list  $L = \emptyset$ .

*Comment.* The above result shows that we can compute the finite set  $L$  and the value  $\delta$ . From the common sense viewpoint, this seems to be a reasonable way to say that the solution set is computable, and, in our opinion, it makes perfect



practical sense for the above general problem. In other words, as a solution set, we present the union  $\bigcup_{\ell \in L} B_\delta(\ell)$ .

It is worth mentioning that in the previous text, we had a different – more general – definition of a computable set – as a computable compact set (also known as a constructive compact set). It is possible to strengthen Theorem 12 and select  $\delta$  in such a way that this union is a computable compact set:

**Theorem 13.** *There exists an algorithm that, given a decision-related property  $P$  and two tuples  $\underline{\varepsilon}$  and  $\bar{\varepsilon}$  of rational numbers for which  $0 < \underline{\varepsilon}_i < \bar{\varepsilon}_i$ , produces a finite list of elements  $L$  and a rational number  $\delta > 0$  with the following three properties:*

- *Every tuple  $z^*$  that  $\underline{\varepsilon}$ -satisfies the property  $P$  is  $\delta$ -close to some element from the list  $L$ .*
- *Every tuple that is  $\delta$ -close to some element of the list  $L$   $\bar{\varepsilon}$ -satisfies the property  $P$ .*
- *The set  $\bigcup_{\ell \in L} B_\delta(\ell)$  of all tuples which are  $\delta$ -close to some element of the list  $L$  is computable (i.e., is a computable compact set).*

## 4 Proofs

**Proof of Proposition 9.** If  $g_i(x^*) \geq 0$  and  $g_i$  is  $f_i$  is  $\varepsilon_i$ -close to  $f_i$ , then  $|f_i(x^*) - g_i(x^*)| \leq \varepsilon_i$  hence  $f_i(x^*) \geq g_i(x^*) - \varepsilon_i$  and so  $f_i(x^*) \geq -\varepsilon_i$ .

Vice versa, if  $f_i(x^*) \geq -\varepsilon_i$ , then for  $g_i(x) \stackrel{\text{def}}{=} f_i(x) + \varepsilon_i$ , we have  $g_i(x^*) \geq 0$  and at the same time  $|f_i(x) - g_i(x)| = \varepsilon_i \leq \varepsilon_i$ , so  $g_i$  and  $f_i$  are indeed  $\varepsilon_i$ -close.

**Proof of Proposition 10.** If  $g_i(x^*) = 0$  and  $g_i$  is  $f_i$  is  $\varepsilon_i$ -close to  $f_i$ , then  $|f_i(x^*) - g_i(x^*)| \leq \varepsilon_i$  hence  $|f_i(x^*)| \leq \varepsilon_i$ .

Vice versa, if  $|f_i(x^*)| \leq \varepsilon_i$ , then for  $g_i(x) \stackrel{\text{def}}{=} f_i(x) - f_i(x^*)$ , we have  $g_i(x^*) = 0$  and at the same time  $|f_i(x) - g_i(x)| = |f_i(x^*)| \leq \varepsilon_i$ , so  $g_i$  and  $f_i$  are indeed  $\varepsilon_i$ -close.

**Proof of Proposition 11.** Let us first assume that  $f$  and  $g$  are  $\varepsilon$ -close and the condition (19) holds, i.e.,

$$g_i(x_1^*, \dots, x_{i-1}^*, x_i^*, x_{i+1}^*, \dots, x_m^*) \geq g_i(x_1^*, \dots, x_{i-1}^*, x_i, x_{i+1}^*, \dots, x_m^*).$$

Then, due to  $|f_i(\cdot) - g_i(\cdot)| \leq \varepsilon_i$ , we have

$$f_i(x_1^*, \dots, x_{i-1}^*, x_i^*, x_{i+1}^*, \dots, x_m^*) \geq g_i(x_1^*, \dots, x_{i-1}^*, x_i^*, x_{i+1}^*, \dots, x_m^*) - \varepsilon_i \geq$$

$$g_i(x_1^*, \dots, x_{i-1}^*, x_i, x_{i+1}^*, \dots, x_m^*) - \varepsilon_i \geq f_i(x_1^*, \dots, x_{i-1}^*, x_i, x_{i+1}^*, \dots, x_m^*) - 2 \cdot \varepsilon_i,$$

i.e., the desired inequality (18).

Vice versa, let us assume that the inequality (18) holds. Let us define the new functions  $g_i$  as follows:

- for the given  $x^*$ , we take  $g_i(x^*) = f_i(x^*) + \varepsilon_i$ ;
- for all  $x \neq x^*$ , we take  $g_i(x) = f_i(x) - \varepsilon_i$ .

Then, every condition of the type

$$f_i(x^*) = f_i(x_1^*, \dots, x_{i-1}^*, x_i^*, x_{i+1}^*, \dots, x_m^*) \geq f_i(x_1^*, \dots, x_{i-1}^*, x_i, x_{i+1}^*, \dots, x_m^*) - 2 \cdot \varepsilon_i$$

with  $x_i \neq x_i^*$  implies

$$g_i(x^*) = f_i(x^*) + \varepsilon_i \geq (f_i(x_1^*, \dots, x_{i-1}^*, x_i, x_{i+1}^*, \dots, x_m^*) - 2 \cdot \varepsilon_i) + \varepsilon_i = f_i(x_1^*, \dots, x_{i-1}^*, x_i, x_{i+1}^*, \dots, x_m^*) - \varepsilon_i = g_i(x_1^*, \dots, x_{i-1}^*, x_i, x_{i+1}^*, \dots, x_m^*).$$

Thus, for the tuple of functions  $g(x)$ , the value  $x^*$  is indeed a Nash equilibrium. The proposition is proven.

*Comment.* Please note that the functions  $g_i(x)$  defined in the above proof are not continuous and thus, not computable. This is OK since the formulation of the proposition does not require computability of the functions  $g_i(x)$ .

### Proof of Theorems 12 and 13.

1°. Since Theorem 12 trivially follows from Theorem 13, it is sufficient to prove Theorem 13.

2°. Let us denote  $\alpha_0 \stackrel{\text{def}}{=} \max_i \left( \frac{\varepsilon_i}{\bar{\varepsilon}_i} \right)$ . Then, for every  $i$ , we have  $\frac{\varepsilon_i}{\bar{\varepsilon}_i} \leq \alpha_0$  hence  $\varepsilon_i \leq \alpha_0 \cdot \bar{\varepsilon}_i$ . Due to the monotonicity property of the set  $S_\varepsilon$ , we thus have

$$S_\varepsilon \subseteq S_{\alpha_0 \cdot \bar{\varepsilon}}. \quad (20)$$

We will show how to compute a finite set  $L$  and two rational values  $\underline{\delta} < \bar{\delta}$  for which

$$S_{\alpha_0 \cdot \bar{\varepsilon}} \subseteq \bigcup_{\ell \in L} B_{\underline{\delta}}(\ell) \subseteq \bigcup_{\ell \in L} B_{\bar{\delta}}(\ell) \subseteq S_{\bar{\varepsilon}}. \quad (21)$$

Due to (20), this implies that

$$S_\varepsilon \subseteq \bigcup_{\ell \in L} B_{\underline{\delta}}(\ell) \subseteq \bigcup_{\ell \in L} B_{\bar{\delta}}(\ell) \subseteq S_{\bar{\varepsilon}}. \quad (22)$$

3°. The sets  $S_{\alpha_0 \cdot \bar{\varepsilon}}$  and  $S_{\bar{\varepsilon}}$  are particular cases of the general set  $S_{\alpha \cdot \bar{\varepsilon}}$  corresponding to  $\alpha = \alpha_0 < 1$  and to  $\alpha = 1$ . Let us therefore find a general description of the set  $S_{\alpha \cdot \bar{\varepsilon}}$ . We will do it by following the structure of the properties: we start with the basic statements, and we show how this description can be extended to statements obtained by using logical connectives and quantifiers. 3.1°.

A basic statement  $f_i(x_1, \dots, x_m) \geq -\alpha \cdot \bar{\varepsilon}_i$  can be equivalently reformulated as  $g_i(x_1, \dots, x_m) \geq -\alpha$ , where

$$g_i(x_1, \dots, x_m) \stackrel{\text{def}}{=} \frac{f_i(x_1, \dots, x_m)}{\bar{\varepsilon}_i}.$$

Since  $f_i$  is a computable function and  $\bar{\varepsilon}_i$  is a computable number, the function  $g_i(x_1, \dots, x_m)$  is also computable.

3.2°. A basic statement  $f_i(x_1, \dots, x_m) - f(x'_1, \dots, x'_m) \geq -2 \cdot \alpha \cdot \bar{\varepsilon}_i$  can be equivalently reformulated as  $h_i(x_1, \dots, x_m) \geq -\alpha$ , where

$$h_i(x_1, \dots, x_m) \stackrel{\text{def}}{=} \frac{f_i(x_1, \dots, x_m) - f(x'_1, \dots, x'_m)}{2 \cdot \bar{\varepsilon}_i}.$$

Since  $f_i$  is a computable function and  $\bar{\varepsilon}_i$  is a computable number, the function  $h_i(x_1, \dots, x_m)$  is also computable.

3.3°. Let us assume that we have a property  $P \& P'$ , where properties  $P$  and  $P'$  have already been represented in the form  $f(z) \geq -\alpha$  and  $f'(z) \geq -\alpha$  for computable functions  $f(z)$  and  $f'(z)$ . Then, the property  $P \& P'$  can be represented as  $g(z) \geq -\alpha$ , where  $g(z) \stackrel{\text{def}}{=} \min(f(z), f'(z))$ .

Since both functions  $f(z)$  and  $f'(z)$  are computable, we can use a result from the previous section to conclude that the new function  $g(z)$  is also computable.

3.4°. Let us assume that we have a property  $P \vee P'$ , where properties  $P$  and  $P'$  have already been represented in the form  $f(z) \geq -\alpha$  and  $f'(z) \geq -\alpha$  for computable functions  $f(z)$  and  $f'(z)$ . Then, the property  $P \vee P'$  can be represented as  $g(z) \geq -\alpha$ , where  $g(z) \stackrel{\text{def}}{=} \max(f(z), f'(z))$ .

Since both functions  $f(z)$  and  $f'(z)$  are computable, we can use a result from the previous section to conclude that the new function  $g(z)$  is also computable.

3.5°. Let us assume that we have a property  $\exists t_{t \in X_i} P(t, z)$ , where the property  $P(t, z)$  have already been represented in the form  $f(t, z) \geq -\alpha$  for a computable function  $f(t, z)$ . Then, the property  $\exists t_{t \in X_i} P(t, z)$  can be represented as  $g(z) \geq -\alpha$ , where  $g(z) \stackrel{\text{def}}{=} \max_{t \in X_i} f(t, z)$ .

Since the function  $f(t, z)$  is computable, we can use a result from the previous section to conclude that the new function  $g(z)$  is also computable.

3.6°. Let us assume that we have a property  $\forall t_{t \in X_i} P(t, z)$ , where the property  $P(t, z)$  have already been represented in the form  $f(t, z) \geq -\alpha$  for a computable

function  $f(t, z)$ . Then, the property  $\forall t \in X_i P(t, z)$  can be represented as  $g(z) \geq -\alpha$ , where  $g(z) \stackrel{\text{def}}{=} \min_{t \in X_i} f(t, z)$ .

Since the function  $f(t, z)$  is computable, we can use a result from the previous section to conclude that the new function  $g(z)$  is also computable.

3.7°. Thus, an  $(\alpha \cdot \bar{\varepsilon})$ -version of each decision-related property can be reformulated in the equivalent form  $f(z) \geq -\alpha$  for an appropriate computable function  $f(z)$ . Thus,  $S_{\alpha \cdot \bar{\varepsilon}} = \{z : f(z) \geq -\alpha\}$ .

4°. Since the function  $f(z)$  corresponding to the desired decision-related property  $P$  is computable, there exists the corresponding algorithm  $\omega(\varepsilon)$ .

Let us take, as  $\varepsilon$ , the rational number  $\varepsilon = \frac{1 - \alpha_0}{4}$ , and let us take  $\bar{\delta} = \omega_f(\varepsilon)$ , and  $\underline{\delta} = \frac{\bar{\delta}}{2}$ .

Each  $z$  is a tuple of elements from computable (compact) sets  $X_i$ . Thus, the set  $Z$  of all possible values  $z$  is a Cartesian product of computable compact sets and thus, a computable compact set itself. In particular, this means that we can algorithmically compute a  $\underline{\delta}$ -net  $z^{(1)}, \dots, z^{(M)}$  for the set  $Z$ .

Since the function  $f(z)$  is computable, we can compute all the values  $f(z^{(i)})$  with arbitrary accuracy. Let us denote the rational number resulting from computing  $f(z^{(i)})$  with accuracy  $\varepsilon$  by  $\tilde{f}(z^{(i)})$ . As the desired set  $L$ , we will now take

$$L = \left\{ z^{(i)} : \tilde{f}(z^{(i)}) \geq -\alpha_0 - 2 \cdot \varepsilon \right\}. \quad (23)$$

(Both compared values  $\tilde{f}(z^{(i)})$  and  $-\alpha_0 - 2 \cdot \varepsilon$  are rational, so we can algorithmically check the above inequality.)

Let us prove that for these  $\underline{\delta}$ ,  $\bar{\delta}$ , and  $L$ , we indeed have the property (21).

4.1°. Let us first prove that  $S_{\alpha_0 \cdot \bar{\varepsilon}} \subseteq \bigcup_{\ell \in L} B_{\bar{\delta}}(\ell)$ , i.e., that if  $f(z) \geq -\alpha_0$ , then there exists an  $z^{(i)} \in L$  for which  $d(z, z^{(i)}) \leq \bar{\delta}$ .

Indeed, since the values  $z^{(i)}$  form a  $\underline{\delta}$ -net for the set  $Z$ , there exists an  $i$  for which  $d(z, z^{(i)}) \leq \underline{\delta}$ . All we need to prove now is that  $z^{(i)} \in L$ , i.e., by definition of the set  $L$ , that  $\tilde{f}(z^{(i)}) \geq -\alpha_0 - 2 \cdot \varepsilon$ .

Indeed, because of our choice of  $\underline{\delta}$  as  $\frac{1}{2} \cdot \omega_f(\varepsilon)$ , the condition  $d(z, z^{(i)}) \leq \underline{\delta} = \frac{1}{2} \cdot \omega_f(\varepsilon)$  implies that  $d(z, z^{(i)}) \leq \omega_f(\varepsilon)$ . By definition of the modulus of continuity, this implies that  $|f(z) - f(z^{(i)})| \leq \varepsilon$ . Thus,  $f(z^{(i)}) \geq f(z) - \varepsilon$  and since  $f(z) \geq -\alpha_0$ , we conclude that  $f(z^{(i)}) \geq -\alpha_0 - \varepsilon$ .

Now, by definition of  $\tilde{f}(z^{(i)})$ , we have  $|\tilde{f}(z^{(i)}) - f(z^{(i)})| \leq \varepsilon$  hence  $\tilde{f}(z^{(i)}) \geq f(z^{(i)}) - \varepsilon$ . We already know that  $f(z^{(i)}) \geq -\alpha_0 - \varepsilon$ , so we conclude that  $\tilde{f}(z^{(i)}) \geq -\alpha_0 - 2 \cdot \varepsilon$  hence  $z^{(i)} \in L$ . The statement is proven.

4.2°. Let us now prove that  $\bigcup_{\ell \in L} B_{\bar{\delta}}(\ell) \subseteq S_{\bar{\varepsilon}}$ , i.e., that if for some  $z \in Z$  and  $i$  for which  $z^{(i)} \in L$ , we have  $d(z, z^{(i)}) \leq \bar{\delta}$ , then  $f(z) \geq -1$ .

Indeed,  $z^{(i)} \in L$  means that  $\tilde{f}(z^{(i)}) \geq -\alpha_0 - 2 \cdot \varepsilon$ . Since  $\tilde{f}(z^{(i)})$  is an  $\varepsilon$ -approximation to  $f(z^{(i)})$ , we have  $f(z^{(i)}) \geq \tilde{f}(z^{(i)}) - \varepsilon$  and therefore,  $\tilde{f}(z^{(i)}) \geq -\alpha_0 - 3 \cdot \varepsilon$ .

Due to  $d(z, z^{(i)}) \leq \bar{\delta} = \omega_f(\varepsilon)$ , we have  $|f(z) - f(z^{(i)})| \leq \varepsilon$  hence  $f(z) \geq f(z^{(i)}) - \varepsilon$  and thus,  $f(z) \geq -\alpha_0 - 4 \cdot \varepsilon$ . By definition of  $\varepsilon$ , this means that  $f(z) \geq -1$ , i.e., that indeed  $\bigcup_{\ell \in L} B_{\bar{\delta}}(\ell) \subseteq S_{\bar{\varepsilon}}$ .

The property (21) is proven.

5°. To complete the proof, we must show that there exists a  $\delta$  for which  $\underline{\delta} \leq \delta \leq \bar{\delta}$  – and for which therefore

$$\bigcup_{\ell \in L} B_{\underline{\delta}}(\ell) \subseteq \bigcup_{\ell \in L} B_{\delta}(\ell) \subseteq \bigcup_{\ell \in L} B_{\bar{\delta}}(\ell),$$

for which the union  $\bigcup_{\ell \in L} B_{\delta}(\ell)$  is a computable compact set.

Indeed, for an arbitrary  $\delta$ , the condition that  $z \in \bigcup_{\ell \in L} B_{\delta}(\ell)$  means that there exists  $\ell \in L$  for which  $d(z, \ell) \leq \delta$ . This condition is equivalent to  $g(z) \leq \delta$ , where  $g(z) \stackrel{\text{def}}{=} \min_{\ell \in L} d(z, \ell)$ . Thus,  $\bigcup_{\ell \in L} B_{\delta}(\ell) = \{z : g(z) \leq \delta\}$ .

According to the properties of the minima of computable functions, the function  $g(z)$  is also a computable function on a computable compact set. According to [Bishop and Bridges 1985], for every computable function  $g(z)$  and for every two computable numbers  $\underline{\delta} < \bar{\delta}$ , we can algorithmically find a value  $\delta \in (\underline{\delta}, \bar{\delta})$  for which the set  $\{z : g(z) \leq \delta\}$  is a computable compact set.

For this  $\delta$ , the set  $\bigcup_{\ell \in L} B_{\delta}(\ell) = \{z : g(z) \leq \delta\}$  is computable. The theorem is proven.

## 5 Computational complexity and feasibility of the resulting algorithms

**Computational complexity: general case.** Once we established that the algorithms exist, the natural next question is: how efficient are these algorithms? According to the proofs, the above algorithms require that we consider all the elements of the corresponding  $\varepsilon$ -net, its number of steps grows as the number of these elements. For an  $m$ -dimensional box this number is  $\approx V/\varepsilon^m$ , so it grows exponentially with the dimension  $m$  of the box.

This is, however, acceptable, since in general, the optimization problems are NP-hard [Kreinovich et al. 1998], and therefore, the worst-case exponential time is inevitable (unless, of course, it turns out that, contrary to the expectations of

most computer scientists,  $P = NP$  and thus, all such problems can be solved in feasible (polynomial) time).

**Implementation of the above algorithms: interval computations.** The usual implementation of the above algorithms involve interval computations; see, e.g., [Jaulin et al. 2001, Moore et al. 2009].

Interval computations were originally designed to estimate the uncertainty of the result of data processing in situations in which we only know the upper bounds  $\Delta$  on the measurement errors. In this case, based on the measurement result  $\tilde{x}$ , we can only conclude that the actual (unknown) value  $x$  of the desired quantity is in the interval  $[\tilde{x} - \Delta, \tilde{x} + \Delta]$ .

In interval computations, at each intermediate stage of the computation, we have intervals of possible values of the corresponding quantities.

**From interval computations to more sophisticated set computations.**

In interval computations, at every intermediate stage of the computations, we only keep the intervals of possible values of each quantity, but we do not keep the information about the relations between these quantities. As a result, we often have bounds with excess width.

To remedy this problem, in [Ceberio et al. 2006], [Ceberio et al. 2007], and [Kreinovich 2009], we proposed an extension of interval technique to *set computations*, where on each stage, in addition to intervals of possible values of the quantities, we also keep sets of possible values of pairs (triples, etc.). As a result, in several practical problems, such as

- estimating statistics (variance, correlation, etc.) under interval uncertainty, and
- solutions to ordinary differential equations (ODEs) with given accuracy,

this new formalism enables us to find estimates in feasible (polynomial) time; see [Ceberio et al. 2007] and [Kreinovich 2009].

*Comment.* The idea of using a grid to describe and compute sets is similar to the ideas from [Escardó 2009], where real numbers are represented as  $\sum a_n \cdot 2^{-n}$ , with  $a_n \in \{-1, 1\}$ , and sets are described in terms of possible values of  $a_n$ .

Alternative approaches to efficient set representation and set computations can be found, e.g., in [Collins 2007] and [Collins 2007a].

## Acknowledgments

This work was supported in part by NSF grant HRD-0734825, by Grant 1 T36 GM078000-01 from the National Institutes of Health, by CCA'2009 conference, and by Ettore Majorana Center for Scientific Culture, Erice, Italy.

One of the authors (V.K.) is thankful to all the participants of CCA'2009 for valuable discussions.

## References

- [Aberth 2007] Aberth, O.: “Introduction to Precise Numerical Methods”, Academic Press, San Diego, California, 2007.
- [Beeson 1985] Beeson, M.: “Foundations of Constructive Mathematics: Metamathematical Studies”, Springer, Berlin/Heidelberg/New York, 1985.
- [Beeson 1987] Beeson, M.: “Some relations between classical and constructive mathematics”; *Journal of Symbolic Logic* 43 (1987) 228-246.
- [Bishop and Bridges 1985] Bishop, E., Bridges, D. S.: “Constructive analysis”, Springer-Verlag, Berlin-Heidelberg-New York, 1985.
- [Bridges and Viță 2006] Bridges, D. S., Viță, L. S.: “Techniques of Constructive Analysis”, Springer, New York, 2006.
- [Ceberio et al. 2006] Ceberio, M., Ferson, S., Kreinovich, V. et al.: “How To Take Into Account Dependence Between the Inputs: From Interval Computations to Constraint-Related Set Computations”, In: *Proc. 2nd Int’l Workshop on Reliable Engineering Computing*, Savannah, Georgia, February 22–24, 2006, pp. 127–154; final version in *Journal of Uncertain Systems* 1, No. 1 (2007), 11–34.
- [Ceberio et al. 2007] Ceberio, M., Kreinovich, V., Pownuk, A., Bede, B.: “From Interval Computations to Constraint-Related Set Computations: Towards Faster Estimation of Statistics and ODEs under Interval, p-Box, and Fuzzy Uncertainty”, In: Melin, P., Castillo, O., Aguilar, L. T., Kacprzyk, J., Pedrycz, W.: (eds.), *Foundations of Fuzzy Logic and Soft Computing, Proceedings of the World Congress of the International Fuzzy Systems Association IFSA’2007*, Cancun, Mexico, June 18–21, 2007, Springer Lecture Notes on Artificial Intelligence, 2007, Vol. 4529, pp. 33–42.
- [Collins 2007] Collins, P.: “Optimal semicomputable approximations to reachable and invariant sets”, *Theoretical Computer Science* 41, No. 1 (2007), 33–48.
- [Collins 2007a] Collins, P.: “Effective computations in linear systems”, In: *Computation and Logic in the Real World*, Lecture Notes in Computer Science 4497 (2007), 169–178.
- [Escardó 2009] Escardó, M. In: Bauer, A., Dillhage, R., Hertling, P., Ko, K.-I., and Rettinger, R. (eds.), *Proceedings of the Sixth International Conference on Computability and Complexity in Analysis CCA’2009*, Ljubljana, Slovenia, August 18–22, 2009, pp. 4–15.
- [Fernández and Tóth 2006] Fernández, J., Tóth, B.: “Obtaining the efficient set of biobjective competitive facility location and design problems”; In: “Proceedings of EURO XXI”, Reykjavík, Iceland, July 2–5, 2006.
- [Fernández and Tóth 2007] Fernández, J., Tóth, B.: “Obtaining an outer approximation of the efficient set of nonlinear biobjective problems”; *Journal of Global Optimization* 38, No. 2 (2007), 315-331.
- [Fernández and Tóth 2009] Fernández, J., Tóth, B.: “Obtaining the efficient set of nonlinear biobjective optimization problems via interval branch-and-bound methods”; *Computational Optimization and Applications* 42, No. 3 (April 2009), 393-419.
- [Fernández et al. 2006] Fernández, J., Tóth, B., Plastria, F., Pelegrín, B.: “Reconciling franchisor and franchisee: a planar multiobjective competitive location and design model”; In: “Recent Advances in Optimization”, Springer Lecture Notes in Economics and Mathematical Systems 563 (2006), 375-398.
- [Figueira et al. 2004] Figueira, J., Greco, S., Ehrgott, M. (eds.): “Multiple Criteria Decision Analysis: State of the Art Surveys”, Kluwer, Dordrecht, 2004.
- [Jaulin et al. 2001] Jaulin, L. et al.: “Applied Interval Analysis”, Springer, London, 2001.
- [Kreinovich 2009] Kreinovich, V.: “From Interval Computations to Constraint-Related Set Computations: Towards Faster Estimation of Statistics and ODEs Under Interval and P-Box Uncertainty”, In: Bauer, A., Dillhage, R., Hertling, P., Ko, K.-I., and

- Rettinger, R. (eds.), Proceedings of the Sixth International Conference on Computability and Complexity in Analysis CCA'2009, Ljubljana, Slovenia, August 18–22, 2009, pp. 4–15.
- [Kreinovich et al. 1998] Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P.: “Computational Complexity and Feasibility of Data Processing and Interval Computations”, Kluwer, Dordrecht, 1998.
- [Kubica and Woźniak 2008] Kubica, B. J., Woźniak, A.: “Interval methods for computing the Pareto-front of a multicriterial problem”; Lecture Notes in Computer Science, Springer, Berlin, 4967 (2008), 1382-1391.
- [Kubica and Woźniak 2010] Kubica, B. J., Woźniak, A.: “An interval method for seeking the Nash equilibria of non-cooperative games”; In: Wyrzykowski, R. (ed.) “Proceedings of the Eighth International Conference on Parallel Processing and Applied Mathematics PPAM'2009, Wrocław, Poland, September 13–16, 2009”; Lect. Notes Comp. Sci., Springer, Berlin (to appear).
- [Kushner 1985] Kushner, B. A.: “Lectures on Constructive Mathematical Analysis”; American Mathematical Society, Providence, Rhode Island, 1985.
- [Moore et al. 2009] Moore, R. E., Kearfott, R. B., Cloud, M. J.: “Introduction to interval analysis”, SIAM Press, Philadelphia, Pennsylvania, 2009.
- [Nachbar and Zame 1996] Nachbar, J. H., Zame, W. R.: “Non-computable strategies and discounted repeated games”; *Economic theory* 8 (1996), 103–122
- [Nickel and Puerto 2005] Nickel, S., Puerto, J.: “Location Theory: A Unified Approach”, Springer-Verlag, Berlin, 2005.
- [Ruzika and Wiecek 2005] Ruzika, S., Wiecek, M. M.: “Approximation methods in multiobjective programming”, *Journal of Optimization Theory and Applications* 126 (2005), 473-501.
- [Tóth and Fernández 2006] Tóth, B., Fernández, J.: “Obtaining the efficient set of nonlinear biobjective optimization problems via interval branch-and-bound methods”; In: “Proceedings of the 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics SCAN'06”, Duisburg, Germany, September 26–29, 2006.
- [G.-Toth and Kreinovich 2009] G.-Toth, B., Kreinovich, V.: “Validated methods for computing Pareto-sets: general algorithmic analysis”; *International Journal of Applied Mathematics and Computer Science* 19, 3 (2009), 369-380.
- [Villaverde and Kreinovich 1993] Villaverde, K., Kreinovich, V.: “A linear-time algorithm that locates local extrema of a function of one variable from interval measurement results”; *Interval Computations*, No. 4 (1993), 176–194.