Bounded Queries in Computability Theory
An Introduction
by
William Gasarch

1 Introduction

Notation 1.1 If $A \subseteq \mathsf{N}$ and $x \in \mathsf{N}$ then

$$A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \tag{1}$$

Notation 1.2 K denotes the HALTING set. This is tradition.

The following is the definition of a very basic function that is used repeatedly in the study of bounded queries in computability theory.

Definition 1.3 Let $n \geq 1$ and $A \subseteq \mathsf{N}$. C_n^A is the string-valued function defined by

$$\mathrm{C}_n^A(x_1, \ldots, x_n) = A(x_1) \cdots A(x_n).$$

$(A(x_1) \cdots A(x_n))$ denotes the *string of length* n that consists of the bits $A(x_1), \ldots, A(x_n)$, *not* multiplication. Hence $\mathrm{C}_n^A \colon \mathsf{N}^n \to \{0,1\}^n$.

Clearly, the function C_n^A can be computed with n queries to A. Can it be computed with fewer? The next theorem shows that if $A = K$ (K is the HALTING SET) and $n \geq 3$, the answer is yes. The proof of the theorem uses the following easy lemma, the proof of which we leave as an exercise.

Definition 1.4 If $A \subseteq \mathsf{N}$ and $n \in \mathsf{N}$. Then GE_n^A is the set of all tuples of naturals (any length) such that at least n of them are in A.

Definition 1.5 If $A, B \subseteq \mathsf{N}$ then $A \leq B$ means that there is a computable function f such that $x \in A$ iff $f(x) \in B$. Note that this means that any question to A can be answered by asking ONE question to B.

2 We CAN compute C_3^K with 2 queries

Lemma 2.1 *Let $n \in \mathsf{N}$.*

1. *Let A be any Σ_1 set. Then $\mathrm{GE}_n^A \le K$. Hence any question to GE_n^A can be answered using ONE query to K.*
2. *$\mathrm{GE}_n^K \le K$. Hence any question to GE_n^K can be answered using ONE query to K.*

Theorem 2.2 C_3^K *can be computed with two queries to K.*

Proof: The following is an algorithm that computes C_3^K with two queries to K.

1. Input (x_1, x_2, x_3).
2. Ask $(x_1, x_2, x_3) \in \mathrm{GE}_2^K$ (using ONE query to K by Lemma 2.1). Note that this is will tell if at least 2 of x_1, x_2, x_3 are in K.
3. There are two cases depending on the answer to the first query.
   (a) YES. So we know that at least 2 of $\{x_1, x_2, x_3\}$ are in K. Now ask $(x_1, x_2, x_3) \in \mathrm{GE}_3^K$ (using ONE query to K).
      i. If the answer is YES then we know that ALL THREE of $\{x_1, x_2, x_3\}$ are in K. Hence just output (1,1,1).
      ii. If the answer is NO then we know that *exactly TWO* of $\{x_1, x_2, x_3\}$ are in K. What do to? We can't ask any more questions! Here is what we do: WE RUN ALL THREE PROGRAMS UNTIL TWO OF THEM HALT. Here is the KEY: (1) we know that two of them WILL HALT, and (2) when we find the two that do halt we KNOW the other one WILL NEVER HALT so we need not run it anymore. Hence we know exactly which of $\{x_1, x_2, x_3\}$ are in HALT. Output the appropriate 3 bits.

   (b) NO. So we know that at at most 1 of $\{x_1, x_2, x_3\}$ is in K. Now ask $(x_1, x_2, x_3) \in \mathrm{GE}_1^K$ (using ONE query to K).

   i. If the answer is NO then we know that NONE OF $\{x_1, x_2, x_3\}$ are in K. Hence just output (0,0,0).

   ii. If the answer is YES then we know that *exactly ONE* of $\{x_1, x_2, x_3\}$ are in K. What do to? We can't ask any more questions! Here is what we do: WE RUN ALL THREE PROGRAMS UNTIL ONE OF THEM HALTS. Here is the KEY: (1) we know that one of them WILL HALT, and (2) when we find the one that does halt we KNOW the other two WILL NEVER HALT so we need not run it anymore. Hence we know exactly which of $\{x_1, x_2, x_3\}$ are in HALT. Output the appropriate 3 bits.

▮

3 We CANNOT Compute C_2^K with 1 query to X

SO we have C_3^K can be computed with 2 queries to K. Yeah! Can we do better? Can we compute C_3^K with 1 query to K? NO. In fact we will show that we can't compute C_2^K with 1 query to K. But wait! There's more! We will show that for ALL sets X, C_2^K cannot be computed with 1 query to X. We need a new way to look at the notion of 2-queries-to a set.

Definition 3.1 A partial function f (partial means that it need not be defined on all elements of N) is partial computable if there is TM M such that, for all x:

1. If $f(x)$ is defined then $M(x) \downarrow = f(x)$.
2. If $f(x)$ is not defined then $M(x) \uparrow$.

Lemma 3.2 *Let X be a set, and let f be a function that can be computed with one query to X. Then there are partial computable functions f_1, f_2 such that*

$$(\forall x)[f(x) \in \{f_1(x), f_2(x)\}].$$

Proof: Choose an algorithm ALG that computes f with at most one query to X. For $x \in \mathsf{N}$, define $f_1(x)$ and $f_2(x)$ as follows:

$f_1(x)$**:** Run ALG on input x, and answer the query with YES;

$f_2(x)$**:** Run ALG on input x, and answer the query with NO.

Note that, for every x, at least one of $f_1(x)$ and $f_2(x)$ converges and outputs $f(x)$. Hence $(\forall x)[f(x) \in \{f_1(x), f_2(x)\}]$. ∎

Note 3.3 Note that Lemma 3.2 also applies to any function f whose domain is N^n for some $n > 1$. If $n = 2$, the lemma's conclusion will be written as $(\forall x, y)[f(x, y) \in \{f_1(x, y), f_2(x, y)\}]$.

Theorem 3.4 *There is no set X such that C_2^K can be computed with one query to X.*

Proof: Let X be a set, and suppose C_2^K can be computed with one query to X. We show that K is decidable, a contradiction.

By Lemma 3.2 there exists partial computable functions f_1, f_2 so that

$$(\forall x, y)[\mathrm{C}_2^K(x, y) \in \{f_1(x, y), f_2(x, y)\}].$$

We can assume that, for each $i \in \{1, 2\}$ and all $x, y \in \mathsf{N}$, $f_i(x, y) \downarrow \implies f_i(x, y) \in \{0, 1\}^2$. Either the following algorithm, A1, will decide K, or the very reason why it fails to do so will yield an algorithm, A2, that *works*.

1. Input x.
2. Look for $y \in \mathsf{N}$ and $b, c_1, c_2 \in \{0, 1\}$ so that $f_1(x, y) \downarrow= (b, c_1)$ and $f_2(x, y) \downarrow= (b, c_2)$. (This step may not terminate.)
3. (If this step is reached, then y, b, c_1, c_2 were found in step 2.) Output b.

Let $x \in \mathsf{N}$. We show that *if* $\mathrm{A1}(x) \downarrow$, *then* $\mathrm{A1}(x) = K(x)$. So suppose that $\mathrm{A1}(x) \downarrow$. Since step 2 terminates, we have found $y \in \mathsf{N}$ and $b, c_1, c_2 \in \{0, 1\}$ such that $f_1(x, y) \downarrow= (b, c_1)$ and $f_2(x, y) \downarrow= (b, c_2)$; hence $\mathrm{C}_2^K(x, y) \in \{(b, c_1), (b, c_2)\}$, so $K(x) = b$.

Hence if $(\forall x)[\mathrm{A1}(x) \downarrow]$, then K is computable. So suppose there is some x_0 such that $\mathrm{A1}(x_0) \uparrow$. We use this x_0 to devise a new algorithm.

Since $\mathrm{A1}(x_0) \uparrow$, note that, for every y, it cannot be the case that $f_1(x_0, y)$ and $f_2(x_0, y)$ converge and their outputs agree on the first component. Let $b_0 = K(x_0)$.

1. Input y.
2. Compute the values of $f_1(x_0, y)$ and $f_2(x_0, y)$ simultaneously, stopping when you find some $b \in \{0, 1\}$ such that one of $f_1(x_0, y)$, $f_2(x_0, y)$ converges to (b_0, b).
3. (If this step is reached, then b was found in step 2.) Output b.

Let $y \in \mathsf{N}$. We show that $\mathrm{A2}(y) \downarrow= K(y)$. Since

$$\mathrm{C}_2^K(x_0, y) \in \{f_1(x_0, y), f_2(x_0, y)\},$$

we know that $(\exists i, b)[f_i(x_0, y) = (b_0, b)]$. Hence $\mathrm{A2}(y) \downarrow$. If in step 2 it is discovered that (say) $f_1(x_0, y) \downarrow= (b_0, b)$, then it cannot be the case that $(\exists b')[f_2(x_0, y) \downarrow= (b_0, b')]$, since this would imply that $f_1(x_0, y)$ and $f_2(x_0, y)$ converge and their outputs agree on the first component, contrary to the choice of x_0. Hence either $f_2(x_0, y)$ diverges, or it converges and is wrong on the first component. It follows that $f_1(x_0, y) = \mathrm{C}_2^K(x_0, y)$, so $K(y) = b = \mathrm{A2}(y)$. ▮

4 Questions to Ponder

The algorithm for C_3^K with 2 queries to K asked the queries sequentially. That is, the algorithm asked a question, got the answer, and then based on that answer asked another question. What if we demand the algorithm as the questions in parallel?

Ponder the following questions (the answers are known).

1. Is then an algorithm for C_3^K that uses 2 parallel queries to K?
2. Is then a set $X \subseteq \mathsf{N}$ and algorithm for C_3^K that uses 2 parallel queries to X?