The inverse of the Ackermann function is primitive recursive

Armando B. Matos

May 7, 2014

1 The Ackermann function and its inverse

The purpose of this personal note is to understand why the inverse of the Ackermann function is primitive recursive; this is mentioned in [3, 2].

Definition 1

\[
A(0, n) = n + 1 \quad (1)
\]
\[
A(m + 1, 0) = A(m, 1) \quad (2)
\]
\[
A(m + 1, n + 1) = A(m, A(m + 1, n)) \quad (3)
\]

The following table contains some values of \(A(m, n)\). The entries marked “…” correspond to very large integers.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>13</td>
<td>29</td>
<td>61</td>
<td>125</td>
<td>253</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>65533</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

Some simple results about the Ackermann function

The following simple facts will be useful. They include Lemmas 2.11, 2.12, and 2.13 of [1]. The proofs are easy exercises of mathematical induction.

**Lemma 1** For all \(m, n \in \mathbb{N}\)

\[
A(m, n) > m + n \quad (4)
\]
\[
A(m, n) < A(m, n + 1) \quad (5)
\]
\[
A(m, n) < A(m + 1, n) \quad (6)
\]
Proof. (4): Consider the values of $A$ along the path $(0, n) - (1, n) - \ldots - (m, n)$. We have $A(0, n) = n + 1$ and by (5) each step increases $A$ by at least 1 (there are $m$ steps). It follows that $A(m, n) \geq m + n + 1$ from which the result follows.

(5): Induction on $m$. The case $m = 0$ is simple. Assume that $A(m, n) < A(m, n + 1)$. We have $A(m + 1, x + 1) = A(m, A(m + 1, x)) > A(m + 1, x)$, where an easy consequence of (4), namely $A(m, n) > n$, was used.

(6): Induction on $m$ and $n$. The case $m = 0$ is simple. Assume $A(m, n) < A(m + 1, n)$. Then (case (2)): $A(m + 1, 0) = A(m, 1) < A(m + 1, 1) = A(m + 2, 0)$. The new induction hypothesis is $A(m + 1, n) < A(m + 2, n)$ and we get $A(m + 2, n + 2) = A(m + 1, A(m + 2, n)) > A(m + 1, A(m + 1, n)) > A(m, A(m + 1, n)) = A(m + 1, n + 1)$. Property (5) was used. □

2 The graph of the Ackermann function is primitive recursive

Theorem 1 The graph of the Ackermann function is primitive recursive.

Proof. Given $x$, $y$, and $z$ to test if $A(x, y) = z$, we can ignore the arguments $m$ and $n$ outside the rectangle $0 \leq m \leq z$, $0 \leq n \leq z$, as well as the values $A(m, n)$ greater than $z$, because, using Lemma 1:

1. If $m > z$, $A(m, n) > z$ for every $n$.
2. If $n > z$, $A(m, n) > z$ for every $m$.
3. If $A(m, n)$ is used as argument of another computation of $A$ (line (3) of Definition 1, page 1), the “final result” will be greater than $z$.

The computation of $A(m, n)$ is either

- Immediate, when $m = 0$: $A(0, n) = n + 1$.
- Dependent on $A(m - 1, 1)$ when $n = 0$: $A(m, 0) = A(m - 1, 1)$.
- Dependent on $A(m - 1, w)$ with $w = A(m, n - 1)$, when $m, n \geq 1$.

Suppose that $x$, $y$, $z$ are given and that we want an algorithm that outputs 1 if $A(x, y) = z$ and 0 otherwise. Note that by line (4) of Lemma 1, if the answer is $z$, that is if $A(x, y) = z$, we must have $x \leq z$ and $y \leq z$.

Thus, the following algorithm computes the Ackermann graph. “⋆” denotes a value greater than $z$; it can be ignored.
1. Input: $x, y, z$.

2. Output: 1 if $A(x, y) = z$, 0 if $A(x, y) \neq z$.

1. Compute $A(m, n)$ inside the rectangle $0 \leq m, n \leq z$.
   
   (a) Compute and save $A(0, 0), A(0, 1), \ldots, A(0, z)$.
   
   (b) Compute and save $A(1, 0), \ldots, A(z, 0)$.
   
   (c) For $m = 1, 2, \ldots, z$:
      
      Compute and save $A(m, 1), A(m, 2), \ldots, A(m, z)$.

   In computation above, if $A(m, n) > z$, mark the value of $A(m, n)$ as $\star$.

   Comment. Using this order of computation, whenever we compute $A(m, n)$ (using the definition 1), every value of $A$ that is needed for that computation (rules (2) or (3) of Definition 1) is already known. Thus, there are no “recursive calls”.

2. Find if $A(x, y) = z$
   
   (a) If $x > z$ or $y > z$ output 0 ($(x, y, z)$ not in the graph)
   
   (b) Otherwise search for a stored triple of the form $(x, y, w)$ (in particular we can have $w = \star$).
      
      i. If $w = z$ output 1 ($(x, y, z)$ in the graph).
      
      ii. If $w \neq z$ output 0 ($(x, y, z)$ not in the graph; this includes of course the case $w = \star$).

Comment. This algorithm can easily be made primitive recursive if we use a single integer $M$ (the “memory”) to code all the triples $(m, n, p)$ with $A(m, n) = p$ already computed. The total number of such triples is $(z + 1)^2$ and each occupies $O(\log(z))$ bits. The insertion of a triple in $M$ and the question “what is the computed value of $A(m, n)$?” can be implemented in a primitive recursive fashion. A value greater than $z$ ($\star$ above) can be coded by 0, as $A(m, n)$ is never 0.

Note. The proof above uses a bottom-up computation. If the inductive Definition 1 is directly used, it is possible to define a primitive recursive top-down algorithm.

3 The inverse of the Ackermann function is primitive recursive

As the Ackermann function is not onto, its inverse is not total.

Theorem 2. Suppose that $A(x, y)$ is an increasing function such that $A(x, y) \geq \max(x, y)$. By Lemma 1 this holds for the Ackermann function. Then, if the graph of $A(x, y)$ is primitive recursive, the inverse function
Given $z$, output:

- $(x, y)$ such that $A(x, y) = z$.
- NO if there is no such $(x, y)$.

is also primitive recursive.

The reason is clear: it is enough to compute the graph of the Ackermann function in the rectangle $0 \leq m, n \leq z$, which, as shown in Section 2, can be done in a primitive recursive way. Anyhow we present a proof below. **Proof.**

**Input:** $z$.
**Output:** $(x, y)$ such that $A(x, y) = z$ or NO if there is no such $(x, y)$.

Compute $(x, y, z)$ in the rectangle $0 \leq x, y \leq z$.

for $x = 0$ to $z$

for $y = 0$ to $z$

if $(x, y, z)$ is a computed triple,

output $(x, y)$ and STOP;

output NO.

**Notes.** For some other other forms of inverse, such as $A^{-1}(z) \overset{\text{def}}{=} \{(x, y) \mid A(x, y) \geq z \text{ and } x + y \text{ is as small as possible}\}$ (this is a set) this result (that is, Theorem 2 above) is also true.

The key for the success of the inversion algorithm is the fact that any possible inverse $(x, y)$ of $z$ satisfies $x \leq z$ and $y \leq z$. For any other primitive recursive function $g$ with $x \leq g(z)$ and $y \leq g(z)$, such as $g(z) = 2^{2^z}$, the existence of the inversion algorithm is also assured. □

Consider now the “diagonal” Ackermann function $A'(m) \overset{\text{def}}{=} A(m, m)$, which (I think) is not PR, but has a PR inverse. It seems that we may conclude that there are PR functions (as $A'^{-1}(m)$) whose inverse $(A'(m))$ is not PR... but not so fast! We have been dealing with “inverses” that are not the mathematical inverse of a function and more care is needed – in particular, $(A'^{-1})^{-1} \neq A'$.

**References**

