THE ACKERMANN-FUNCTION
EFFORT IN SPACE AND TIME

ROLAND LIEGER AND JOHANN BLIEBERGER

DEPARTMENT OF AUTOMATION (183/1)
TECHNICAL UNIVERSITY VIENNA
TREITLSTR. 3/4
A-1040 VIENNA
AUSTRIA
EMAIL: RLIEGER@AUTO.TUWIEN.AC.AT
EMAIL: BLIEB@AUTO.TUWIEN.AC.AT

ABSTRACT. The Ackermann-Function is a well-known function in computer science. The first section of this paper is devoted to some useful properties. Next we take a look at the effort needed to compute a value, in terms of computing time as well as in stack space. In the final section we show how the results obtained in [BL94] can be applied to facilitate or even automate the analysis.

1. The Ackermann-Function $\mathcal{A}(x, y)$

The Ackermann-Function is quite a weird thing. At first sight it looks simple and it definitely is easy to implement, but it is quite difficult to understand, what this function really does. Let's look at its definition:

$$\begin{aligned} \mathcal{A}(0, y) &= y + 1 \\ \mathcal{A}(x+1, 0) &= \mathcal{A}(x, 1) \\ \mathcal{A}(x+1, y+1) &= \mathcal{A}(x, \mathcal{A}(x+1, y)) \\ x, y &\in \mathbb{N}_0 \end{aligned}$$

Using this simple, recursive formula, it is easy to produce a small table of $\mathcal{A}(x, y)$ for a few, small values of x and y.

		y								
	$\mathcal{A}(x,y)$	0	1	2	3	4	5	6	7	8
	0	1	2	3	4	5	6	7	8	9
	1	2	3	4	5	6	7	8	9	10
x	2	3	5	7	9	11	13	15	17	19
	3	5	13	29	61	125	253	509	1021	2045
	4	13	65533	...						
	5	65533	...							

Supported by the Austrian Science Foundation (FWF) under grant P10188-MAT.

Obviously this table is not complete, but the values that have been omitted are all a good deal bigger than can be expressed by a 32-bit integer.

2. Solving $\mathcal{A}(x,y)$ for Special Values of x

Looking at the first lines of the table for $\mathcal{A}(x,y)$, a law is quite obvious. With a little more effort, it is even possible to find laws for the harder case $x=3$. There exists no closed form for $x \geq 4$.

$$\begin{aligned}
\mathcal{A}(0,y) &= y+1 \\
\mathcal{A}(1,y) &= y+2 \\
\mathcal{A}(2,y) &= 2y+3 \\
\mathcal{A}(3,y) &= 2^{y+3}-3 \\
\mathcal{A}(4,y) &= 2^{\mathcal{A}(4,y-1)+3}-3 \text{ for all } y \geq 1
\end{aligned}$$

Proof.

$\mathcal{A}(0,y)$: $\mathcal{A}(0,y) = y+1$ by definition of $\mathcal{A}(x,y)$.

$$\begin{aligned}
\mathcal{A}(1,y)\text{:}\quad & y=0\text{:} & \mathcal{A}(1,0) &= \mathcal{A}(0,1) = 2 = y+2 \\
& y \geq 1\text{:} & \mathcal{A}(1,y) &= \mathcal{A}(0,\mathcal{A}(1,y-1)) = \mathcal{A}(0,(y-1)+2) \\
& & &= \mathcal{A}(0,y+1) = (y+1)+1 = y+2
\end{aligned}$$

$$\begin{aligned}
\mathcal{A}(2,y)\text{:}\quad & y=0\text{:} & \mathcal{A}(2,0) &= \mathcal{A}(1,1) = 3 = 2y+3 \\
& y \geq 1\text{:} & \mathcal{A}(2,y) &= \mathcal{A}(1,\mathcal{A}(2,y-1)) = \mathcal{A}(1,2(y-1)+3) \\
& & &= \mathcal{A}(1,2y+1) = (2y+1)+2 = 2y+3
\end{aligned}$$

$$\begin{aligned}
\mathcal{A}(3,y)\text{:}\quad & y=0\text{:} & \mathcal{A}(3,0) &= \mathcal{A}(2,1) = 5 = 2^{y+3}-3 \\
& y \geq 1\text{:} & \mathcal{A}(3,y) &= \mathcal{A}(2,\mathcal{A}(3,y-1)) = \mathcal{A}(2,2^{(y-1)+3}-3) \\
& & &= \mathcal{A}(2,2^{y+2}-3) = 2(2^{y+2}-3)+3 = 2^{y+3}-3
\end{aligned}$$

$$\begin{aligned}
\mathcal{A}(4,y)\text{:}\quad & y=0\text{:} & \mathcal{A}(4,0) &= \mathcal{A}(3,1) = 13 \\
& y \geq 1\text{:} & \mathcal{A}(4,y) &= \mathcal{A}(3,\mathcal{A}(4,y-1)) = 2^{\mathcal{A}(4,y-1)+3}-3 \quad \square
\end{aligned}$$

3. Some Properties of $\mathcal{A}(x,y)$

Property 1. $\mathcal{A}(x,y) > (x+y)$

Proof.

$$\begin{aligned}
x=0\text{:}\quad & \mathcal{A}(0,y) = y+1 > 0+y = x+y \\
x \geq 1\text{:}\quad & y=0\text{: } \mathcal{A}(x,0) = \mathcal{A}(x-1,1) > (x-1)+1 = x+0 = x+y \\
& y \geq 1\text{: } \mathcal{A}(x,y) = \mathcal{A}(x-1,\mathcal{A}(x,y-1)) > (x-1)+\mathcal{A}(x,y-1) \geq \\
& \qquad \geq (x-1)+x+(y-1)+1 = (x-1)+x+y \geq x+y \quad \square
\end{aligned}$$

Remark 3.1. $\mathcal{A}(x,y) > (x+y)$ implies $\mathcal{A}(x,y) \geq (x+y)+1$

Collorary 1. $\mathcal{A}(x,y) > x$ and $\mathcal{A}(x,y) > y$.

Property 2. $\mathcal{A}(x,y) < \mathcal{A}(x,y+1)$

Proof.

$x = 0$: $\mathcal{A}(0,y) = y+1 < (y+1)+1 = \mathcal{A}(0,y+1)$
$x \geq 1$: $\mathcal{A}(x,y) \leq (x-1) + \mathcal{A}(x,y) < \mathcal{A}(x-1,\mathcal{A}(x,y)) = \mathcal{A}(x,y+1)$ □

Collorary 2. $\mathcal{A}(x,y) < \mathcal{A}(x,y+n)$ for all $n \geq 1$.

Property 3. $\mathcal{A}(x,y+1) \leq \mathcal{A}(x+1,y)$

Proof.

$x = 0$: $\mathcal{A}(0,y+1) = (y+1)+1 = y+2 = \mathcal{A}(1,y)$
$x \geq 1$: $y = 0$: $\mathcal{A}(x,0+1) = \mathcal{A}(x+1,0)$ (by definition of the Ackermann-Function)
$y \geq 1$: $\mathcal{A}(x,y+1) \leq \mathcal{A}(x,(x+1)+(y-1)) <$
$< \mathcal{A}(x,\mathcal{A}(x+1,y-1)) = \mathcal{A}(x+1,y)$ □

Property 4. $\mathcal{A}(x,y) < \mathcal{A}(x+1,y)$

Proof. Using $\mathcal{A}(x,y) < \mathcal{A}(x,y+1)$ and $\mathcal{A}(x,y+1) \leq \mathcal{A}(x+1,y)$ it is obvious that $\mathcal{A}(x,y) < \mathcal{A}(x+1,y)$. □

Collorary 3. $\mathcal{A}(x,y) < \mathcal{A}(x+m,y)$ for all $m \geq 1$.

Collorary 4. $\mathcal{A}(x,y) < \mathcal{A}(x+m,y+n)$ for all $m,n \geq 0$, $m+n \geq 1$.

4. Execution Time of $\mathcal{A}(x,y)$

In this section we want to determine the execution time of $\mathcal{A}(x,y)$ for various values of x and y. To get rid of all language, compiler and machine dependencies we will measure the execution time in terms of (recursive) calls to $\mathcal{A}(x,y)$, rather than in seconds. Further we will assume a straight-forward implementation of $\mathcal{A}(x,y)$. Thus we forbid the caching of previously computed values, or the use of closed forms for $\mathcal{A}(x,y)$ for some $x > 0$, even so we are aware that either measure would significantly speed up computation. Practical experiments quickly produce a small table:

	$\mathcal{T}(x,y)$	y: 0	1	2	3	4	5	6	7	8
	0	1	1	1	1	1	1	1	1	1
	1	2	4	6	8	10	12	14	16	18
x	2	5	14	27	44	65	90	119	152	189
	3	15	106	541	2432	10307	42438	...		
	4	107	...							

Of course it is also possible to find these values by pure thinking. Let's start with the following set of recursive equations for $\mathcal{T}(x,y)$:

$$\begin{aligned}
\mathcal{T}(0,y) &= 1 \\
\mathcal{T}(x+1,0) &= 1 + \mathcal{T}(x,1) \\
\mathcal{T}(x+1,y+1) &= 1 + \mathcal{T}(x+1,y) + \mathcal{T}(x,\mathcal{A}(x+1,y))
\end{aligned}$$

Starting there, we compute special equations for specific values of x:

$$
\begin{aligned}
\mathcal{T}(0,y) &= 1\\
\mathcal{T}(1,y) &= 2(y+1)\\
\mathcal{T}(2,y) &= 2y^2+7y+5 = (y+1)(2y+5)\\
\mathcal{T}(3,y) &= \frac{128}{3}2^{2y} - 40\cdot 2^y + 3y + \frac{37}{3}\\
&= \frac{32}{3}(2^{2(y+1)}-1) - 20(2^{y+1}-1) + 3(y+1)
\end{aligned}
$$

Proof.

$\mathcal{T}(0,y)$: $\mathcal{T}(0,y) = 1$ by definition of the recursion for $\mathcal{T}$

$$
\begin{aligned}
\mathcal{T}(1,y)\text{:}\quad & y=0\text{:}\quad \mathcal{T}(1,0) = 1+\mathcal{T}(0,1) = 1+1 = 2 = 2(0+1) = 2(y+1)\\
& y\geq 1\text{:}\quad \mathcal{T}(1,y) = 1+\mathcal{T}(1,y-1)+\mathcal{T}(0,\mathcal{A}(1,y-1))\\
&\qquad = 1+2((y-1)+1)+\mathcal{T}(0,(y-1)+2)\\
&\qquad = 1+2y+1 = 2(y+1)
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{T}(2,y)\text{:}\quad & y=0\text{:}\quad \mathcal{T}(2,0) = 1+\mathcal{T}(1,1) = 1+2(1+1) = 5 = 2y^2+7y+5\\
& y\geq 1\text{:}\quad \mathcal{T}(2,y) = 1+\mathcal{T}(2,y-1)+\mathcal{T}(1,\mathcal{A}(2,y-1)\\
&\qquad = 1+2(y-1)^2+7(y-1)+5+\mathcal{T}(1,2(y-1)+3)\\
&\qquad = 1+2y^2-4y+2+7y-7+5+\mathcal{T}(1,2y+1)\\
&\qquad = 2y^2+3y+1+2((2y+1)+1)\\
&\qquad = 2y^2+3y+1+4y+4 = 2y^2+7y+5
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{T}(3,y)\text{:}\quad & y=0\text{:}\quad \mathcal{T}(3,0) = 1+\mathcal{T}(2,1) = 1+(2\cdot 1^2+7\cdot 1+5) = 15\\
&\qquad = \tfrac{128}{3} - 40 + 0 + \tfrac{37}{3} = \tfrac{128}{3}2^{2y} - 40\cdot 2^y + 3y + \tfrac{37}{3}\\
& y\geq 1\text{:}\quad \mathcal{T}(3,y) = 1+\mathcal{T}(3,y-1)+\mathcal{T}(2,\mathcal{A}(3,y-1))\\
&\qquad \mathcal{T}(3,y-1) = \tfrac{128}{3}2^{2(y-1)} - 40\cdot 2^{y-1} + 3(y-1) + \tfrac{37}{3}\\
&\qquad\qquad = \tfrac{128}{3}2^{-2}\cdot 2^{2y} - 40\cdot 2{-}1\cdot 2^{y-1} + 3y - 3 + \tfrac{37}{3}\\
&\qquad\qquad = \tfrac{32}{3}2^{2y} - 20\cdot 2^y + 3y + \tfrac{28}{3}\\
&\qquad \mathcal{T}(2,\mathcal{A}(3,y-1)) = \mathcal{T}(2,2^{(y-1)+3}-3) = \mathcal{T}(2,2^{y+2}-3)\\
&\qquad\qquad = 2(2^{y+2}-3)^2 + 7(2^{y+2}-3) + 5\\
&\qquad\qquad = 2(2^{2(y+2)} - 2\cdot 3\cdot 2^{y+2} + 9) + 7(2^{y+2}-3) + 5\\
&\qquad\qquad = 32\cdot 2^{2y} - 48\cdot 2^y + 18 + 28\cdot 2^y - 21 + 5\\
&\qquad\qquad = 32\cdot 2^{2y} - 20\cdot 2^y + 2\\
&\quad \mathcal{T}(3,y) = 1 + \tfrac{32}{3}2^{2y} - 20\cdot 2^y + 3y + \tfrac{28}{3} + 32\cdot 2^{2y} - 20\cdot 2^y + 2\\
&\qquad = \tfrac{32}{3}(4\cdot 2^{2y}) - 40\cdot 2^y + 3y + \tfrac{28}{3} + 3\\
&\qquad = \tfrac{128}{3}2^{2y} - 40\cdot 2^y + 3y + \tfrac{37}{3} \quad \square
\end{aligned}
$$

5. Maximal Stack Depth $\mathcal{S}(x,y)$ of $\mathcal{A}(x,y)$

Further it is important to see, how much space on the stack is necessary, to compute $\mathcal{A}(x,y)$. Again we use a straight-forward implementation and get rid of all external dependencies by measuring stack space in terms of number of unfinished recursions rather than in bytes. Experiments yield the following data:

y

	$\mathcal{S}(x,y)$	0	1	2	3	4	5	6	7	8
	0	1	1	1	1	1	1	1	1	1
	1	2	3	4	5	6	7	8	9	10
x	2	4	6	8	10	12	14	16	18	20
	3	7	15	31	63	127	255	511	1023	2047
	4	16	2^{16}	...						

Again we could have obtained them easily by thinking a little bit:

$$\begin{aligned}
\mathcal{S}(0,y) &= 1 \\
\mathcal{S}(x+1,0) &= 1+\mathcal{S}(x,1) \\
\mathcal{S}(x+1,y+1) &= 1+\max(\mathcal{S}(x+1,y),\mathcal{S}(x,\mathcal{A}(x+1,y)))
\end{aligned}$$

Solving these equations for various x yields:

$$\begin{array}{rcl}
\mathcal{S}(0,y) = & 1 & \\
\mathcal{S}(1,y) = & y+2 & =\mathcal{A}(1,y) \\
\mathcal{S}(2,y) = & 2(y+2) & =\mathcal{A}(2,y)+1 \\
\mathcal{S}(3,y) = & 2\cdot 2^{y+2}-1 & =\mathcal{A}(3,y)+2 \\
\mathcal{S}(x,y) = & \mathcal{A}(x,y)+(x-1) & \text{for all } x\geq 1
\end{array}$$

Proof.

$$\begin{array}{lll}
x=0\text{:} & \mathcal{S}(0,y)=1 \text{ by definition of the recursion for } \mathcal{S} & \\
x=1\text{:} & y=0\text{:}\ \mathcal{S}(1,0) & =1+\mathcal{S}(0,1)=1+1=2=y+2 \\
 & y\geq 1\text{:}\ \mathcal{S}(1,y) & =1+\max(\mathcal{S}(1,y-1),\mathcal{S}(0,\mathcal{A}(1,y-1))) \\
 & & =1+\max((y-1)+2,\mathcal{S}(0,(y-1)+2)) \\
 & & =1+\max(y+1,1)=1+(y+1) \\
 & & =y+2=\mathcal{A}(x,y)+(x-1) \\
x\geq 2\text{:} & y=0\text{:}\ \mathcal{S}(x,0) & =1+\mathcal{S}(x-1,1)=1+\mathcal{A}(x-1,1)+((x-1)-1) \\
 & & =\mathcal{A}(x,0)+(x-1)+(1-1)=\mathcal{A}(x,y)+(x-1) \\
 & y\geq 1\text{:}\ \mathcal{S}(x,y) & =1+\max(\mathcal{S}(x,y-1),\mathcal{S}(x-1,\mathcal{A}(x,y-1))) \\
 & & =1+\max(\mathcal{A}(x,y-1)+(x-1), \\
 & & \qquad\mathcal{A}(x-1,\mathcal{A}(x,y-1))+((x-1)-1)) \\
 & & =\max(\mathcal{A}(x,y-1)+1,\mathcal{A}(x,y))+(x-1) \\
 & & =\mathcal{A}(x,y)+(x-1)\quad\square
\end{array}$$

6. Automated Analysis of Space and Time Complexity

In this section we will show how the methods developed in [BL94] can be used to obtain an estimate of the complexity of the Ackermann-Function. To provide for maximal reading comfort we will use the notation of [BL94] and frequently mention corresponding definition numbers.

The *parameter space* [Def. 2.1] $\mathcal{F}$ of $\mathcal{A}(x,y)$ is $\mathbb{N}_0\times\mathbb{N}_0$ and the set $\mathcal{F}_0$ of *terminating values* [Def. 2.1] is $\{(0,y)|y\in\mathbb{N}_0\}$.

$\mathcal{A}(x,y)$ is *well defined.* [Def. 2.2], as x is monotonically decreasing during the recursion. Where x remains constant, y decreases, which in due time leads to a decrement in x. For $x=0$ the recursion terminates.

Depending on the value (x, y) the set of *direct successors* $\mathcal{R}(x, y)$ [Def. 2.3] has quite a different form:

$$\mathcal{R}(x,y) = \begin{cases} \emptyset & \text{if } x = 0 \\ \{(x-1,1)\} & \text{if } x > 0, y = 0 \\ \{(x,y-1),(x-1,\mathcal{A}(x,y-1))\} & \text{if } x > 0, y > 0 \end{cases}$$

Similary the set of *necessary parameter values* $\mathcal{R}^*(x, y)$ [Def.2.4] is defined depending on (x, y).

$$\mathcal{R}^*(x,y) = \begin{cases} \emptyset & \text{if } x = 0 \\ \left\{(x',y')| \begin{array}{l} (x' < x \text{ or } (x' = x \text{ and } y' < y)) \text{ and} \\ (x',y') \neq (0,0) \text{ and } \mathcal{A}(x',y') \leq \mathcal{A}(x,y) \end{array} \right\} & \text{if } x > 0 \end{cases}$$

Looking at the definition of the *recursion depth* recdep[Def. 2.6] in [BL94] and comparing it to the stack depth $\mathcal{S}$ used above it is obvious that:

$$\text{recdep}(x,y) = \mathcal{S}(x,y) - 1$$

and thus

$$\begin{array}{ll} \text{recdep}(0,y) = 0 & \text{for } y \in \mathbb{N}_0 \\ \text{recdep}(x,y) = \mathcal{A}(x,y) + x - 2 & \text{for } x > 0,\ y \in \mathbb{N}_0 \end{array}$$

Therefore the parameter space is divided into *equivalence classes* $\mathcal{F}_k$ [Def. 2.5] such that

$$\begin{array}{ll} \mathcal{F}_0 = \{(0,y)|y \in \mathbb{N}_0\} & \\ \mathcal{F}_k = \{(x,y)|\mathcal{A}(x,y) + x - 2 = k;\ \ x,y \in \mathbb{N}_0\} & \text{for } k > 0 \end{array}$$

The easiest way to define $(x_1, y_1) \prec (x_2, y_2)$ [Def. 2.7] would be using the simple condition

$$(x_1,y_1) \prec (x_2,y_2) \Leftrightarrow \text{recdep}(x_1,y_1) < \text{recdep}(x_2,y_2)$$

This is equivalent to

$$(x_1,y_1) \prec (x_2,y_2) \Leftrightarrow \begin{cases} \text{True} & \text{if } x_1 = 0, x_2 = 0 \\ \text{False} & \text{if } x_1 = 0, x_2 > 0 \\ \mathcal{A}(x_1,y_1) + x_1 < \mathcal{A}(x_2,y_2) + x_2 & \text{if } x_1 > 0, x_2 > 0 \end{cases}$$

This does make $\mathcal{A}(x, y)$ a *monotonical recursive procedure* [Def. 2.7]. Unfortunately it does not make it *time monotonical* [Def. 5.1].

Obviously the space effort for the declaration part [Def. 4.1] for each step in the recursion does not depend on the actual parameter value. Therefore $\mathcal{D}(x, y) = \sigma_d$, const.

The *maximum successor* $\mathcal{N}(x, y)$ [Def. 4.3] encountered in the computation of $\mathcal{A}(x, y)$ is

$$\mathcal{N}(x,y) = \begin{cases} \text{undefined} & \text{if } x = 0 \\ (x-1,1) & \text{if } x > 0, y = 0 \\ (1,y-1) & \text{if } x = 1, y > 0 \\ (x-1,\mathcal{A}(x,y-1)) & \text{if } x > 1, y > 0 \end{cases}$$

Proof.

Trivial for $x = 0$ or $y = 0$.

$$\begin{aligned}
x = 1, y > 0 : \mathcal{R}(x,y) &= \{(x, y-1), (x-1, \mathcal{A}(x, y-1))\} = \\
&= \{(1, y-1), (0, \mathcal{A}(1, y-1)\} \\
&\quad \mathrm{recdep}(1, y-1) = (y-1) + 1 = y > 0 = \mathrm{recdep}(0, \mathcal{A}(1, y-1)) \\
x > 1, y > 0 : \mathcal{R}(x,y) &= \{(x, y-1), (x-1, \mathcal{A}(x, y-1))\} \\
&\quad \mathrm{recdep}(x, y-1) = \mathcal{A}(x, y-1) + (x-2) \\
&\qquad \leq \mathcal{A}(x, y-1) + (x-2) + (x-2) \\
&\qquad = (x-1) + \mathcal{A}(x, y-1) + (x-1) - 2 \\
&\qquad < \mathcal{A}(x-1, \mathcal{A}(x, y-1)) + (x-1) - 2 \\
&\qquad = \mathrm{recdep}(x-1, \mathcal{A}(x, y-1)) \quad \square
\end{aligned}$$

As $\mathcal{D}(x,y)$ [Def. 4.3] is constant (for any reasonable implementation) $\mathcal{A}(x,y)$ is trivially *space-monotonical* [Def. 4.4].

As mentioned before $\mathcal{A}(x,y)$ is not *time monotonical* [Def. 5.2] if the above definition of " $\prec$ " is used. This can best be seen using a simple example.
Let $(x_1, y_1) = (1, 13)$ and $(x_2, y_2) = (3, 1)$. As $\mathrm{recdep}(1, 13) = 14 = \mathrm{recdep}(3, 1)$ and (for all reasonable implementations) $\tau(1, 13) = \tau(3, 1)$ we find that $(1, 13) \approx (3, 1)$ and $(1, 13) \sqsubseteq (3, 1)$ as well as (for reasons of symmetry) $(1, 13) \sqsupseteq (3, 1)$ [Def. 5.1].
Now $\mathcal{R}(1, 13) = \{(1, 12), (0, 14)\}$ and $\mathcal{R}(3, 1) = \{(3, 0), (2, 5)\}$.
As expected $\mathrm{recdep}(1, 12) = \mathrm{recdep}(2, 5) = 13 = 14 - 1$, $\tau(1, 12) = \tau(2, 5)$ and therefore $(1, 12) \sqsubseteq (2, 5)$ and $(1, 12) \sqsupseteq (2, 5)$. Unfortunately $\mathrm{recdep}(0, 14) = 1 \neq 6 = \mathrm{recdep}(3, 0)$ and thus $(0, 14) \not\sqsupseteq (3, 0)$. This is in violation of the requirement for time monotonicity $(x_1, y_1) \sqsubseteq (x_2, y_2) \Rightarrow (x'_{1,k}, y'_{1,k}) \sqsubseteq (x'_{2,k}, y'_{2,k})$ for all k.

References

[BL94] Johann Blieberger and Roland Lieger. Worst-case space and time complexity of recursive procedures. (to appear), 1994.