Ackermann's Function[*]

George Tourlakis

February 18, 2008

1 What

The Ackermann function was proposed, naturally, by Ackermann. The version here is a simplification offered by Robert Ritchie.

What the function does is to provide us with an example of a number-theoretic *intuitively computable*, <u>total</u> function that is not in $\mathcal{PR}$.

Another thing it does is it provides us with an example of a function $\lambda\vec{x}.f(\vec{x})$ that is "hard to compute" ($f \notin \mathcal{PR}$) but whose *graph* —that is, the predicate $\lambda y\vec{x}.y = f(\vec{x})$— is "easy to compute" ($\in \mathcal{PR}_*$).[1]

Definition 1.1 The Ackermann function, $\lambda nx.A_n(x)$, is given, for all $n \geq 0, x \geq 0$ by the equations

$$\begin{aligned} A_0(x) &= x + 2 \\ A_{n+1}(x) &= A_n^x(2) \end{aligned}$$

where h^x is function iteration (repeated composition of h with itself a variable number of times, x)

$$\underbrace{h \circ h \circ h \circ \cdots \circ h}_{x \text{ copies of } h}$$

More precisely, for all x, y,

$$\begin{aligned} h^0(y) &= y \\ h^{x+1}(y) &= h\Big(h^x(y)\Big) \end{aligned} \qquad \blacksquare$$

Remark 1.2 An alternative way to define the Ackermann function, extracted directly from Definition 1.1, is as follows:

$$A_0(x) = x + 2$$

[*]Lecture notes for CS4111 and CS5111; Winter 2006, 2008

[1]Here the colloquialisms "easy to compute" and "hard to compute" are aliases for "primitive recursive" and "not primitive recursive" respectively. This is a hopelessly coarse rendering of easy/hard and a much better gauge for the time complexity of a problem is on which side of $O(2^n)$ it lies. However, our gauge will have to do here: All I want to leave you with is that for some functions it is <u>easier</u> to compute the graph —to the quantifiable extent that it is in $\mathcal{PR}_*$— than the function itself —to the extent that it fails being primitive recursive.

$$\begin{aligned} A_{n+1}(0) &= 2 \\ A_{n+1}(x+1) &= A_n(A_{n+1}(x)) \end{aligned} \qquad \blacksquare$$

2 Properties of the Ackermann function

We have a sequence of less-than-earth shattering —but useful— theorems. So we will just call them lemmata.

Lemma 2.1 For each $n \geq 0$, $\lambda x.A_n(x) \in \mathcal{PR}$.

Proof Induction on n:

For the basis, clearly $A_0 = \lambda x.x + 2 \in \mathcal{PR}$.

Assume the case for (arbitrary, fixed) n —i.e., $A_n \in \mathcal{PR}$— and go to $n + 1$. Immediate from 1.2, last two equations. $\blacksquare$

It turns out that the function blows up far too fast with respect to the argument n. We now quantify this remark.

The following unambitious sounding lemma is the key to proving the growth properties of the Ackermann function. It is also the least straightforward to prove, as it requires a double induction —at once on n and x— as dictated by the fact that the "recursion" of definition 1.2 does not leave any argument fixed.

Lemma 2.2 For each $n \geq 0$ and $x \geq 0$, $A_n(x) > x + 1$.

Proof We start an induction on n:

$\underline{n\text{-Basis.}}$ $n = 0$: $A_0(x) = x + 2 > x + 1$; check.

$\underline{n\text{-I.H.}}$[2] $\underline{\text{For all } x}$ and a fixed (but unspecified) n, assume $A_n(x) > x + 1$.

$\underline{n\text{-I.S.}}$[3] For all x and the above fixed (but unspecified) n, $\underline{\text{prove}}$ $A_{n+1}(x) > x + 1$.

We do the n-I.S. by induction on x:

$\underline{x\text{-Basis.}}$ $x = 0$: $A_{n+1}(0) = 2 > 1$; check.

$\underline{x\text{-I.H.}}$ For the above fixed n, fix an x (but leave it unspecified) for which assume $A_{n+1}(x) > x + 1$.

$\underline{x\text{-I.S.}}$ For the above fixed (but unspecified) n and x, $\underline{\text{prove}}$ $A_{n+1}(x + 1) > x + 2$.

Well,

$$\begin{aligned} A_{n+1}(x+1) &= A_n(A_{n+1}(x)) \quad \text{by 1.2} \\ &> A_{n+1}(x) + 1 \quad \text{by } n\text{-I.H.} \\ &> x + 2 \quad \text{by } x\text{-I.H.} \end{aligned}$$

$\blacksquare$

[2] I.H. is an acronym for Induction Hypothesis. Formally, what we are proving is "$(\forall n)(\forall x)A_n(x) > x + 1$". Thus, as we start on an induction on n, its I.H. is "$(\forall x)A_n(x) > x + 1$" for a fixed unspecified n.

[3] I.S. is an acronym for Induction Step. Formally, the step is to prove —from the Basis and I.H.— "$(\forall x)A_{n+1}(x) > x + 1$" for the n that we fixed in the I.H. It turns out that this is best handled by induction on x.

Lemma 2.3 $\lambda x.A_n(x) \nearrow$.

NOTE. "$\lambda x.f(x) \nearrow$" means that the (total) function f is *strictly increasing*, that is, $x < y$ implies $f(x) < f(y)$, for any x and y. Clearly, to establish the property one just needs to check for the arbitrary x that $f(x) < f(x+1)$.

Proof We handle two cases separately.
$\underline{A_0}$: $\lambda x.x + 2 \nearrow$.
$\underline{A_{n+1}}$: $A_{n+1}(x+1) = A_n(A_{n+1}(x)) > A_{n+1}(x) + 1$ —the "$>$" by Lemma 2.2. ■

Lemma 2.4 $\lambda n.A_n(x+1) \nearrow$.

Proof $A_{n+1}(x+1) = A_n(A_{n+1}(x)) > A_n(x+1)$ —the "$>$" by Lemmata 2.2 (left argument $>$ right argument) and 2.3. ■

NOTE. The "$x + 1$" in 2.4 is important since $A_n(0) = 2$ for all n. Thus $\lambda n.A_n(0)$ is increasing but *not* strictly (constant).

Lemma 2.5 $\lambda y.A^y_n(x) \nearrow$.

Proof $A^{y+1}_n(x) = A_n(A^y_n(x)) > A^y_n(x)$ —the "$>$" by Lemma 2.2. ■

Lemma 2.6 $\lambda x.A^y_n(x) \nearrow$.

Proof Induction on y:
For $y = 0$ we want that $\lambda x.A^0_n(x) \nearrow$, that is, $\lambda x.x \nearrow$, which is true.
We take as I.H. that

$$A^y_n(x+1) > A^y_n(x) \tag{1}$$

We want

$$A^{y+1}_n(x+1) > A^{y+1}_n(x) \tag{2}$$

But (2) follows from (1) by applying A_n to both sides of "$>$" and invoking Lemma 2.3. ■

Lemma 2.7 For all n, x, y, $A^y_{n+1}(x) \geq A^y_n(x)$.

Proof Induction on y:
For $y = 0$ we want that $A^0_{n+1}(x) \geq A^0_n(x)$, that is, $x \geq x$.
We take as I.H. that

$$A^y_{n+1}(x) \geq A^y_n(x)$$

We want

$$A^{y+1}_{n+1}(x) \geq A^{y+1}_n(x)$$

This is true because

$$A^{y+1}_{n+1}(x) = A_{n+1}\Big(A^y_{n+1}(x)\Big) \overset{\text{if } y = x = 0 \text{ we have "="; else by 2.4}}{\geq} A_n\Big(A^y_{n+1}(x)\Big) \overset{\text{2.3 and I.H.}}{\geq} A^{y+1}_n(x)$$

■

Definition 2.8 For a predicate $P(\vec{x})$ we say that $\underline{P(\vec{x}) \text{ is true } \textit{almost everywhere}}$ —in symbols "$P(\vec{x})$ a.e."– iff the set of (vector) inputs that make the predicate false is finite. That is, the set $\{\vec{x} : \neg P(\vec{x})\}$ is finite.

A statement such as "$\lambda xy.Q(x, y, z, w)$ a.e." can also be stated, less formally, as "$Q(x, y, z, w)$ a.e. $\underline{\text{with respect to } x \text{ and } y}$". ■

Lemma 2.9 $A_{n+1}(x) > x + l$ a.e. with respect to x.

NOTE. Thus, in particular, $A_1(x) > x + 10^{350000}$ a.e.

Proof In view of Lemma 2.4 and the note following it, it suffices to prove

$$A_1(x) > x + l \text{ a.e. with respect to } x$$

Well, since

$$A_1(x) = A_0^x(2) = \overbrace{(\cdots(((y+2)+2)+2)+\cdots+2)}^{x\text{ 2's}}\,\|_{\text{evaluated at } y\,=\,2} = 2 + 2x$$

we ask: Is $2 + 2x > x + l$ a.e. with respect to x? You bet. It is so for all $x > l - 2$ (only $x = 0, 1, \ldots, l - 2$ fail). ■

Lemma 2.10 $A_{n+1}(x) > A_n^l(x)$ a.e. with respect to x.

Proof If one (or both) of l or n is 0, then the result is trivial. For example,

$$A_0^l(x) = \overbrace{(\cdots(((x+2)+2)+2)+\cdots+2)}^{l\text{ 2's}} = x + 2l$$

In the preceding proof we saw that $A_1(x) = 2x + 2$. Clearly, $2x + 2 > x + 2l$ as soon as $x > 2l - 2$, that is, a.e with respect to x.

Let us then assume $l \geq 1, n \geq 1$. We note that (straightforwardly, via Definition 1.1)

$$A_n^l(x) = A_n(A_n^{l-1}(x)) = A_{n-1}^{A_n^{l-1}(x)}(2) = A_{n-1}^{A_{n-1}^{A_n^{l-2}(x)}(2)}(2) = A_{n-1}^{A_{n-1}^{A_{n-1}^{A_n^{l-3}(x)}(2)}(2)}(2)$$

The straightforward observation that we have a "ladder" of k A_{n-1}'s precisely when the top-most exponent is $l - k$ can be ratified by induction on k (not done here). Thus I state

$$A_n^l(x) = {}^{k\ A_{n-1}}\left\{ A_{n-1}^{\cdot^{\cdot^{\cdot^{A_{n-1}^{A_n^{l-k}(x)}(2)}}}} \right. \ddots(2)$$

In particular, taking $k = l$,

$$A_n^l(x) = {}^{l\ A_{n-1}}\left\{ A_{n-1}^{\cdot^{\cdot^{\cdot^{A_{n-1}^{A_n^{l-l}(x)}(2)}}}} \right. \ddots(2) = {}^{l\ A_{n-1}}\left\{ A_{n-1}^{\cdot^{\cdot^{\cdot^{A_{n-1}^{x}(2)}}}} \right. \ddots(2) \qquad (*)$$

Let us now take $x > l$.

Thus, by $(*)$,

$$A_{n+1}(x) = A_n^x(2) = {}^{x\ A_{n-1}}\left\{ A_{n-1}^{\cdot^{\cdot^{\cdot^{A_{n-1}^{2}(2)}}}} \right. \ddots(2) \qquad (**)$$

By comparing $(*)$ and $(**)$ we see that the first "ladder" is topped (after l A_{n-1} "steps") by x and the second is topped by

$$ {}_{x-l\ A_{n-1}}\left\{\begin{matrix} & & A_{n-1}^{2}(2) \\ & \cdot^{\cdot^{\cdot}} & \\ A_{n-1} & & \end{matrix}\right. \ddots (2) $$

Thus —in view of the fact that $A_n^y(x)$ increases with respect to each of the arguments n, x, y— we conclude by answering:

$$ \text{“Is}\quad {}_{x-l\ A_{n-1}}\left\{\begin{matrix} & & A_{n-1}^{2}(2) \\ & \cdot^{\cdot^{\cdot}} & \\ A_{n-1} & & \end{matrix}\right. \ddots (2) > x \text{ a.e. with respect to } x\text{?”} $$

Yes, because by $(**)$ this is the same question as "is $A_{n+1}(x-l) > x$ a.e. with respect to x?" which has been answered in Lemma 2.9. ■

Lemma 2.11 For all n, x, y, $A_{n+1}(x+y) > A_n^x(y)$.

Proof

$$\begin{aligned} A_{n+1}(x+y) &= A_n^{x+y}(2) \\ &= A_n^x\Big(A_n^y(2)\Big) \\ &= A_n^x\Big(A_{n+1}(y)\Big) \\ &> A_n^x(y) \quad \text{by Lemmata 2.2 and 2.6} \end{aligned}$$

■

3 The Ackermann function majorises all the functions of $\mathcal{PR}$

We say that a function f majorises another one, g, iff $g(\vec{x}) \le f(\vec{x})$ for all $\vec{x}$. The following theorem states precisely in what sense "the Ackermann function majorises all the functions of $\mathcal{PR}$".

Theorem 3.1 For every function $\lambda\vec{x}.f(\vec{x}) \in \mathcal{PR}$ there are numbers n and k, such that for all $\vec{x}$ we have $f(\vec{x}) \le A_n^k(\max(\vec{x}))$.

Proof The proof is by induction on $\mathcal{PR}$. Throughout I use the abbreviation $|\vec{x}|$ for $\max(\vec{x})$ as it is notationally friendlier.

For the basis, f is one of:

- *Basis.*

  Basis 1. $\lambda x.0$. Then $A_0(x)$ works $(n=0, k=1)$.

  Basis 2. $\lambda x.x+1$. Again $A_0(x)$ works $(n=0, k=1)$.

  Basis 3. $\lambda\vec{x}.x_i$. Once more $A_0(x)$ works $(n=0, k=1)$: $x_i \le |\vec{x}| < A_0(|\vec{x}|)$.

- *Propagation with composition.* Assume as I.H. that

$$f(\vec{x}_m) \leq A_n^k(|\vec{x}_m|) \tag{1}$$

and

$$\text{for } i = 1, \dots, m,\ g_i(\vec{y}) \leq A_{n_i}^{k_i}(|\vec{y}|) \tag{2}$$

Then

$$\begin{aligned} f(g_1(\vec{y}), \dots, g_m(\vec{y})) &\leq A_n^k(|g_1(\vec{y}), \dots, g_m(\vec{y})|), \text{ by (1)} \\ &\leq A_n^k(|A_{n_1}^{k_1}(|\vec{y}|), \dots, A_{n_m}^{k_m}(|\vec{y}|)|), \text{ by 2.6 and (2)} \\ &\leq A_n^k\left(|A_{\max n_i}^{\max k_i}(|\vec{y}|)\right), \text{ by 2.6 and 2.7} \\ &\leq A_{\max(n,n_i)}^{k+\max k_i}(|\vec{y}|), \text{ by 2.7} \end{aligned}$$

- *Propagation with primitive recursion.* Assume as I.H. that

$$h(\vec{y}) \leq A_n^k(|\vec{y}|) \tag{3}$$

and

$$g(x, \vec{y}, z) \leq A_m^r(|\vec{x}, y, z|) \tag{4}$$

Let f be such that

$$\begin{aligned} f(0, \vec{y}) &= h(\vec{y}) \\ f(x+1, \vec{y}) &= g(x, \vec{y}, f(x, \vec{y})) \end{aligned}$$

I claim that

$$f(x, \vec{y}) \leq A_m^{rx}\left(A_n^k(|x, \vec{y}|)\right) \tag{5}$$

I prove (5) by induction on x:

For $x = 0$, I want $f(0, \vec{y}) = h(\vec{y}) \leq A_n^k(|0, \vec{y}|)$. This is true by (3) since $|0, \vec{y}| = |\vec{y}|$.

As an I.H. assume (5) for fixed x.

The case for $x + 1$:

$$\begin{aligned} f(x+1, \vec{y}) &= g(x, \vec{y}, f(x, \vec{y})) \\ &\leq A_m^r(|\vec{x}, y, f(x, \vec{y})|), \text{ by (4)} \\ &\leq A_m^r\left(\left|\vec{x}, y, A_m^{rx}\left(A_n^k(|x, \vec{y}|)\right)\right|\right), \text{ by I.H.(5) and 2.6} \\ &= A_m^r\left(A_m^{rx}\left(A_n^k(|x, \vec{y}|)\right)\right), \text{ by } |\vec{w}| \geq w_i \text{ and 2.6} \\ &= A_m^{r(x+1)}\left(A_n^k(|x, \vec{y}|)\right) \end{aligned}$$

With (5) proved, let me set $l = \max(m, n)$. By Lemma 2.7 I now get

$$f(x, \vec{y}) \leq A_l^{rx+k}(|x, \vec{y}|) \underset{\text{Lemma 2.11}}{<} A_{l+1}(|x, \vec{y}| + rx + k) \tag{6}$$

Now, $|x, \vec{y}| + rx + k \leq (r+1)|x, \vec{y}| + k$ thus, (6) and 2.3 yield

$$f(x, \vec{y}) < A_{l+1}((r+1)|x, \vec{y}| + k) \tag{7}$$

To simplify (7) note that there is a number q such that

$$(r+1)x + k \leq A_1^q(x) \tag{8}$$

for all x. Indeed, this is so since (easy induction on y) $A_1^y(x) = 2^y x + 2^y + 2^{y-1} + \cdots + 2$. Thus, to satisfy (8), just take $y = q$ large enough to satisfy $r+1 \leq 2^q$ and $k \leq 2^q + 2^{q-1} + \cdots + 2$.

By (8), (7) and 2.3 yield

$$f(x, \vec{y}) < A_{l+1}(A_1^q(|x, \vec{y}|)) \leq A_{l+1}^{1+q}(|x, \vec{y}|)$$

(by Lemma 2.7) which is all we want. ■

NB. Reading the proof carefully we note that the subscript argument of the *majorant*[4] is precisely the depth of nesting of primitive recursion. Indeed, the initial functions have a majorant with subscript 0; composition has a majorant with subscript no more than the maximum subscript of the component parts —no increase; primitive recursion has a majorant with a subscript that is bigger than the maximum subscript of the h and g-majorants by precisely 1.

Corollary 3.2 $\lambda nx.A_n(x) \notin \mathcal{PR}$.

Proof By contradiction: If $\lambda nx.A_n(x) \in \mathcal{PR}$ then also $\lambda x.A_x(x) \in \mathcal{PR}$. By the theorem above, for some n, k, $A_x(x) \leq A_n^k(x)$, for all x, hence, by 2.10

$$A_x(x) < A_{n+1}(x), \text{ a.e. with respect to } x \tag{1}$$

On the other hand, $A_{n+1}(x) < A_x(x)$ a.e. with respect to x —indeed for all $x > n+1$ by 2.4— which contradicts (1). ■

4 The Graph of the Ackermann function is in $\mathcal{PR}_*$

How does one compute a yes/no answer to the question

$$\text{“}A_n(x) = z\text{?”} \tag{1}$$

Thinking “recursively” (in the programming sense of the word), we will look at the question by considering three cases, according to the definition in the Remark 1.2:

(a) If $n = 0$, then we will directly check (1) as “is $x + 2 = z$?”.

(b) If $x = 0$, then we will directly check (1) as “is $2 = z$?”.

[4]The function that does the majorising

(c) In all other cases, i.e., $n > 0$ and $x > 0$, for an appropriate w, we may naturally[5] ask *two* questions (both must be answerable "yes" for (1) to be true): "Is $A_{n-1}(w) = z$?", and "is $A_n(x-1) = w$?"

Assuming that we want to pursue this by pencil and paper or some other equivalent means, it is clear that the pertinent info that we are juggling are ordered triples of numbers such as n, x, z, or $n-1, w, z$, etc. That is, the letter "A", the brackets, the equals sign, and the position of the arguments (subscript vs. inside brackets) are just ornamentation, and the string "$A_i(j) = k$", *in this section's context*, does not contain any more information than the ordered triple "i, j, k".

Thus, to "compute" an answer to (1) we need to write down enough triples, in stages (or steps), as needed to justify (1): At each stage we may write a triple i, j, k down just in case one of (i)–(iii) holds:

(i) $i = 0$ and $k = j + 2$

(ii) $j = 0$ and $k = 2$

(iii) $i > 0$ and $j > 0$, and for some w, we have already written down the two triples $i-1, w, k$ and $i, j-1, w$.

Pause. Since "i, j, k" abbreviates "$A_i(j) = k$", Lemma 2.2 implies that $w < k$.

Our theory is more competent with numbers (than with pairs, triples, etc., preferring to *code* such tuples[6] into single numbers), thus if we were to carry out the pencil and paper algorithm *within our theory*, then we would be well advised to code all these triples, which we write down step by step, by single numbers: E.g., $\langle i, j, k \rangle$.

We note that our computation is tree-like, since a "complicated" triple such as that of case (iii) above *requires* two similar others to be already written down, each of which in turn will require two *earlier* similar others, etc, until we reach "leaves" (cases (i) or (ii)) that can be dealt directly without passing the buck.

This "tree", just like the tree of a mathematical proof,[7] can be arranged in a sequence, of triples $\langle i, j, k \rangle$, so that the presence of a "$\langle i, j, k \rangle$" implies that all its dependencies appear *earlier* (to its left).

We will code such a sequence by a single number, u, using the prime-power coding of sequences given in class:

$$\langle a_0, \ldots, a_{z-1} \rangle = \Pi_{i<z} p_i^{a_i+1}$$

Given any number u we can primitively recursively check whether it is a code of an Ackermann function computation or not:

Theorem 4.1 The predicate

$$Comp(u) \overset{\text{def}}{=} u \text{ codes an Ackermann function computation}$$

is in $\mathcal{PR}_*$.

[5] $A_n(x) = A_{n-1}(A_n(x-1))$.

[6] As in quin*tuples*, n-tuples. This word has found its way in the theoretician's dictionary, if not in general purpose dictionaries.

[7] Assuming that modus ponens is the only rule of inference, the proof a formula A depends, in general, on that of "earlier" formulae $X \to A$ and X, which in turn depend (require) earlier formulae each, and so on and so on, until we reach formulae that are axioms.

Proof We will use some notation that will be useful to make the proof more intuitive (this notation also appears in the Kleene Normal Form notes posted). Thus we introduce two predicates: $\lambda vu.v \in u$ and $\lambda vwu.v <_u w$. The first says

$$u = \langle \ldots, v, \ldots \rangle$$

and the second says

$$u = \langle \ldots, v, \ldots, w, \ldots \rangle$$

Both are in $\mathcal{PR}_*$ since

$$v \in u \equiv Seq(u) \wedge (\exists i)_{<lh(u)} (u)_i = v$$

and

$$v <_u w \equiv Seq(u) \wedge (\exists i)_{<lh(u)} (\exists j)_{<lh(u)} \big((u)_i = v \wedge (u)_j = w \wedge i < j\big)$$

We can now define $Comp(u)$ by a formula that makes it clear that it is in $\mathcal{PR}_*$:

$$Comp(u) \equiv Seq(u) \wedge (\forall v)_{\leq u} \Big[v \in u \rightarrow Seq(v) \wedge lh(v) = 3 \wedge \Big\{$$

$\{$**Comment:** Case (i), p.8$\}$ $(v)_0 = 0 \wedge (v)_2 = (v)_1 + 2 \vee$

$\{$**Comment:** Case (ii)$\}$ $(v)_1 = 0 \wedge (v)_2 = 2 \vee$

$\{$**Comment:** Case (iii)$\}$ $\Big((v)_0 > 0 \wedge (v)_1 > 0 \wedge$

$$(\exists w)_{<v} (\langle (v)_0 \dot{-} 1, w, (v)_2 \rangle <_u v \wedge \langle (v)_0, (v)_1 \dot{-} 1, w \rangle <_u v)\Big)\Big\}\Big]$$

The "Pause" on p.8 justifies the bound on $(\exists w)$ above. Indeed, we could have used the tighter bound "$(v)_2$". Clearly $Comp(u) \in \mathcal{PR}_*$. ■

Thus $A_n(x) = z$ iff $\langle n, x, z \rangle \in u$ for some u that satisfies $Comp$, for short

$$A_n(x) = z \equiv (\exists u)(Comp(u) \wedge \langle n, x, z \rangle \in u) \tag{1}$$

If we succeed to find a bound for u that is a primitive recursive function of n, x, z then we will have succeeded showing:

Theorem 4.2 $\lambda nxz.A_n(x) = z \in \mathcal{PR}_*$.

Proof Let us focus on a computation u that as soon as it verifies $A_n(x) = z$ quits, that is, it only codes $\langle n, x, z \rangle$ and just the needed predecessor triples, no more. How big can such a u be?

Well,

$$u = \cdots p_r^{\langle i,j,k \rangle + 1} \cdots p_l^{\langle n,x,z \rangle + 1} \tag{2}$$

for appropriate l (=$lh(u) - 1$). For example, if all we want is to verify $A_0(3) = 5$, then $u = p_0^{\langle 0,3,5 \rangle + 1}$.

Similarly, if all we want to verify is $A_1(1) = 4$, then —since the "recursive calls" here are to $A_0(2) = 4$ and $A_1(0) = 2$— two possible u-values work: $u = p_0^{\langle 0,2,4 \rangle + 1} p_1^{\langle 1,0,2 \rangle + 1} p_2^{\langle 1,1,4 \rangle + 1}$ or $u = p_0^{\langle 1,0,2 \rangle + 1} p_1^{\langle 0,2,4 \rangle + 1} p_2^{\langle 1,1,4 \rangle + 1}$.

How big need l be? No bigger than needed to provide distinct *positions* ($l + 1$ such) in the computation, for all the "needed" triples i, j, k. Since z is the largest possible output (and larger than any input) computed, there are no more than $(z+1)^3$ triples possible, so $l+1 \leq (z+1)^3$. Therefore, (2) yields

$$\begin{aligned} u &\leq \cdots p_r^{\langle z,z,z\rangle+1} \cdots p_l^{\langle z,z,z\rangle+1} \\ &= \left(\Pi_{i\leq l} p_i\right)^{\langle z,z,z\rangle+1} \\ &\leq p_l^{(l+1)(\langle z,z,z\rangle+1)} \\ &\leq p_{(z+1)^3}^{((z+1)^3+1)(\langle z,z,z\rangle+1)} \end{aligned}$$

Setting $g = \lambda z.p_{(z+1)^3}^{((z+1)^3+1)(\langle z,z,z\rangle+1)}$ we have $g \in \mathcal{PR}$ and we are done by (1):

$$A_n(x) = z \equiv (\exists u)_{\leq g(z)}(Comp(u) \wedge \langle n,x,z\rangle \in u) \qquad \blacksquare$$