Mathematics for Computer Scientists 2 (G52MC2)

L10 : Primitive recursion

Thorsten Altenkirch

School of Computer Science
University of Nottingham

November 17, 2009

What is the fastest growing function?

- Given $f, g : \mathbb{N} \to \mathbb{N}$ we say f grows faster than g ($f \succ g$), if

$$\exists n : \mathbb{N}, \forall i : \mathbb{N}, i \geq n \to f\ i > g\ i$$

- For example:

$$\begin{aligned} f_0\ n &= \mathrm{S}\ n \\ f_1\ n &= n + n \\ f_2\ n &= nn \end{aligned}$$

$$f_2 \succ f_1 \succ f_0$$

- Do you know a function which grows faster than f_2 ?
- Exponentiation ($f_3 \succ f_2$) :

$$f_3\ n = n^n$$

- Can we do better?
- Is there a function which grows faster than any function we can define by *primitive recursion*?

Primitive recursion

- Given a function $f : \mathbb{N} \to \mathbb{N}$ and $n, m : \mathbb{N}$ we define it's n-fold repetetion:

$$f^n\, m = \underbrace{f\,(f\, \ldots (f}_{n\,\text{times}}\; m) \ldots)$$

- More formally:

$$\begin{aligned} f^0\, m &= m \\ f^{(\mathrm{S}\, n)}\, m &= f\,(f^n\, m) \end{aligned}$$

- Defining addition, multiplication and exponentiation using repetition:

$$\begin{aligned} m + n &= \mathrm{S}^m\, n \\ m \times n &= (n+)^m\, 0 \\ &= (\lambda i : \mathbb{N}, n + i)^m\, 0 \\ m^n &= (m\times)^n\, 1 \end{aligned}$$

Primitive recursion

- Following the same scheme we define superexpontiation:

$$\mathrm{super}\, m\, n = (\lambda i : \mathbb{N}, n^i)^m\, n$$

- This allows us to define:

$$f_4\, n = \mathrm{super}\, n\, n$$

which grows faster than exponentiation: $f_4 \succ f_3$.

- Functions definable using repetetion are called **primitive recursive**.

Ackermann's function

- Ackermann (a student of Hilbert) defined the following function:

$$
\begin{array}{rcl}
\mathrm{ack} & : & \mathbb{N} \to \mathbb{N} \to \mathbb{N} \\
\mathrm{ack}\, 0\, n & = & \mathrm{S\, n} \\
\mathrm{ack}\, (\mathrm{S}\, m)\, n & = & (\mathrm{ack}\, m)^{(S\, n)}\, 1
\end{array}
$$

- What does ack compute?

$$
\begin{array}{rcl}
\mathrm{ack}\, 0\, n & = & n + 1 \\
\mathrm{ack}\, 1\, n & = & n + 2 \\
\mathrm{ack}\, 2\, n & = & 2 * n + 3 \\
\mathrm{ack}\, 3\, n & = & 2^{(n+3)} - 3 \\
\mathrm{ack}\, 4\, n & = & \underbrace{2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}}_{n+3} - 3
\end{array}
$$

Ackermann's function

- We define $f_\omega : \mathbb{N} \to \mathbb{N}$ as

$$f_\omega\, n = \mathrm{ack}\, n\, n$$

- How many values of f_ω can you calculate?

$$\begin{aligned}
f_\omega\, 1 &= 1 \\
f_\omega\, 1 &= 3 \\
f_\omega\, 2 &= 7 \\
f_\omega\, 3 &= 61 \\
f_\omega\, 4 &= 2^{2^{2^{65536}}} - 3 \\
&\ldots
\end{aligned}$$

- **Theorem:** f_ω grows faster than any primitive recursive function.