1 Fast-Growing Functions

Consider the following primitive recursive function:

\[ F_0(n) = n + 1 \]

\[ F_{i+1}(n) = F_i(F_i(...F_i(n)...)) \]

Or

\[ F_{i+1}(n) = F_i^{n+1}(n) \]

Consider a few values:

\[ F_1(n) = n + (n + 1) > 2n \]

\[ F_2(n) > 2^{n+1} > 2^n \]

\[ F_3(n) > 2^{2^{2^n}} \]

where the stack of 2s has a height of \( n + 1 \).

These functions are growing very fast. Using them, we can define the Ackermann function:

\[ A(n) = F_0(n) \]

We now define “eventually dominates”. \( G(n) \) eventually dominates \( f(n_1...n_k) \) if \( \exists N_0 \) such that \( \forall n_1...n_k G(max(n_1...n_k,N_0)) > f(n_1...n_k) \). It can be shown that the Ackermann function eventually dominates each \( F_i \). (See the handwritten notes.)

Theorem: If \( f(n_1...n_k) \) is primitive recursive, then \( \exists i \) such that \( F_i \) dominates \( f \).

(Note that this exactly characterizes the growthrate of primitive recursive functions and of the time to compute primitive recursive functions.)

Proof (Sketch?): We prove this by induction, first showing that all base cases (\( Z() \), \( S(n) \), and \( \Pi^0_1 \)) are dominated by \( F_1 \). For the inductive step, assume \( F_i \) eventually dominates \( f(\bar{n}) = g(h_i(\bar{n})...h_l(\bar{n})) \).

\[ f(\bar{n}) < F_i(F_i(max(\bar{n}))) \] where \( F_i(max(\bar{n})) \) is less than \( h_j(\bar{n}) \) by the inductive hypothesis, which is less than \( F_{i+1}(max(\bar{n})) \)
\[ f(0, \vec{n}) = g(\vec{n}) \]
\[ f(m + 1, \vec{n}) = h(m, f(m, \vec{n}), \vec{n}) \]

By hypothesis: \( g \) and \( h \) are eventually dominated by some \( F_i \). \( f \) is eventually dominated by some \( F_{i+1} \).

\( f(m, \vec{n}) = h(m - 1, h(m - 2, \ldots g(n)) \) where there are \( m + 2 \) applications of \( h \) hidden in the ellipsis.

As a corollary, the Ackermann function is not primitive recursive. It is not dominated by any primitive recursive function.

2 Time Hierarchy Theorem

If

\[ \lim_{n \to \infty} \frac{T_1(n)}{T_2(n + 1)} \to \infty \]

then there are functions computable with runtime \( T_1(n) \) but not computable with runtime \( T_2(n) \).

3 Busy Beaver

\( BB(n) \) is the max \( m \) such that there exists a turing machine \( M \) with less than \( n \) states such that \( M() \) runs for \( m \) steps and halts. The alphabet is fixed to \( \Sigma = \{0\} = \Gamma \).

Theorem: Given any total recursive function, \( BB(n) \) eventually dominates it.

4 First Order Logic

Theorem: The validity of a first order logic sentence is undecidable.

We begin by observing that well-formed formulas in first order logic can be written on turing machine tapes. Checking the well-formedness is easy.

Proof to be continued next class.