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 \( f, g : \mathbb{N} \rightarrow \mathbb{N} \) we say \( f \) grows faster than \( g \) (\( f \succ g \)), if
  \[ \exists n : \mathbb{N}, \forall i : \mathbb{N}, i \geq n \rightarrow f(i) > g(i) \]

- For example:
  \[
  \begin{align*}
  f_0(n) &= S(n) \\
  f_1(n) &= n + n \\
  f_2(n) &= nn
  \end{align*}
  \]

  \[ f_2 \succ f_1 \succ f_0 \]

- 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 \textit{primitive recursion}?
Given a function \( f : \mathbb{N} \rightarrow \mathbb{N} \) and \( n, m : \mathbb{N} \) we define its \( n \)-fold repetition:

\[
f^n m = f ( f \ldots ( f \ m) \ldots )
\]

More formally:

\[
f^0 m = m \\
f^{(S \ n)} m = f (f^n m)
\]

Defining addition, multiplication and exponentiation using repetition:

\[
m + n = S^m n \\
m \times n = (n+)^m 0 \\
= (\lambda i : \mathbb{N}, n + i)^m 0 \\
m^n = (m \times)^n 1
\]
Following the same scheme we define superexponentiation:

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

This allows us to define:

$$f_4 n = \text{super } n n$$

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

Functions definable using repetition are called **primitive recursive**.
Ackermann’s function

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

\[
\text{ack} : \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}
\]

\[
\text{ack } 0 \ n = S \ n
\]

\[
\text{ack } (S \ m) \ n = (\text{ack } m)^{(S \ n)} \ 1
\]

- What does \( \text{ack} \) compute?

\[
\text{ack } 0 \ n = n + 1
\]

\[
\text{ack } 1 \ n = n + 2
\]

\[
\text{ack } 2 \ n = 2 \times n + 3
\]

\[
\text{ack } 3 \ n = 2^{(n+3)} - 3
\]

\[
\text{ack } 4 \ n = 2^{2^{\cdots^{2}}_{n+3}} - 3
\]
Ackermann’s function

- We define $f_\omega : \mathbb{N} \rightarrow \mathbb{N}$ as
  
  $$f_\omega n = \text{ack } n n$$

- How many values of $f_\omega$ can you calculate?

  
  \begin{align*}
  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{align*}

- **Theorem**: $f_\omega$ grows faster than any primitive recursive function.